# Populating the Semantic Web

**Kristina Lerman**[1], **Cenk Gazen**[2,3], **Steven Minton**[2] **and Craig Knoblock**[1]
1. USC Information Sciences Institute
2. Fetch Technologies
3. Carnegie Mellon University
{lerman,knoblock}@isi.edu
{gazen,minton}@fetch.com

**Abstract**

The vision of the Semantic Web is that a vast store of online information "meaningful to computers will unleash a revolution of new possibilities". Unfortunately, the vast majority of information on the Web is formatted to be easily read by human users, not computer applications. In order to make the vision of the Semantic Web a reality, tools for automatically annotating Web content with semantic labels will be required. We describe the ADEL system that automatically extracts records from Web sites and semantically labels the fields. The system exploits similarities in the layout of Web pages in order to learn the grammar that generated these pages. It them uses this grammar to extract structured records from these Web pages. ADEL system also exploits the fact that sites in the same domain will provide the same, or similar data. By collecting labeled examples of data during the training stage, we are able to learn structural descriptions of data fields and later use these descriptions to semantically label new data fields. We show that on a Used Car shopping domain, ADEL achieves precision of 64% and recall of 89% on extracting and labeling data columns.

## Introduction

The vision of the Semantic Web is that a vast store of online information "meaningful to computers will unleash a revolution of new possibilities" (Berners-Lee, Hendler, & Lassila 2001). Unfortunately, the vast majority of information on the Web is formatted to be easily read by human users, not computer applications. Semantic markup, such as XML and DAML (DAML Project ), will enable the exchange of information between applications; however, only a tiny fraction of the Web pages are currently annotated with semantic markup. This situation in not likely to improve soon, because there is little incentive for information providers to convert existing content to the new standard. Moreover, even where semantically annotated sources exist, not all information providers will conform to the same schema. Clearly, in order to make the vision of the Semantic Web a reality, tools for converting regular Web content into semantically annotated one will be required.

Web wrappers were developed to facilitate the extraction of structured information from Web sources (Muslea, Minton, & Knoblock 2001; Kushmerick, Weld, & Doorenbos 1997; Freitag & Kushmerick 2000; Cohen & Jensen 2001). However, these methods are a poor choice from the point of view of automatically creating semantically meaningful content. Wrappers are notoriously brittle — any change in the source requires a new wrapper to be generated. A bigger problem is that wrapper creation demands a significant amount of user intervention — in order to learn the correct wrapper, the user must provide the data schema and example records to the wrapper induction system. Because user involvement is so costly, the focus recently has been on minimizing the number of examples the user has to label (Muslea, Minton, & Knoblock 2002). Still, even when user effort is significantly reduced, the amount and the rate of growth of information on the Web will quickly require fully automatic solutions.

Fortunately, on-line information sources contain much explicit and implicit structure, both in layout and content, that can be exploited for the purposes of automatic information extraction. For example, Web sites that dynamically generate pages from electronic databases in response to user queries, e.g., on-line catalogs, directories, are highly uniform, both in terms of how the information is organized, visually presented and the type of information that is displayed. Recently, a number of approaches (Lerman, Knoblock, & Minton 2001; Crescenzi, Mecca, & Merialdo 2001b; Arasu & Garcia-Molina 2003) have focused on exploiting structure within a page in order to automatically extract records from it.

The difficulties inherent in automatic record extraction are even more pronounced in the problem of semantically labeling the extracted fields. Despite the Web wrappers' long track record, automatic labeling of extracted data has only recently begun to be addressed (Arlotta *et al.* 2003). The current paper describes a domain independent approach to automatically extracting records from Web sites and semantically labeling the fields, or mapping them to a schema. The work is built on our previous research in automatic generation and maintenance of Web wrappers (Lerman, Minton, & Knoblock 2003). Our starting assumption is that content on different Web sites in the same domain contains many similarities: e.g., used car sites generally give the year, make and model of the car, as well as its mileage and price. Additional information, such as the color and engine specifications may also be given. The data fields will have a similar format on different sites within the domain. If we learn the representation of fields on one site, we can apply it to label content on other sites within the domain. In order to automatically extract records from a site, we exploit similarities in the layout of data pages. Our results on the Used Car domain indicate our approach is a feasible method for automatically extracting and labeling data.

## Related Work

Researchers have addressed the problem of extracting data from Web tables; however, very little work on the semantic labeling of the extracted content has been made. At the other end of the spectrum are the database and information integration researchers, who done extensive work on the problem of schema matching and integration. This work is similar to our research as discussed below.

Existing approaches to extracting data from Web tables can be classified as heuristic (Chen, Tsai, & Tsai 2000) or machine learning (Borkar, Deshmukh, & Sarawagi 2001; Hurst 2001; Wang & Hu 2002). Heuristic approaches to detect tables and record boundaries in Web documents by using the DOM or domain-specific rules features such as percentages, date/time, etc. (Chen, Tsai, & Tsai 2000) to identify tables. Machine-learning approaches learn a model of data from a set of labeled training examples using hand-selected features. These include Hidden Markov-based probabilistic models (Borkar, Deshmukh, & Sarawagi 2001), Naive Bayes classifiers (Hurst 2001), and domain-independent classifiers that use non-text layout (average number of columns/rows, average cell length and consistency) and content features (image, form, hyperlink, alphabetic, digit, others) (Wang & Hu 2002). These works require many training examples in order to identify tables on HTML pages, and in a few cases correctly extract data from them. No attempt is made in these works to assign labels to data.

Researchers made several attempts to exploit the layout of Web pages for the purpose of automatically extracting data from them (Crescenzi, Mecca, & Merialdo 2001b; 2001a; Arasu & Garcia-Molina 2003). The RoadRunner system (Crescenzi, Mecca, & Merialdo 2001b; 2001a) is an example of such a system. The premise behind RoadRunner is that many Web pages are generated by a grammar, which can be inferred from example pages. Thus, RoadRunner can learn the table template and use it to automatically extract data from the Web site, as we do. RoadRunner's focus is on a subclass of Web pages that can be generated by union-free grammar. The learning algorithm is exponential, and further simplifications are necessary to keep it computationally tractable. Disjunctions are necessary to represent alternative layout instructions often used by Web sites for a same field. In addition to extracting data, RoadRunner researchers have made efforts to extract column labels from HTML tables (Arlotta *et al.* 2003).

Our work is very similar to schema matching or integration (Rahm & Bernstein 2001; Doan, Domingos, & Halevy 2003), where the object is to produce semantic mapping to map instances of data from one scheme to another. This is an important task in information integration, where queries that are expressed in some common language have to be translated to the local schema of the database before they can be submitted to the database. Past works on schema matching (Li & Clifton 2000; Doan, Domingos, & Halevy 2001; 2003) included machine learning techniques that learn to classify new object instances based on features that include local schema names and content features. The content features used in these works are global in nature, such as word frequencies and format. Our approach, on the other hand, uses finer-grained descriptions enabled by the use of patterns to describe the structure of data.

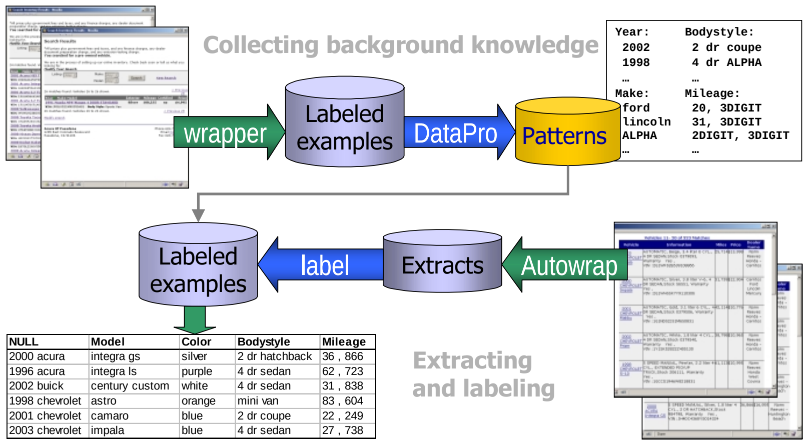
## Automatic Data Extraction and Labeling

The architecture of the Automatic Data Extraction and Labeling (ADEL) system is shown in Figure 1. We will use the Used Cars shopping site as the running example. The training stage consists of background knowledge acquisition, where we collect data in a particular domain, e.g., Used Cars, and learn a structural description of it. In order to collect a large number of examples, we created wrappers for some sites in the domain. We specified the schema of the data and labeled example records on several pages. We used the wrapper building tool developed by Fetch Technologies. It provides a GUI to simplify data entry and uses the wrapper induction system (Muslea, Minton, & Knoblock 2001) to learn the correct wrapper extraction rules.

Next, we learn a description of the data fields. We represent the structure of a data field by sequences of tokens and token types, which we call *patterns*. We use the DataPro algorithm (Lerman, Minton, & Knoblock 2003) to learn patterns from examples collected by the wrappers, as described in Section Modeling Data Content.

The second stage of ADEL consists of analyzing HTML pages from a new site in the domain, extracting records from them and labeling fields. Data intensive Web sites have a surprisingly uniform structure that we can exploit for the purpose of automatic information extraction. The entry point is an index page or HTML form for the user to input her query. The result of the query is a list of items or a collection of records from a database. The results are usually displayed as a list or a table on an automatically generated page. We call such a results page the *list page*. Each item or record often has a link to a *detail page* that contains additional information about that entry. Detail pages are also generated automatically and populated with results of database queries. We envision that the user will provide a pointer to the top-level page — index page or a form — and the system will automatically navigate the site, retrieving all pages. Currently, ADEL is not able to process forms; instead, we manually retrieve list pages and provide them to the system.

The next step is to extract data from pages. The pages have to be classified into separate types, such as list or detail pages, before data can then be extracted from pages of a single type. In the experiments described in this paper, we



```
Year:        Bodystyle:
 2002         2 dr coupe
 1998         4 dr ALPHA
 …            …
Make:        Mileage:
 ford         20, 3DIGIT
 lincoln      31, 3DIGIT
 ALPHA        2DIGIT, 3DIGIT
 …            …
```

| NULL | Model | Color | Bodystyle | Mileage |
|---|---|---|---|---|
| 2000 acura | integra gs | silver | 2 dr hatchback | 36 , 866 |
| 1996 acura | integra ls | purple | 4 dr sedan | 62 , 723 |
| 2002 buick | century custom | white | 4 dr sedan | 31 , 838 |
| 1998 chevrolet | astro | orange | mini van | 83 , 604 |
| 2001 chevrolet | camaro | blue | 2 dr coupe | 22 , 249 |
| 2003 chevrolet | impala | blue | 4 dr sedan | 27 , 738 |

Figure 1: Architecture of the Automatic Data Extraction and Labeling system

skip this step and provide several pages of the same type to the algorithm. Autowrap, the automatic data extraction algorithm is described in Section Data Extraction.

Finally, we use the patterns learned on the training examples to assign semantic labels to the automatically extracted records, as described in Section Labeling. We validated the ADEL system on the Used Cars domain. Results of this work are presented in the results section.

## Modeling Data Content

The data modeling step is used to learn the structure of data fields from examples. We represent the structure of data by patterns of tokens and token types. In previous work, we developed a flexible pattern language and presented an efficient algorithm, DataPro, for learning patterns from examples of a field (Lerman & Minton 2000; Lerman, Minton, & Knoblock 2003). The pattern language contains specific tokens and general token types. Specific types refer to unique text strings, such as “California” or “sea”, while the general types describe the syntactic category to which the token’s characters belong, such as numeric, alphabetic, etc.The token types are organized in a hierarchy, which allows for multi-level generalization.[1] The pattern language can be extended to include other syntactic types or domain-specific semantic types. In addition to patterns, we also remember the mean length of a field and its variance.

DataPro algorithm finds patterns that describe many of the examples of a field and are highly unlikely to describe a random token sequence. As an example, names can be represented as a set of patterns such as “capitalized word followed by an initial” and “capitalized word followed by a capitalized word,” whereas addresses can be represented as “number followed by two capitalized words followed by the word Blvd,” etc.

The symbolic representation of content by patterns of tokens and token types is very flexible and general. In our previous research we found that a set of patterns describing how a field begins and ends, allowed us monitor wrapper’s accuracy or locate examples of the field on new pages with considerable accuracy (Lerman, Minton, & Knoblock 2003). In Section Labeling we show how to apply patterns to recognize known data fields.

## Data Extraction

As we discussed above, many Web sites that present information contained in databases follow a *de facto* convention in displaying information to the users and allowing them to navigate it. This convention affects how the Web site is organized, and gives us additional information we can leverage for information extraction. Such Web sites generate list and detail pages dynamically from templates and fill them with results of database queries.

Consider a typical list page from a Web site. As the server constructs the page in response to a query, it generates a header, followed in many cases by an advertisement, then possibly a summary of the results, such as “`Displaying 1-10 of 214 records.`”, table header and footer, followed by some concluding remarks, such as a copyright information or navigation aids. We call this part of the page the *page template*. The page template of a list page contains data that is shared by all list pages and is invariant from page to page. The page template can also be thought of as the grammar that generates the pages. Given two, or preferably more, example list pages from a site, the Autowrap algorithm can derive the grammar used to generate the pages and use it to extract data from them.

Autowrap uses an *ad hoc* method to induce a grammar for the Document Object Model (DOM) trees of the pages. The kinds of grammars it induces allow us to extract single data items as well as lists, where the rows of a list contain data items or nested lists.

[1]A text token is a punctuation mark (PUNCT) or an alphanumeric token (ALNUM). If it is alphanumeric, it could be alphabetic type (ALPHA) or a number (NUMBER). If alphabetic, it could also be a capitalized word (CAPS) or an all-capitalized word (ALLCAPS). The number category is further subdivided into 1DIGIT to 5DIGIT numbers.

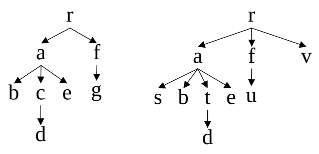


Figure 2: Example trees

The induction algorithm has two main stages: Finding repeating sub-structures and merging grammars into a more general one. Both stages use ideas based on templates.

A template is a sequence of alternating slots and stripes where the stripes are the common sub-structures among all the pages and slots are the placeholders for pieces of data that go in between the stripes. One way to find the template of a set of pages is to find the longest common subsequence (LCS) of all the pages. The LCS immediately gives the stripes of the template and with a little bookkeeping, the slots can also be found.

The template idea can easily be extended to trees, and in particular to the DOM structure. Given a set of sequences of DOM elements, we find the LCS and then for each element in the LCS, we recursively apply the algorithm to the set of child elements. For example, consider two trees **r(a(b c(d) e) f(g))** and **r(a(s b t(d) e) f(u) v)** where the parenthesis group the children of the preceding element, as shown in Figure 2. First we find the LCS of the single-element sequences **[r]** and **[r]** and then proceed down to the child sequences **[a f]** and **[a f v]**. The LCS of these two is **[a f]**, so we first recurse down to the child sequences of the two **a** nodes and then to those of the two **f** nodes to get **[b e]** and **[]**. Since there are no more child sequences, we combine the LCS's to get a tree template: **r(a(b e) f())**.

Once we have the template, we can use it to extract the slots by finding the nodes in the original DOM structures that are not in the stripes of the template. This gives us **.(.(. c(d) .) .(g))** and **.(.(s . t(d) .) .(u) v)**. With a little more work, we can align the data in columns and represent the data in a table:

| | | | |
|---|---|---|---|
| | c(d) | g | |
| s | t(d) | u | v |

The template can also be used as a similarity measure between two DOM sub-structures, since similar structures will have templates with bigger stripes than those that are not similar. The first step of the induction algorithm uses this similarity measure to find repeating sub-structures among the children of a DOM node. In particular, the algorithm looks for consecutive sequences of nodes whose similarity measure is above a threshold. The sequences, which are assumed to be the rows of a list, are used to induce a row template. As the algorithm traverses up from the leaves towards the root, it replaces the repeating sub-structures with the row templates. For example, starting with the tree **r(a(b(c 1) b(c 2)) d a(b(c 3) b(c 4) b(c 5)) d)**, the algorithm first finds row templates for the inner lists within the children of **a** nodes: **r(a([b(c)]*) d a ([b(c)]*) d)** where the notation **[...]*** represents a row template. Next, the algorithm examines the children of **r** (in the intermediate tree). With repeating substructures represented as templates, the algorithm can detect the similarity between the two rows of the outer list and find a row template for the children of **r**: **r([a([b(c)]*) d]*)**.

The second step of the induction algorithm merges the templates that are induced for each DOM tree. The merging algorithm is the same as the template-finding algorithm, except the row templates are treated slightly differently: The order of the nodes within row templates may be rotations of one another even though the lists represented by the templates are very similar. For example, **r(a b c a b c a b c)** and **r(b c a b c a b c)** will give **r([a b c]*)** and **r([b c a]*)** as templates. The merging algorithm treats this special case and chooses the best alignment among the rotations.

## Labeling

When provided with several list pages, Autowrap extracts all tables from these pages. These include the one with the data we are interested in extracting, as well as tables containing extraneous information. The next step in the process is to label the columns of every table and output the correct table of data, which we define to be one that has the most labeled columns.

We use learned patterns to map columns to data fields. The basic premise is to check how well a field describes a column of data, given a list of patterns that describe the data field and its mean length. We have developed a set of heuristics to score how well a field describes a column. The column is assigned the field with the highest score. Factors that increase a field's score include

- Number of patterns that match examples in the column
- How close examples are in length to the field's mean length
- Pattern weight — where the more specific patterns are given higher weight

## Results

We validated the ADEL system on the Used Cars domain. We wrapped two used cars sites — Anaheim Lincoln Mercury and Mercedes Benz of Laguna Niguel — and collected on the order of $250$ records from these sites. We normalized all data by lowercasing it. We then ran the DataPro algorithm on the records to learn descriptions of the fields. The resulting patterns and field lengths are displayed in Table 1. Fields Mileage and Price had many specific patterns that we do not display in the table.

Next, we attempted to extract and label data from three new sites in the Used Cars domain.

We manually collected three list pages from each new site. Autowrap automatically induced the template for each set of three pages and extracted all data from them. Autowrap found, on average, six tables of data on each page. These tables were processed further. Again, we normalized data by lowercasing it. In addition, we fixed some of the segmentation errors Autowrap made. For instance, Autowrap does not recognize the sequence "`\\r\\n`" or a comma as a field delimiter. Thus, "Marina del Rey, CA" is extracted as a single field, rather than two. These problems will eventually be

| | | |
|---|---|---|
| **Year** | **Color** | **Bodystyle** |
| $< 1.0 \pm 0.0 >$ | $< 1.12 + -0.32 >$ | $< 3.06 \pm 0.24 >$ |
| [1999] | [smoke silver] | [4 dr sedan] |
| [2002] | [desert silver] | [4 dr ALPHA ALPHA] |
| [2000] | [silver] | [2 dr coupe] |
| [2003] | [black] | |
| [2001] | [grey] | |
| [4DIGIT] | [ALPHA] | |
| **Make** | **Mileage** | **Engine** |
| $< 1.31 \pm 0.46 >$ | $< 2.38 \pm 0.92 >$ | $< 3.83 \pm 0.38 >$ |
| [mercedes, benz] | [47K] | [5 . 0l v8] |
| [ford] | [21 , 3DIGIT] | [4 . 3l v8] |
| [mercury] | [20 , 3DIGIT] | [3 . 2l 6cyl] |
| [lincoln] | [2DIGIT , 3DIGIT] | [3 . 2l ALNUM] |
| **Model** | **Price** | [2 . 3l] |
| $< 1.88 \pm 0.76 >$ | $< 4.0 \pm 0.0 >$ | [2 . ALNUM] |
| [s430] | [$ 2 , 988] | **VIN** |
| [ranger 2wd] | [$ 25 , 3DIGIT] | $< 1.0 \pm 0.0 >$ |
| [mountaineer ALNUM] | [$ 15 , 988] | [ALNUM] |
| [gr marquis ls] | . . . | |
| [ls v6] | [$ 29 , 3DIGIT] | |
| [ls v8] | [$ 30 , 988] | |
| [navigator 2wd] | [$ 31 , 3DIGIT] | |
| [navigator ALNUM] | [$ 2DIGIT , 995] | |
| [ALPHA lx] | [$ 2DIGIT , 990] | |
| [ALPHA ALPHA] | [$ 2DIGIT , 900] | |

Table 1: Patterns learned for data fields in the Used Cars domain

fixed in the Autowrap source. For now, we manually segmented fields on "`\\r\\n`" and "," (except for numbers, in which case we did not segment them on the comma).

The columns were scored against the patterns in Table 1 according to the criteria described in Section Labeling. The column was assigned the label of the highest scoring field. The table with the most labeled columns was output in comma-delimited spreadsheet format. Table 2 shows a section of a table of labeled data for one site.

In all, Autowrap extracted 27 columns of data from the three sites. In one site, the algorithm made a mistake in the list template, resulting in it improperly concatenating every two rows, thereby producing a table with 8 columns, rather than 4. The problem of correct list segmentation is addressed in a different work (Lerman *et al.* 2004), which shows how to improve Autowrap's results with additional information from detail pages. In this work we are concerned with labeling columns rather than extracting structured records.

Nineteen of the 27 columns were correctly labeled, 9 were incorrectly labeled, and two were unlabeled, resulting in precision and recall of $P = 0.64$ and $R = 0.89$.[2] However, five of the columns that were incorrectly labeled were not in the schema we created: e.g., , "Transmission", "Stock Number", "Warranty" and "Dealer" in Table 2. When these are excluded from the incorrectly labeled columns count, precision increases to $P = 0.80$.

[2]We define True Positives (TP) as correctly labeled columns; False Positive (FP) as incorrectly labeled columns; and False Negatives (FN) as unlabeled columns; therefore, $P = TP/(TP+FP)$ and $R = TP/TP + FN$.

## Discussion

We have described the ADEL system that automatically extracts data from HTML pages and labels it. Although the system is still under development, initial results show good performance on test sites in the Used Cars domain.

ADEL can extract data from HTML pages containing single records of data, lists of records, and even nested lists. It does so by inducing the grammar of the pages and using the grammar to extract data. In order to label the data, we first need to learn the structural descriptions of the data fields. We do this by accumulating labeled data from some sites in the domain, and used the learned descriptions to label data from new sites in the same domain.

ADEL system has shown good performance on labeling data on three new Web sites in the Used Cars domain. Of the 27 columns of data extracted from pages in these sites, 19 were correctly labeled and 2 were not labeled by the system. Five of the incorrectly labeled columns contained data that was not in the schema we created for the site, such as "Transmission", "Stock Number", "Warranty" and "Dealer". We can expand the schema by allowing user to specify patterns describing new data fields. We did this for the field "Transmission." The patterns for the field were `[automatic]` and `[5 speed manual]`. With the manually coded patterns, we were able to label the transmission field on the site. Our algorithm also consistently mislabeled the "Price" field. This was because without the $, "Price" and "Mileage" fields are very similar (two digit number followed by a three digit number). Autowrap did not always extract the $ in the price field.

The most significant challenge for automatic labeling appears to be inconsistent data formats. For example, we failed to correctly label the "Engine" field in Table 2 because is appeared on this site as "3.2 liter 6 cyl.", whereas we learned a description from sites that described engines as "3.2l 6cyl." Likewise, some sites presented Body Style as "4dr sedan", while others as "sedan, 4dr". Conceivably, we could include such data transformation rules into the labeling algorithm, thereby significantly improving performance of the system.

Unlike (Arlotta *et al.* 2003), we do not rely on the column labels provided in the HTML page to label the schema. Even though each site in the Used Cars domain provides similar information, the schema labels used were not always consistent with other sites. For example, one site applied the label "Vehicle" to data composed of fields "Year Make Model", and "Information" for all the details about the car, such as body style, color, engine, etc. Other differences included using "MSRP" rather than "Price" and "Exterior" rather than "Color." A scheme based on extracting column labels from HTML tables will not allow one to recognize that a field "MSRP" on one site provides the same information as the field "Price" on another site. Reading column labels will be useful for augmenting the schema by discovering new fields and labels.

| - | Model | Color | Color | Bodystyle | - | VIN | Mileage | Mileage | Model |
|---|---|---|---|---|---|---|---|---|---|
| 1998 acura | 2 . 5tl | automatic | silver | 4 dr sedan | 2.5 liter 5 cyl. | jh4ua265xwc007394 | 60 , 913 | 14 , 976 | norm reeves |
| 2001 acura | 3 . 2cl | automatic | red | 2 dr coupe | 3.2 liter 6 cyl. | 19uya42741a003823 | 76 , 674 | 19 , 934 | norm reeves |
| 2000 acura | 3 . 2tl | automatic | white | 4 dr sedan | 3.2 liter v-6 | 19uua5668ya054953 | 41 , 421 | 19 , 595 | norm reeves |
| 2000 acura | integra gs | 5 speed manual | silver | 2 dr hatchback | 1.8 liter 4 cyl. | jh4dc4368ys014324 | 36 , 866 | 16 , 895 | norm reeves |
| 1996 acura | integra ls | 5 speed manual | purple | 4 dr sedan | 1.8 liter 4 cyl. dohc | jh4db7553ts011459 | 62 , 723 | 9 , 595 | norm reeves |
| 1998 chevrolet | astro | automatic | orange | mini van | 4.3 liter v-6 | 1gndm19w0wb180216 | 83 , 604 | 8 , 903 | cerritos ford |
| 2001 chevrolet | camaro | automatic | blue | 2 dr coupe | 3.8 liter 6 cyl. | 2g1fp22k412109568 | 22 , 249 | 13 , 975 | norm reeves |

Table 2: Results

## Acknowledgements

This material is based upon work supported in part by the National Science Foundation under Award No. IIS-0324955, in part by the Defense Advanced Research Projects Agency (DARPA), through the Department of the Interior, NBC, Acquisition Services Division, under Contract No. NBCHD030010, in part by the Air Force Office of Scientific Research under grant number F49620-01-1-0053, in part by the United States Air Force under contract number F49620-02-C-0103, and in part by a gift from the Microsoft Corporation.

## References

Arasu, A., and Garcia-Molina, H. 2003. Extracting structured data from web pages. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*.

Arlotta, L.; Crescenzi, V.; Mecca, G.; and Marialdo, P. 2003. Automatic annotation of data extracted from large web sites. In *Proc. of the Sixth Int. Workshop on Web and Databases (WebDB03)*.

Berners-Lee, T.; Hendler, J.; and Lassila, O. 2001. The semantic web. *Scientific American*.

Borkar, V.; Deshmukh, K.; and Sarawagi, S. 2001. Automatic segmentation of text into structured records full text. In *Proc. of the 2001 ACM SIGMOD Int. Conf. on Management of data*, Special Interest Group on Management of Data, 175–186. ACM.

Chen, H.; Tsai, S.; and Tsai, J. 2000. Mining tables from large scale html texts. In *18th Int. Conf. on Computational Linguistics (COLING)*.

Cohen, W., and Jensen, L. 2001. A structured wrapper induction system for extracting information from semi-structured documents. In *Proc. of the IJCAI Workshop on Adaptive Text Extraction and Mining*.

Crescenzi, V.; Mecca, G.; and Merialdo, P. 2001a. Automatic web information extraction in the ROADRUNNER system. In *Proc. of the Int. Workshop on Data Semantics in Web Information Systems (DASWIS-2001)*.

Crescenzi, V.; Mecca, G.; and Merialdo, P. 2001b. ROADRUNNER: Towards automatic data extraction from large web sites. In *Proc. of the 27th Conf. on Very Large Databases (VLDB)*.

Doan, A.; Domingos, P.; and Halevy, A. Y. 2001. Reconciling schemas of disparate data sources: A machine-learning approach. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, 509–520.

Doan, A.; Domingos, P.; and Halevy, A. 2003. Learning to match the schemas of databases: A multistrategy approach. *Machine Learning Journal* 50:279–301.

Freitag, D., and Kushmerick, N. 2000. Boosted wrapper induction. In *Proc. of the 7th Conf. on Artificial Intelligence (AAAI-2000)*, 577–583.

Hurst, M. 2001. Layout and language: Challenges for table understanding on the web. In *In Web Document Analysis, Proc. of the 1st Int. Workshop on Web Document Analysis*.

Kushmerick, N.; Weld, D. S.; and Doorenbos, R. B. 1997. Wrapper induction for information extraction. In *Proc. of the Intl. Joint Conf. on Artificial Intelligence*, 729–737.

Lerman, K., and Minton, S. 2000. Learning the common structure of data. In *Proc. of the 15th National Conf. on Artificial Intelligence (AAAI-2000)*.

Lerman, K.; Getoor, L.; Minton, S.; and Knoblock, C. A. 2004. Using the Structure of Web Sites for Automatic Segmentation of Tables. to appear in *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*.

Lerman, K.; Knoblock, C. A.; and Minton, S. 2001. Automatic data extraction from lists and tables in web sources. In *Proc. of the workshop on Advances in Text Extraction and Mining (IJCAI-2001)*. Menlo Park: AAAI Press.

Lerman, K.; Minton, S.; and Knoblock, C. 2003. Wrapper maintenance: A machine learning approach. *Journal of Artificial Intelligence Research* 18:149–181.

Li, W., and Clifton, C. 2000. Semint: A tool for identifying attribute correspondence in heterogeneous databases using neural networks. *Data and Knowledge Engineering* 33:49–84.

Muslea, I.; Minton, S.; and Knoblock, C. A. 2001. Hierarchical wrapper induction for semistructured information sources. *Autonomous Agents and Multi-Agent Systems* 4:93–114.

Muslea, I.; Minton, S.; and Knoblock, C. 2002. Active + semi-supervised learning = robust multi-view learning. In *Proc. of the 19th Int. Conf. on Machine Learning (ICML 2002)*, 435–442. Morgan Kaufmann, San Francisco, CA.

DAML Project. http://www.daml.org.

Rahm, E., and Bernstein, P. 2001. On matching schemas automatically. *VLDB Journal* 10(4).

Wang, Y., and Hu, J. 2002. Detecting tables in html documents. In *Fifth IAPR Int. Workshop on Document Analysis Systems, Princeton, New Jersey*.