

# Filtering Web Pages with Application Ontologies

Li Xu

Department of Computer Science  
University of Arizona South  
lxu@email.arizona.edu

David W. Embley

Department of Computer Science  
Brigham Young University  
embley@cs.byu.edu

## Abstract

Automatically recognizing which HTML documents on the Web contain objects that are “of interest” for a user is non-trivial. As a step toward solving this problem for information filtering in which a user expresses a long-term need for information with a specific profile, we propose an approach based on ontological descriptions. The HTML documents we consider include semistructured HTML documents, HTML tables, and HTML forms. Given the keywords and values and kinds of values recognized by an ontological specification in an HTML document, we apply several heuristics: (1) a density heuristic that measures the percent of the document that appears to apply to the application ontology, (2) an expected-value heuristic that compares the number and kind of values found in the 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 an HTML document contains objects of interest with respect to an application ontology. Our experimental results show that we have been able to achieve about 95% for both recall and precision.

**Keywords:** Information filtering, application-ontology filtering, conceptual-model-based filtering, ontology specification, machine-learned classification, information retrieval.

# 1 Introduction

The World Wide Web contains abundant repositories of information in HTML documents—indeed, it contains so much that locating information entities that are “of interest” for a user becomes a huge challenge. Even sorting through a tiny subset of HTML documents for particular objects of interest is overwhelming. How can we automatically select just those documents that have the needed information for a user?

In this paper, we focus on the specialized subproblem called *information filtering (IF)*. IF attempts to solve the problem of information gathering for long-term needs of a particular user or user group. Typically, a user (or user group) needs information satisfying a particular query specification or profile. The filter checks a document set (e.g. new documents that come on line on the Web) and returns those that satisfy the query profile.

Search engines are not precise enough to filter documents on the Web. In order to find relevant Web documents that contain information of interest, users issue queries composed of keywords that express their interests. To evaluate the relevance of an HTML document to a user query, search engines mainly apply keyword-based techniques to filter the document set based on common terms that appear in both the user query and the Web page. The user query and the Web page, however, are typically constructed independently. As research in [1] shows, however, people use the same terms for the same concepts with a probability of less than 0.20. This fact helps explain the imprecision in results returned by search engines.

More general information retrieval (IR) techniques [2] typically do not solve the IF problem well either. Although IF and IR are two sides of the same coin [3], IR systems are usually designed to facilitate retrieving information units quickly for relatively short-term information needs of a diverse large group of users, whereas IF systems are commonly designed to personalize interests of a particular user or a group of users to support the users’ long-term needs. Nevertheless, some IR techniques may prove useful for IF. IR approaches like [4] and [5] that exploit semantics in document content by embedding semantic mark-ups in meta languages could potentially help solve the IF problem. Semantic annotation to mark up Web content, however, is still only being explored and there are no encouraging results that are good enough to be practical [6, 7].

Other researchers have attempted to more directly solve the filtering problem. SIFT [8] uses both a Boolean model and a VSM model for IF. Both models, however, rely on keywords, which results in the same problem search engines encounter. In other systems [9, 10, 11], instead of explicitly expressing user profiles, learning based techniques are applied to determine the information needs of users based on user input documents. These IF systems transform the contents of a set of documents to a concrete user query. The transformation is typically based on AI techniques, but the inherent complexity of the transformation problem makes it difficult to solve the learning task efficiently.

In this paper, we present an approach to IF that applies an application ontology to explicitly specify user interests. We base our approach on application ontologies [12], which are conceptual-model snippets [13, 14] of standard ontologies [15, 16], and we apply techniques from data extraction [12, 17, 18], information retrieval [2, 19], and machine learning [20]. By exploiting the content of HTML documents and using ontological specification in application ontologies, we construct automated processes to filter document sets to satisfy a user’s information needs.

We call documents that contain items of interest *relevant* documents and documents that do not contain items of interest *irrelevant* documents. For the automated processes to filter application objects in Web documents, we must be careful not to discard relevant documents and not to accept irrelevant documents. In order to measure the performance of the automated processes, we use popular metrics available in information retrieval systems. A process that discards too many relevant documents has poor *recall*—the ratio of the number of relevant documents accepted to the total number of relevant documents. A process that accepts too many irrelevant documents has poor *precision*—the ratio of the number of relevant documents accepted to the total number of documents accepted. The harmonic mean of the precision and recall, which is called an *F-measure*, is a standard way to combine precision and recall. We wish to have an automated recognition process that has a high F-measure, i.e. that has both high recall and high precision.

This paper presents our contribution to IF as follows. Section 2 contains some preliminaries and provides an example to which we refer throughout the paper to illustrate our ideas. Section 3 describes application ontologies, on which we base our IF work. Section 4 presents the high-level architecture of the IF framework we have built to recognize relevant HTML documents for application ontologies. Given an application ontology and a set of HTML documents, Section 5 explains how we automatically obtain threshold statistics for determining document relevance using a set of heuristics including: (1) a density heuristic, (2) an expected-values heuristic, and (3) a grouping heuristic. Section 6 provides an empirical evaluation of our approach including our experimental results, which—for the two applications we tried (car advertisements and obituaries)—each have an F-measure of 95%. Section 7 describes related work and more particularly compares it with our approach. Section 8 gives concluding remarks and our plans for future work.

## 2 Preliminaries

Before giving details of our approach, we first need to discuss our assumption about HTML documents, application ontologies, and the filtering task.

### 2.1 HTML Documents

The HTML documents on the Web we consider include semistructured HTML documents such as the lists of ads in Figure 1, HTML tables such as the one in Figure 2, and HTML forms such

as the one in Figure 3(a), which when filled-in and processed, yields a table or a semistructured list such as the one in Figure 3(b).

We assume that the HTML documents are data-rich and narrow in ontological breadth [12]. A document is *data rich* if it has a number of identifiable constants such as dates, names, account numbers, ID numbers, part numbers, times, currency values, and so forth. A document is *narrow in ontological breadth* if we can describe its application domain with a relatively small ontological model. The documents in Figures 1–3 are all data rich and narrow in ontological breadth.

When evaluating the relevancy of HTML documents to a particular application, we want to exploit the contents rather than HTML tags and layout features. Thus, even though the designers of the HTML documents express content in various ways, we recognize the documents as relevant to the application mainly based on three patterns: *multiple-record* documents, *single-record* documents, and *application forms*, which we distinguish by the contents with respect to the application. The documents in Figures 1, 2, 3(b) are all multiple-record documents because they contain similar descriptions of several different items. Figure 4 shows a car ad linked from *Honda Accord EX* in Figure 2, which we call a single-record document because it declares the various features of only one item—the *Honda Accord EX* for sale. In addition to single- and multiple-record documents, Figure 3(a) is an application form. When considering a form, we may have, in addition to the labeled form fields, (1) selection lists with possible values of interests, and (2) the results returned, if we can automatically fill in and submit the form using default values as discussed in [23, 24].

## 2.2 Filtering Task

Real-world applications require that the recognition of document relevancy be flexible, robust, and scalable. Even if we can manually construct rules to automate this recognition, the automation depends largely on how knowledgeable a human expert is, who makes the automatic rules for the particular application domain. Especially when porting to a new application domain, existing recognition rules may no longer be appropriate. To resolve these challenges, in our approach we construct automatic recognition rules via machine learning. We reformulate the problem of recognizing relevant HTML documents for an application into a *classification* task: given a set of HTML documents consisting of semistructured HTML documents, HTML tables, and HTML forms, we attempt to assign each HTML document a concept class, which is either *relevant* or *irrelevant*, with respect to a particular application. Using car ads as an example, we want to classify the multiple-record document in Figure 1(a) as being relevant whereas we want to classify the multiple-record document in Figure 1(b) as being irrelevant.

Like a typical machine learning classification task, relevancy testing proceeds in two phases: *training* and *test*. In the training phase, we supervise the machine to train a *learner* for an

Last Updated  
Monday, January 24, 2000 12:19pm Cars for Sale

DEPENDABLE CAR  
1989 Subaru SW. Auto, AC, \$1900 OBO. Call (336)835-8579.

FACTORY WARRANTY  
1998 Elantra. Black 4 door w/ tinted windows. Auto, pb, ps, cruise, am/fm cassette stereo, a/c. Excellent condition pay off OBO. Call (336)526-5444 anytime & leave message.

1994 HONDA ACCORD EX  
Auto, power everything, jade green w/gold package. Under 100K miles. Call (336)526-1081 after 7pm.

1999 GRAND AM  
27,000 miles, silver, auto, still under warranty. \$14,000 OBO. Call (336)366-4996 anytime.

`53 Chevy Bel Aire. All original, looks like new. Serious inquiries only. \$8500. Call (336)468-8924 after 4 pm.

TWO GREAT CARS  
1973 MGB convertible. British racing green. Mags, new tires, 4-speed, 1 owner, excellent running condition. \$4500.  
1977 Olds Cutlass Supreme. New white paint job w/ 1/2 red Landau top, original mags & new tires. Auto., 1 owner, low mileage, loaded. Call (336)984-2843.

95 FORD CONTOUR  
5-speed, great condition, one owner, \$5300. Call (336)526-8853 & leave message if no answer.

SEIZED CARS FROM \$500  
Sports, luxury & economy cars, trucks, 4x4's utility and more. For current listings, call 1-800-311-5048 ext. 10012.

1996 VW JETTA GL  
26,000 miles. 4 door, 5-speed, AC, sunroof, 1 owner. \$11,000. Call (336)874-7317 anytime.

`85 Buick Park Avenue. \$500. Head may be cracked. Will run. Body good condition. Call (336)526-2768.

`95 Ford Thunderbird. Loaded, V-8, 45K, \$6995. Call S&J Motors at (336)874-3403.

`96 Mercury Tracer. 4 door, 5 speed, 34K, \$4995. Call S&J Motors at (336)874-3403.

`88 Firebird. V8, 5.0, fuel injected, T-tops, 109,000 miles, red, runs great. \$1880. Call (336)526-1164 anytime.

1990 CONVERSION VAN  
350 motor, auto, new tires, TV, VCR, captain chairs, front & rear AC. \$4,995. Call (336)320-2658 anytime.

COMMERCIAL WORK VAN  
`95 Chevy Astro, V6, w/ac & fully equipped utility shelves. \$9400. Call (336)526-2675 & leave message.

Last update: Wednesday, December 22, 1999

Select a category

Apartment For Rent For Sale or Rent Lost or Found  
For Rent Help Wanted  
For Sale House For Rent

Apartment For Rent  
ONE EFFICIENCY, 2 & 3 bdrm, all utilities paid.  
Call 281-2051 -

For Rent  
HOUSING SOLUTIONS - Free TV cable furn. \$60/wk - \$210/mo. 281-4060. -

For Sale  
1998 JD 455 mower, 60' deck. Call for price. Also, homemade Go-Cart. Call after 5:30 pm 218-281-1128. -

For Sale or Rent  
10,000 SQ. FT. office building. Handicap accessible.  
Call 281-3631. -

Help Wanted  
NOW HIRING full time and part time customer service representatives. Advancement possible and weekly pay. Must be able to work weekends and holidays. Apply at Superamerica, 411 N Main St., Crookston, MN EOE -

PART-TIME AND weekend help working with developmentally disabled adults. Call Melissa or Karen at 281-3872. -

REM-NORTHWEST Services, Inc. has a full time Program Coordinator/Coordinator position open in Crookston working with four developmentally disabled adults. Duties include hiring, staff supervision, scheduling, oversight of most areas of the home's operation. Applicant must be 18 years of age or older. Must have a high school diploma or equivalent. One year experience serving people with developmental disabilities preferred. Must have a valid driver's license and driving record that meets REM's insurability requirements. Insurance and benefits available. If interested call for application at 218-281-5113. E.O.E. -

REM-NORTHWEST Services, Inc. has full and part time Coordinator positions available in Crookston, MN, working with citizens who are developmentally disabled. Excellent benefits are offered including health, dental, life, 401K and profit sharing for full and part time employees working 20 hours a week or more. Exceptional training is provided. Applicants must be 18, have a valid driver's license and high school diploma or GED. Apply by calling for application at 218-281-5113 or 1-800-532-7655. E.O.E. -

House For Rent  
3 BDRM HOUSE \$450/mo. 281-1970. 22 STEEL BUILDINGS, NEW, must sell. 40x60x14 was \$17,500 now \$10,971; 50x100x16 was \$27,850 now \$19,990; 80x135x16 was \$79,850 now \$42,990; 100x175x20 was \$129,650 now \$78,850. 1-800-406-5126. -

Lost or Found  
FOUND: Golden retriever about 4 months old. Found 7 miles south of Crookston. Call after 5:30 pm 281-1128. -

(a) Car advertisements retrieved from <http://www.elkintribune.com/>

(b) Items for sale advertisements retrieved from <http://www.crookstontimes.com>

Figure 1: A car-ads Web document and a non-car-ads Web document

Vehicles

Search New

Pre-Owned

Specials

Get A Quote

Financing

Vehicle Pricing

Finance

Specials

Contact

Service

About

Home

**BOB HOWARD HONDA**

## Pre-Owned Inventory

To see a list of all our cars, trucks, vans and SUV's, [click here](#).

- Looking for a price quote? Check out our [Quick Quote Form](#).
- Need financing? Try our new [Pre-Approval Form](#).
- Check out our [Internet Only Specials](#).

To search for a specific vehicle or model, use our easy search engine below. Our inventory changes daily, so drop us an email or give us a call if you don't see the car you want. We will make sure you find your dream car! You searched for:

- All vehicles available.

**66 matches found.** Vehicles 1 to 25 shown.

Year	Make and Model	Price	Miles	Exterior	Photo
<input type="checkbox"/> 1999	<a href="#">Pontiac Firebird</a>	Contact Us	32,883	Blue	
<input type="checkbox"/> 2000	<a href="#">Acura RL 3.5</a>	\$23,988	36,657	Silver	
<input type="checkbox"/> 2002	<a href="#">Honda Accord EX</a>	\$21,988	13,875	White	
<input type="checkbox"/> 2002	<a href="#">Honda Passport</a>	\$20,998	10,410	Black	
<input type="checkbox"/> 2002	<a href="#">Acura RSX Type-S</a>	\$20,988	14,208	Red	
<input type="checkbox"/> 2000	<a href="#">Chevrolet Camaro</a>	\$13,995	45,297	white	
<input type="checkbox"/> 2001	<a href="#">Honda Accord Value Package</a>	\$13,995	31,710	Silver	
<input type="checkbox"/> 2001	<a href="#">Chevrolet Silverado C1500</a>	\$13,988	28,022	Pewter	

Show checked vehicles

New search

Show 25 more

All vehicles subject to prior sale. We reserve the right to make changes without notice, and are not responsible for errors.

Bob Howard Honda  
 14137 Broadway Extension  
 Oklahoma City, OK 73013

Toll Free: 1-877-944-2842  
 Phone: 405-936-8666  
 Fax: 405-936-8674  
 E-mail: [sales@bobhowardauto.dealerspace.com](mailto:sales@bobhowardauto.dealerspace.com)

Figure 2: HTML page with table from [21]



**World Wide Wheels**  
The Hottest Automotive Spot on the Net!

Year:  to

Make:

Model:

Color:

Price:  to

(a) HTML form at [22]

1996 ACURA INTEGRA LS GREEN w/ TAN int. 6 CYLINDER 90,833mi. AUTO FOR MORE INFO CALL 630-241-4846 \$7,995 Stock No. K-10318 SUPERIOR MOTOR SPORTS [\[More Detail\]](#)




---

1996 ACURA INTEGRA LS GREEN w/ TAN int. 4 CYLINDER 90,883mi. AUTO \$8,995 Stock No. K-10318 LUXURY MOTORS [\[More Detail\]](#)




---

1994 ACURA INTEGRA GREEN w/ GRAY int. 4 CYLINDER 5 SPD. .If you are Looking for a Great car to fit your Budget? Look no More! This Integra is Loaded with Power Windows, Locks, Mirrors, Moonroof, Air Cond., Alleys and Lots More! Drive it Home Today! Only at The Autobarn! \$5,980 Stock No. VE4152B THE AUTOBARN LIMITED EVANSTON [\[More Detail\]](#)



(b) Retrieved car ads after filling in the form in Figure 3(a)

Figure 3: HTML form at [22]

application. During the training phase, the learner builds a *classification model* (e.g. decision trees [20]) for the application. In the test phase, given an HTML document, the learner applies the classification model to predict whether the document is relevant.

### 3 Application Ontologies

We define an application ontology to be a conceptual-model instance that describes a real-world application in a narrow, data-rich domain of interest. Each of our application ontologies consists of two components: (1) an *object/relationship-model instance*, which 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*, which 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 5 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). The full ontology for car ads is about 600 lines long. Our obituary ontology, which is the other application ontology we discuss in this paper is about 500 lines long, but it references both a first-name lexicon and a last-name lexicon, which each contain several thousand names.

Figure 4: Linked page with additional information [21]

Figure 4: Linked page with additional information [21]



```

1. Car [-> object];
2. Car [0:0.975:1] has Year [1:~];
3. Car [0:0.925:1] has Make [1:~];
4. Car [0:0.908:1] has Model [1:~];
5. Car [0:0.45:1] has Mileage [1:~];
6. Car [0:0.8:1] has Price [1:~];
7. Car [0:2.1:~] has Feature [1:~];
8. PhoneNr [1:~] is for Car [0:1.15:~];
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";
18. ...

```

Figure 5: Car-ads application ontology (partial)

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 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 the Perl syntax. When applied to a textual document, the **extract** clause in a data frame causes a string matching a regular expression to be extracted, but only if the **context** clause also matches the string and its surrounding characters. A **substitute** clause lets us alter the extracted string before we store it in an intermediate file, in which we also store the string’s position in the document and its associated object set name. One of the nonlexical object sets is designated as the *object set of interest*—*Car* for the car-ads ontology. The notation “[-> object]” in Line 1 designates the object set of interest.

We denote a relationship set by a name that includes its object set names (e.g. *Car has Year* and *PhoneNr is for Car*). The *min: max* pairs and *min: ave: max* triples 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; *ave* designates the average number of times an object is expected to 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*, for example, specifies that a car

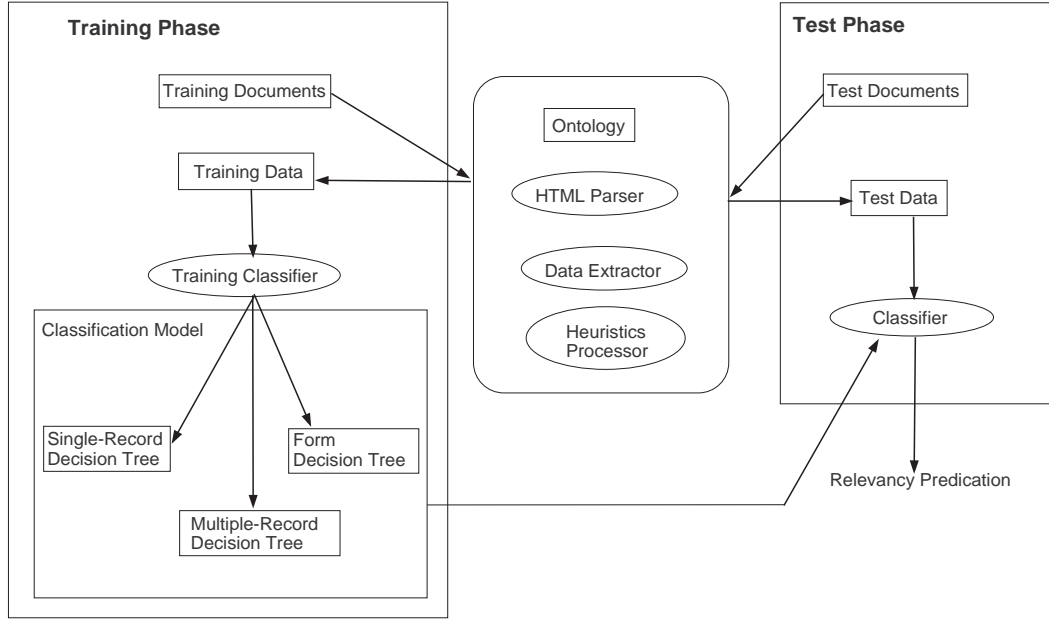


Figure 6: High-level architecture for recognizing relevant HTML documents for an application

need not have any listed features, that a car has 2.1 features on the average, and that there is no specified maximum for the number of features listed for a car.

For our car-ads and obituaries application ontologies, we obtained participation constraints as follows. To make our constraints broadly representative, we selected ten different regions covering the United States and found one car-ads page and one obituary page from each of these regions. From each of these pages we selected twelve individual car-ads/obituaries by taking every  $n/12$ -th car-ad/obituary, where  $n$  was the total number of car-ads/obituaries on the page. We then simply counted by hand and obtained minimum, average, and maximum values for each object set in each relationship set and normalized the values for a single car ad or obituary.

## 4 Architecture for Web Document Filtering

In Figure 6, we present a high-level architecture of our approach for checking relevancy of HTML documents for an application. In the architecture, one application **Ontology**, which specifies object and relationship sets and data frames for an application, is predefined independently of HTML documents. Given an HTML document, we use an **HTML Parser** to parse the document. For each document, we collect two kinds of text components. One kind is the text that appears in the whole document, which we call the *document text component*. The other kind are text fragments that appear within individual forms in the document, each of which we call a *form text component*. A form text component includes the text that labels form fields and values in selection lists. If the document does not contain any form, the set of form text components is

empty. Note that the document text component of an HTML document subsumes all the form text components that appear within the forms as well as the text that is outside the forms in the document. The **Data Extractor** applies the application ontology using ontology-based data-extraction techniques [12] to retrieve data from all the document and form text components in the document.

Based on the data extracted from a text component, which is either a document text component or a form text component, we construct a list of heuristics to evaluate the relevancy of the text component to the application ontology by a **Heuristics Processor**. Each individual heuristic processor evaluates the relevancy of a document to the application ontology. We normalize a measure for each individual heuristic as a confidence measure in the range from 0 to 1. The higher the confidence value, the more confidently we consider the text component appropriate for the application ontology for the particular heuristic. As will become evident, in our approach, we construct heuristics that are tightly dependent on the specification of application ontologies in a flexible and robust way.

We formalize the evaluation of HTML-document relevancy to an application as follows. An application ontology  $O$  specifies the application. HTML documents are structured objects: a document consists of a document text component and a set of form text components. More precisely, a *document*  $d$  is a sequence of text components, written  $d = [t_d, t_{f_1}, \dots, t_{f_n}]$ , where  $n$  is the number of forms in the document  $d$ . Note that  $d = [t_d]$  if the document  $d$  does not contain any form. Given the application ontology  $O$  and an HTML document  $d = [t_d, t_{f_1}, \dots, t_{f_n}]$ , we use  $m$  heuristic rules to compute  $m$  confidence measures  $H_{t_d} = (h_1, h_2, \dots, h_m)$  for the document text component  $t_d$  and use the same  $m$  heuristic rules to compute  $H_{t_{f_i}} = (h_{i1}, h_{i2}, \dots, h_{im})$  for each form text component  $t_{f_i}$  ( $1 \leq i \leq n$ ). Thus we describe the similarity between the HTML document  $d$  and the application ontology  $O$  as a heuristic vector  $d_H = \langle H_{t_d}, H_{t_{f_1}}, \dots, H_{t_{f_n}} \rangle$  over  $n+1$   $m$  tuples of confidence measures. Since we reformulate the recognition of document relevancy into a classification problem, we attempt to assign  $d_H$  to either a concept class  $c_P$ , which represents positive (relevant to the application), or  $c_N$ , which represents negative (irrelevant to the application).

#### 4.1 Training Phase

In the training phase, we train a learner using a **Training Classifier**, which for our work is the popular machine learning algorithm C4.5 [20]. C4.5 is a rule post-pruning decision-tree algorithm. The learning task is to check the suitability of documents for a given application ontology (i.e. to do binary classification by returning “Y” (yes) when a document is suitable and returning “N” (no) otherwise). The bias of C4.5 favors the shortest rule, so that if several rules are equally accurate, a decision tree with the fewest branches is chosen. Actually, the training classifier could use other

learning algorithms. Indeed, our research group has tried both multivariate statistical analysis [25] and logistic regression [26] as alternative learning algorithms (although only for multiple-record documents).

Considering all three patterns (multiple-record documents, single-record documents, and application forms), we divide the learning task into three subtasks: (1) suitability of a document text component that describes multiple records for one application ontology, (2) suitability of a document text component that represents an individual singleton record for one application ontology, and (3) suitability of a form that yields information for one application ontology. C4.5 learns a decision tree for each of the three subtasks.

We use supervised learning in our approach to train the learner. For each application, a human expert selects a set of HTML documents for the application ontology as **Training Documents**. The expert selects training documents for each subtask considering the different kinds of HTML documents in the real world. For example, for the car-ads application, we selected semistructured HTML documents as well as HTML tables containing multiple car ads as training documents for the subtask to train the learner so that the learner can obtain the knowledge it needs to classify a multiple-record car-ads document. Various design patterns of HTML tables, including the page in Figure 2 for example, can be considered as training documents.

The human expert provides the learner with **Training Data** as follows. For each training document  $d$ , the expert creates a *training example* either for the document text component in  $d$  or for one of the form text components in  $d$ , if any.<sup>1</sup> A training example,  $e = (H_x, c_y)$ , is a list of values  $H_x$ , one for each heuristic rule, plus a concept class  $c_y$ , which is either  $c_P$  for a positive training example or  $c_N$  for a negative training example. The training data contains three groups of training examples, each of which is for one of the three subtasks. For a training document  $d$  in the set of training documents, if it contains a form  $f_i$  relevant to  $O$ , the expert uses the list of heuristic values  $H_{t_{f_i}}$ , which is obtained from the form text component  $t_{f_i}$ , to construct a positive training example  $(H_{t_{f_i}}, c_P)$  for the subtask specifying the relevancy of a form to the application ontology  $O$ . Otherwise if the form  $f_i$  is not relevant to  $O$ , the experts builds a negative training example  $(H_{t_{f_i}}, c_N)$  for the subtask. If the document is a single-record document relevant to  $O$ , the expert uses the list of heuristic values  $H_{t_d}$ , which is obtained based on the document text component  $t_d$ , to build a positive training example  $(H_{t_d}, c_P)$  for the subtask specifying the relevancy of a single-record document relevant to  $O$ , or vice versa a negative training example. Similarly, the expert uses  $H_{t_d}$  obtained from the document text component of an HTML document to build a training example for the subtask specifying the relevancy of a multiple-record document with respect to  $O$ .

---

<sup>1</sup>Typically, an informational HTML document contains its primary information either directly on the page or behind one of its forms. The human expert should train the learner by selecting the component that appears to contain the primary information for the document.

```

Input: an HTML document  $d$ , an application ontology  $O$ ,
      and a classification model  $M_O$  ( $\tau_1, \tau_2, \tau_3$ ).
Output: prediction
  Build  $d_H = \langle H_{t_d}, H_{t_{f_1}}, H_{t_{f_2}}, \dots, H_{t_{f_n}} \rangle$  for  $d$ 
  //check the document text component in  $d$ 
  call EvaluateDocText( $d, H_{t_d}, M_O$ )
  //check the form text components in  $d$ 
  for each  $H_{t_{f_i}}$  in  $d_H$  where  $1 \leq i \leq n$ 
    call EvaluateFormText( $d, H_{t_{f_i}}, M_O$ )
  Output the prediction for  $d$ 

```

Figure 7: Test algorithm to recognize relevant HTML documents

The C4.5 algorithm knows how the heuristic values in the training examples should be optionally combined to best match application ontologies with text components that appear in documents. Thus the learner builds a **Classification Model**, which we denote as  $M_O$ , as the output of the training phase. The classification model contains three decision trees. One tree  $\tau_1$  is a set of rules to decide if a document is a single-record document relevant to an application ontology, which we call a *single-record tree*. The second decision tree  $\tau_2$  is a set of rules to decide if a document is a multiple-record document relevant to an application ontology, which we call a *multiple-record tree*. The last decision tree  $\tau_3$  is a set of rules to decide if an HTML form is relevant to an application ontology, which we call an *application-form tree*.

## 4.2 Test Phase

In the test phase, we use a set of HTML documents, which we call **Test Documents**, to evaluate the performance of the learner trained in the training phase for the application ontology  $O$ . Given the classification model built in the training phase, we use the algorithm in Figure 7, which is the **Classifier** in Figure 6, to test the relevancy of an HTML document. The input to the test algorithm is the classification model and an HTML document  $d$  from the set of test documents. The output is a prediction about the relevancy of  $d$  to  $O$ . The classification model has three decision trees at its disposal and classifies a document with a positive prediction if the learner classifies the document as positive based on any one of the three decision trees.<sup>2</sup> Figure 8 shows two subprocedures of the test algorithm that check the relevancy based on the evaluation over either a document text component or a form text component. In Figure 6, the **Test Data** consists of the heuristic vectors computed in the algorithm for the test documents, and the **Relevancy Predictions** are the predictions output from the test algorithm for the test documents.

---

<sup>2</sup>C4.5 trains the learner to obtain the classification model by applying three sets of training examples independently in the training phase. Thus, in the test phase, it is possible that the learner classifies a test document as both a single-record document and a multiple-record document relevant to the application ontology based on the document text component in the document. Moreover, it also could predict that one relevant document contains both relevant forms as well as a singleton record or multiple records for the application. In our approach, however, in the test phase, we are only interested in a prediction. Thus, we declare a document to be relevant if any one of the three trees in the classification model returns  $c_P$ , a positive result.

```

sub EvaluateDocText( $d, H_{t_d}, M_O$ )
  //check a document text component
  Evaluate  $H_{t_d}$  using  $\tau_1$  and obtain  $c_1$ 
  Evaluate  $H_{t_d}$  using  $\tau_2$  and obtain  $c_2$ 
  if either  $c_1$  or  $c_2$  equals  $c_P$ 
    Output  $c_P$  for the document text component of  $d$ 
  else
    Output  $c_N$  for the document text component of  $d$ 

sub EvaluateFormText( $d, H_{t_{f_i}}, M_O$ )
  //check a form text component
  Evaluate  $H_{t_{f_i}}$  using  $\tau_3$  in  $M_O$  and obtain  $c$ 
  Output  $c$  for the form text component of  $f_i$ 

```

Figure 8: Subprocedures of prediction algorithm

Input: an HTML document  $d$ , an application ontology  $O$ ,  
a sample size  $N$ , and a classification model  $M_O$  ( $\tau_1, \tau_2, \tau_3$ ).

Output: prediction

```

call LocateUsefulLinks( $d, M_O, N$ ) and output  $L$ 
Randomly select subsequent links  $L_S$  from  $L$ 
for each linked page  $d'$  in  $L_S$ 
  Compute  $H_{t_{d'}}$  over the document text component of  $d'$ 
  call EvaluateDocText( $d', H_{t_{d'}}, M_O$ ) and output  $c'$ 
  if  $c'$  equals  $c_P$ 
    Retrieve text from  $d'$ 
    Insert the text in  $d$ 
Compute  $H_{t_d}$  over the document text component  $t_d$  of  $d$ 
if  $t_d$  in  $d$  is modified
  Evaluate  $H_{t_d}$  using  $\tau_2$  and output  $c$ 
else
  call EvaluateDocText( $d, H_{t_d}, M_O$ ) and output  $c$ 
Output  $c$  for the document text component of  $d$ 

```

Figure 9: Evaluation of the document text component of an HTML document by applying relevant linked pages

```

sub LocateUsefulLinks( $d, M_O, N$ )
  Collect all the links  $L$  in  $d$ 
  Group  $L$  into groups by link-address prefixes
  Sort the groups in descending order by sizes
  for each group  $G$  in the groups
    Sample and evaluate  $N$  links in  $G$  based on  $M_O$ 
    If the group of linked document is relevant
      Output  $G$  and break the loop
  if no group is found
    Output an empty set

```

Figure 10: Subprocedure to locate useful links in an HTML document

In the algorithm of Figure 7, the learner classifies an HTML document by exploiting the two kinds of text components (regular text and form text) that appear in the document based on the classification model  $M_O$ . In addition to the text components within the document, the learner can check available linked pages and pages returned from form filling to further evaluate the document. The HTML table in Figure 2 contains several links, some of which lead to more detailed descriptions of the car ads in the document. Figure 4 is one of the linked pages from the top page in Figure 2. If we can determine that a linked page (e.g. the document in Figure 4) is relevant to the application, we can use this information to help classify the top page (e.g. the document in Figure 2). Intuitively, if a top page leads to multiple relevant linked pages, we have more confidence that the top page contains multiple records that are of interest. Figure 9 shows the algorithm to evaluate the document text component of a multiple-record HTML document by exploiting relevant documents in linked pages. Because of the expense of retrieving potentially many dozens of linked pages, the algorithm does not explore all linked pages from the top-page HTML document  $d$ . Instead, we first call the subprocedure in Figure 10 to locate a group of potentially useful links from  $d$ . Since we believe that the useful links in a multiple-record document are likely to all be together in a common repository, the procedure to locate the useful links first groups links in  $d$  by (longest) common URL prefix and then sorts the groups of links in descending order based on the number of links in each group since the number of the links that are of interest usually is the largest in a relevant multiple-record document. To both discard spurious groups of links with only one or two members and to avoid processing all the links in a group, we choose a small threshold  $N$  (we chose  $N = 5$  for our algorithms). Then, in the loop of the algorithm in Figure 10, if the number of the links in a group is less than  $N$ , we ignore the evaluation of the group, and if the number of links is greater than  $N$ , we only evaluate  $N$  of them. We evaluate the links in a group by checking the relevancy of the top-level document  $d$  with the text of the linked pages inserted into  $d$ .

As already mentioned, we also use information on pages returned by automatic form filling [23, 27].<sup>3</sup> Figure 11 shows the algorithm to further evaluate an HTML form by considering a

---

<sup>3</sup>We point out that automatic form filling does not always yield results as explained in [23, 27]. Thus, we can

Input: an HTML form  $f_i$ , an application ontology  $O$ ,  
 and a classification model  $M_O$  ( $\tau_1, \tau_2, \tau_3$ ).  
 Output: prediction  
 Retrieve a document  $d'$  by filling in and submitting  $f_i$   
 Compute  $H_{t_{d'}}$  over the document text component of  $d'$   
 call  $\text{EvaluateDocText}(d', H_{t_{d'}}, M_O)$  and output  $c$   
 Output  $c$  for the form  $f_i$

Figure 11: Evaluation of an HTML form using a retrieved document obtained by form filling

document retrieved by form filling. We can exploit the retrieved documents using two strategies based on the preference of system users: if system users prefer a better recall ratio, we evaluate a test document  $d$  using the algorithm in Figure 11 if the learner classifies  $d$  as irrelevant to the application ontology based on the form alone; otherwise, if system users prefer a better precision ratio, we evaluate a test document  $d$  using the algorithm in Figure 11 if the learner classifies  $d$  as relevant to the application based on the form alone.

## 5 Recognition Heuristics

In the high-level architecture of our approach in Figure 6, the heuristics processor computes heuristic measures over document and form text components that appear in a document. In our approach, we consider three kinds of heuristics: *density* heuristics, an *expected values* heuristic, and a *grouping* heuristic. Given an application ontology  $O$ , the set of density heuristics measure the densities of constants and keywords defined in the application ontology  $O$  that appear in a text component  $t_d$ , which is either a document text component or a form text component. The expected-values heuristic uses the Vector Space Model (VSM) [19], a common information-retrieval measure of document relevance, to compare the number of constants expected for each object set, as declared in  $O$ , to the number of constants found in  $t_d$  for each object set. The grouping heuristic measures the occurrence of groups of lexical values found in  $t_d$  with respect to expected groupings of lexical values implicitly specified in  $O$ .

The next three subsections define these heuristics, explain the details about how we provide a measure for each heuristic, and give examples to show how they work. When reading these subsections, bear in mind that in creating these heuristics, we favored simplicity. More sophisticated measures can be obtained. For example, for density measures we could account for uncertainty in constant and keyword matches [28]. For expected values, we could more accurately match object sets with recognized values by using more sophisticated downstream heuristics [12, 29]. For grouping, we could first compute record boundaries [30] and rearrange record values [29]. However, more sophisticated measures are more costly. We have chosen to experiment with less costly heuristics, and, as will be shown, our results bear out the seeming correctness of this choice.

---

only apply this technique when automatic form filling does yield results.



## 5.1 Density Heuristics

A text component  $t_d$  parsed from an HTML document  $d$  that is relevant to a particular application ontology  $O$  should include many constants and keywords for object sets defined in the ontology. Based on this observation, we define a set of density heuristics. We compute the density heuristics with respect to an application ontology and each object set that has keywords or constants specified in the ontology. In both counts we exclude the characters in HTML tags. We compute the density of  $t_d$  with respect to an application ontology  $O$  as follows:

$$\text{Density}(t_d, O) = \text{total number of matched characters} / \text{total number of characters}$$

where *total number of matched characters* is the number of characters of the constants and keywords recognized by  $O$  in  $t_d$ , and *total number of characters* is the total number of characters in  $t_d$ . Further, we compute the density of  $t_d$  with respect to an object set  $o$  in  $O$  as follows:

$$\text{Density}(t_d, o) = \text{total number of matched characters for } o / \text{total number of characters}$$

where *total number of matched characters for  $o$*  is the number of characters of the constants and keywords recognized by regular expressions specified for  $o$  in  $t_d$ , and *total number of characters* is the same as in the computation of  $\text{Density}(t_d, O)$ .

We must be careful, of course, not to count characters more than once. For example, in the phrase “asking only 18K,” a car-ads application ontology might recognize “18K” as potentially both a price and a mileage. Nevertheless, we should only count the number of characters as three, not six. Further, we need determine whether we count the value “18K” for a price or for a mileage.

Consider the document text component  $t_{d_a}$  in the multiple-record document  $d_a$  in Figure 1(a). Recall that the nonlexical object set of the car-ads application ontology is *Car*, and the lexical object sets are *Year*, *Make*, *Model*, *Mileage*, *Price*, *Feature*, and *PhoneNr*. Some of the lexical values found in  $t_{d_a}$  include “1989” (*Year*), “\$1900” (*Price*), “100K” (*Mileage*), “Auto” (*Feature*), “Cruise” (*Feature*), “(336)835-8579” (*PhoneNr*), “Subaru” (*Make*), and “SW” (*Model*). The keywords “Cars for Sale” for the object set of interest *Car*, “miles” and “mileage” for *Mileage*, and “Call” for *PhoneNr* appear in  $d_a$ . The total Number of characters in  $t_{d_a}$  is 1992, whereas the number of matched characters is 696. Hence, the  $\text{Density}(t_{d_a}, O)$  is  $0.3493 = 696/1992$ . For each object set in the car-ads application ontology  $O$ , there is also a density measure. For example, the number of matched characters for *Make* is 47. Therefore,  $\text{Density}(t_{d_a}, \text{Make})$  is  $0.0236 = 47/1992$ .

When we apply the density heuristics for the car-ads application ontology to the document text component  $t_{d_b}$  of the document  $d_b$  in Figure 1(b), the densities are much lower. Although no makes, models, or car features appear, there are years, prices, and phone numbers and the ontology (mistakenly) recognizes “10,000” (in “10,000 SQ. FT.”) and “401K” (the retirement

plan) as potential mileages. Altogether, 229 characters of 2627 are recognized by the car-ads ontology. Thus the density to the car-ads application ontology  $Density(t_{d_b}, O)$  is 0.0871. There are also eight other densities, one for each object set. For example, the document text component of  $d_b$  contains keywords and values for *PhoneNr*, and the density for *PhoneNr* is 0.0533. The density for *Car* is 0.0 since the document text component does not contain any keywords for the object set of interest, *Car*, in the car-ads application ontology.

## 5.2 Expected-Values Heuristic

We apply the VSM model to measure whether a text  $t_d$  parsed from an HTML document  $d$  has the number of values expected for each lexical object set of an application ontology  $O$ . Based on the lexical object sets and the participation constraints in  $O$ , we construct an ontology vector  $V_O$ . Based on the same lexical object sets and the number of constants recognized for these object sets by  $O$  in  $t_d$ , we construct a document vector  $V_{t_d}$ . We measure the relevance of  $t_d$  to  $O$  with respect to our expected-values heuristic by observing the cosine of the angle between  $V_{t_d}$  and  $V_O$ .

To construct the ontology vector  $V_O$ , we (1) identify the lexical object-set names (these become the names of the coefficients of  $V_O$ ) and (2) determine the average participation (i.e. the expected frequency of occurrence) for each lexical object set with respect to the object set of interest specified in  $O$  (these become the values of the coefficients of  $V_O$ ). For example, the ontology vector for the car-ads application ontology is  $\langle Year:0.975, Make:0.925, Model:0.908, Mileage:0.45, Price:0.8, Feature:2.1, PhoneNr:1.15 \rangle$ , where these values are the average participation-constraint values obtained as explained in Section 3. Thus, for a typical single car ad we would expect almost always to find a year, make, and model, but we only expect to find the mileage about 45% of the time, the price about 80% of the time. Further, we expect to see a list of features that on the average have a couple of items in it, and we expect to see a phone number and sometimes more than one phone number.<sup>4</sup>

The names of the coefficients of  $V_{t_d}$  are the same as the names of the coefficients of  $V_O$ . We obtain the value of each coefficient of  $V_{t_d}$  by automatically counting the number of appearances of constant values in  $t_d$  that belong to each lexical object set. Table 1 shows the values of the coefficients of the document vector for the document text component of the car-ads document in Figure 1(a), and Table 2 shows the values of the coefficients of the document vector for the document text component of the non-car-ads document in Figure 1(b).

We have discussed the creation of a document vector as though correctly detecting and classifying the lexical values in a text in a document were easy—but it is not easy. We identify potential lexical values for an object set as explained in Section 3; this can be error-prone, but we can adjust the regular expressions to improve this initial identification and achieve good results

---

<sup>4</sup>It is easy to see that the variance might be useful, as well, but we found that the expected numbers were sufficient to get good results for the examples we tried.

Name of Lexical Object Set	Corresponding Lexical Values Found in the Document	Number of Lexical Values
Year	1989, 1998, 1994, 1999, '53, 1973, 1977, 95, 1996, ...	16
Make	Subaru, HONDA, Chevy, Olds, FORD, VW, Buick, Mercury, ...	10
Model	SW, Elantra, ACCORD, GRAND AM, Cutlass, CONTOUR, JETTA, ...	12
Mileage	100K, 27000, 26000, 45K, 34K, 109000	6
Price	\$1900, \$14,000, \$8500, \$4500, \$5300, \$11,000, \$6995, \$4995, \$1880, ...	11
Feature	Auto, Black, 4 door, pb, ps, cruise, am/fm, cassette, stereo, green, ...	29
PhoneNr	(336)835-8579, (336)526-5444, (336)526-1081, (336)366-4996, ...	15

Table 1: Lexical values found in the multiple-record car advertisements in Figure 1(a)

Name of Lexical Object Set	Corresponding Lexical Values Found in the Document	Number of Lexical Values
Year	1999, 1998, 60, 401K, 50, 80	6
Make		0
Model		0
Mileage	10,000, 401K	2
Price	\$17,500, \$10,971, \$27,850, \$19,990, \$79,850, \$42,990, \$129,650, \$78,850	8
Feature		0
PhoneNr	281-2051, 281-4060, 218-281-1128, 281-3631, 281-3872, 218-281-5113, 218-281-5113, 800-532-7655, 281-1970, 800-406-5126, 281-1128	11

Table 2: Lexical values found in the multiple-record *Items for Sale* document in Figure 1(b)

[12]. After initial identification, we must decide which of these potential object-set/constant pairs to accept. In our downstream processes, we use sophisticated heuristics based on keyword proximity, application-ontology cardinalities, record boundaries, and missing-value defaults to best match object sets with potential constants. For upstream ontology/document matching we use techniques that are far less sophisticated and thus also far less costly. In our simple upstream procedures we consider only three cases: (1) when multiple specializations of an object set are present, a recognized string appears without required accompanying keywords, (2) a recognized string has no overlap either partially or completely with any other recognized string, and (3) a recognized string does overlap in some way with at least one other recognized string. If one object set  $o$  is a specialization of another object set described in an application ontology, the keywords specified for  $o$  in the application ontology are mandatory. For example, there are several dates in an obituaries application: *Death Date*, *Birth Date*, *Funeral Date*, *View Date*, and *Interment Date*. All dates are specializations of *Date* in the obituary application ontology. To distinguish the multiple kinds of dates in HTML documents, it is necessary to use keywords to sort out the difference. For Case 1, we reject a recognized string if keywords are not present. For Case 2, we accept the recognized string for an object set even if the sophisticated downstream processes would reject it. For Case 3, we resolve the overlap simplistically, as follows. There are three subcases: (1) exact match, (2) subsumption, and (3) partial overlap. (1) If a lexical value  $v$  is recognized as potentially belonging to more than one lexical object set, we use the closest keyword that appears before or after  $v$  to determine which object set to choose; if no applicable keyword is found, we choose one of the object sets arbitrarily. (2) If a lexical value  $v$  is a proper substring of lexical value  $w$ , we retain  $w$  and discard  $v$ . (3) If lexical value  $v$  and lexical value  $w$  appear in a Web document, such that a suffix of  $v$  is a prefix of  $w$ , we retain  $v$  and discard  $w$ .

As mentioned, we measure the similarity between an ontology vector  $V_O$  and a document vector  $V_{t_d}$  by measuring the cosine of the angle between them. In particular, we use the *Similarity Cosine Function* defined in [19], which calculates the acute angle  $\text{Similarity}(t_d, O) = \cos \theta = P/N$ , where  $P$  is the inner product of the two vectors, and  $N$  is the product of the lengths of the two vectors. When the distribution of values among the object sets in  $V_{t_d}$  closely matches the expected distribution specified in  $V_O$ , the angle  $\theta$  will be close to zero, and  $\cos \theta$  will be close to one.

Consider the car-ads application ontology  $O$  in Figure 5 and the Web document  $d_a$  in Figure 1(a). The coefficients of  $V_O$  for  $O$  are 0.975, 0.925, 0.908, 0.45, 0.8, 2.1, and 1.15, which are the expected frequency values of lexical object sets *Year*, *Make*, *Model*, *Mileage*, *Price*, *Feature*, and *PhoneNr*, respectively for a single ad in the car-ads application ontology. The coefficients of  $V_{t_{d_a}}$  for  $d_a$  are 16, 10, 12, 6, 11, 29, and 15 (see the last column of Table 1), which are the actual number of appearances of the lexical values in  $d_a$ . We thus compute  $\text{Similarity}(t_{d_a}, O)$  to

be 0.9956. Now consider the car-ads application ontology  $O$  again and the Web document  $d_b$  in Figure 1(b). The coefficients of  $V_O$  are the same, but the coefficients of  $V_{t_{d_b}}$  for  $d_b$  are 6, 0, 0, 2, 8, 0, and 11 (see the last column of Table 2). We thus compute  $\text{Similarity}(t_{d_b}, O)$  to be 0.5669.

### 5.3 Grouping Heuristic

A text  $t_d$  of an HTML document  $d$  likely applies to an application ontology if the values in the text form groups that can be recognized as records for  $O$ . As a simple heuristic to determine whether the recognized values are interleaved in a way that could be considered consistent with potential records of  $O$ , we consider the group of values in a document that should appear at most once in each record and measure how well they are grouped.

We refer to an object set whose values should appear at most once in a record as a *1-max lexical object set*. Maximum participation constraints in an ontology constrain the values of the 1-max object sets to appear at most once in a record. For example, in the car-ads application ontology, the 1-maximum on *Car* in the relationship set *Car*  $[0:0.975:1]$  has *Year*  $[1:*]$  specifies that *Year* is a 1-max object set. Other 1-max lexical objects in the car-ads ontology are *Make*, *Model*, *Mileage*, and *Price*.

Instead of counting the number of 1-max lexical objects in an application ontology  $O$ , a more accurate counting approach is to sum the average values expected for the 1-max objects in  $O$ . Since the average values expected for *Year*, *Make*, *Model*, *Mileage*, and *Price* in the car-ads ontology are 0.975, 0.925, 0.908, 0.45, and 0.8, respectively, the anticipated number of lexical values from these object sets in a car advertisement is 4.058. To obtain an expected group size, we truncate the decimal value of the sum.

The expected group size  $n$  is an estimate of the number of 1-max object-set values we should encounter in a document within a single record. On the average, each record should have  $n$  1-max object sets. Thus, if we list all recognized 1-max object-set values in the order they occur in a document  $d$  and divide this sequence into groups of  $n$ , each group should have  $n$  values from  $n$  different object sets. The closer a document comes to this expectation, the better the grouping measure should be. For the multiple-record car-ads Web document in Figure 1(a), Figure 12(a) shows the first four groups of 1-max lexical object-set values extracted from the document. Similarly, Figure 12(b) shows the first four groups of 1-max lexical object-set values extracted from the document in Figure 1(b).

We measure how well the groups match the expectations with a grouping factor (denoted *Grouping*), which is calculated as follows:

$$\text{Grouping}(t_d, O) = \frac{\text{Sum of Distinct Lexical Values in Each Group}}{\text{Number of Groups} \times \text{Expected Number of Values in a Group}}$$

For example, the number of extracted groups from the document text component  $t_{d_a}$  of  $d_a$  in Figure 1(a) is 13 (1 group of 2, 5 groups of 3, and 7 groups of 4). Since the number of anticipated

Year: 2000	Year: 1999
Year: 1989	Year: 1998
Make: Subaru	Year: 1960
Model: SW	Mileage: 10000
-- Nr of Distinct "One Max" Object Sets: 3	-- Nr of Distinct "One Max" Object Sets: 2
Price: 1900	Mileage: 401000
Year: 1998	Year: 1940
Model: Elantra	Price: 17500
Year: 1994	Price: 10971
-- Nr of Distinct "One Max" Object Sets: 3	-- Nr of Distinct "One Max" Object Sets: 3
Make: HONDA	Year: 1950
Model: ACCORD	Price: 27850
Mileage: 100000	Price: 19990
Year: 1999	Year: 1980
-- Nr of Distinct "One Max" Object Sets: 4	-- Nr of Distinct "One Max" Object Sets: 2
Model: GRAND AM	Price: 79850
Mileage: 27000	Price: 42990
Price: 14000	Price: 129650
Year: 1953	Price: 78850
-- Nr of Distinct "One Max" Object Sets: 4	-- Nr of Distinct "One Max" Object Sets: 1
(a) First four groups of 1-max lexical values extracted from Figure 1(a)	(b) First four groups of 1-max lexical values extracted from Figure 1(b)

Figure 12: Groups of 1-max lexical values extracted from HTML documents

lexical values in each group is four, *Grouping* of  $t_{d_a}$  is

$$Grouping(t_{d_a}, O) = \frac{(2 \times 1) + (3 \times 5) + (4 \times 7)}{(1 + 5 + 7) \times 4} = 0.8653$$

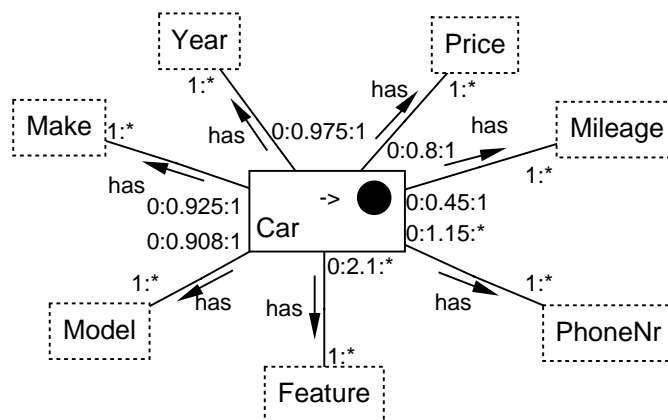
By way of comparison, the number of extracted groups from the document text component  $t_{d_b}$  of HTML document  $d_b$  in Figure 1(b) is 4 (1 group of 1, 2 groups of 2, and 1 group of 3). Since the number of anticipated lexical values in each group is four, the *Grouping* factor for  $t_{d_b}$  is 0.5.

## 6 Empirical Evaluation

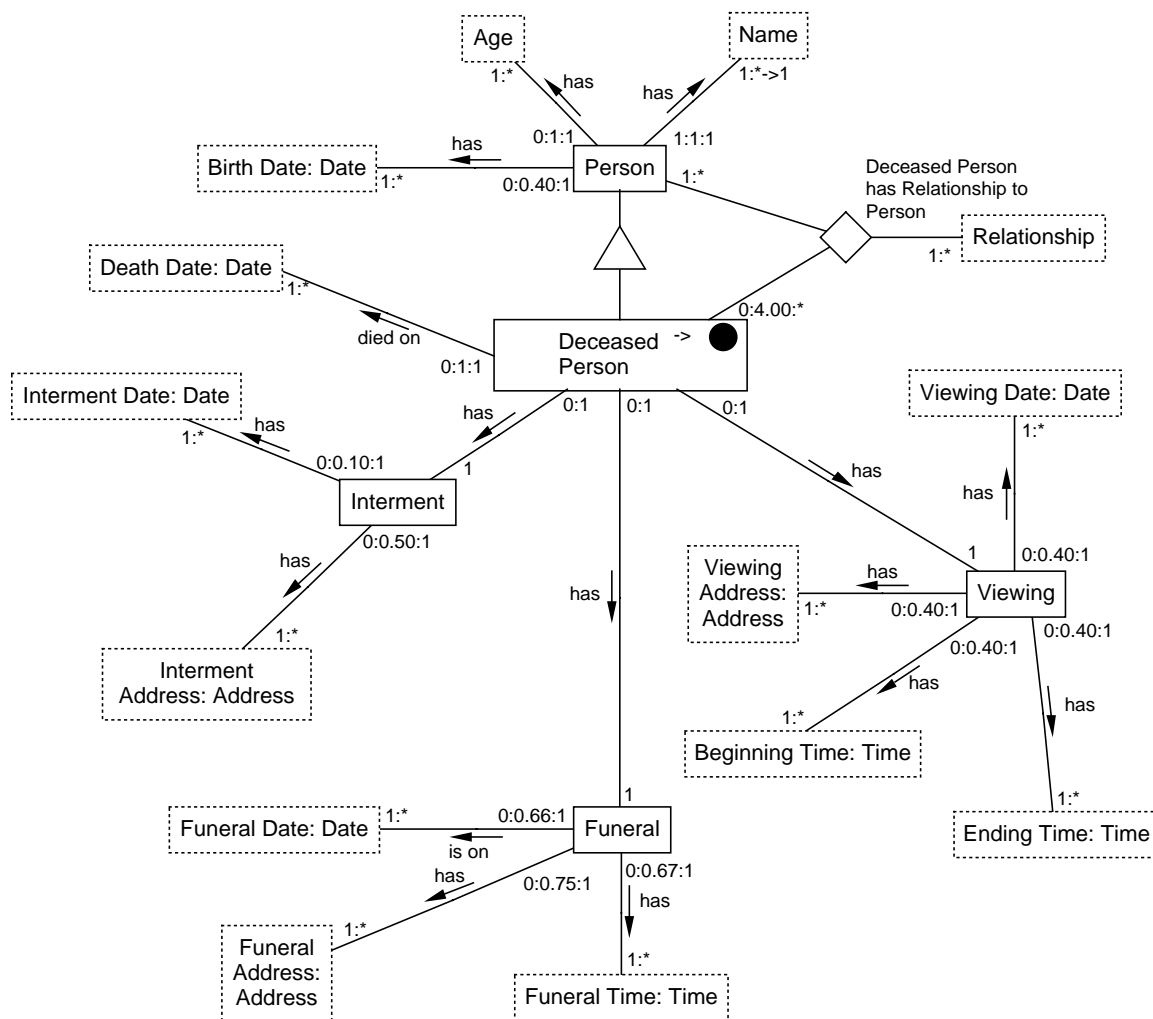
We evaluated our approach on two real-world applications: *car ads* and *obituaries*. Our goals were to evaluate system performance over multiple kinds of HTML documents for real-world applications.

### 6.1 Applications and HTML documents

Figure 13 shows the graphical versions of the two application ontologies. The car-ads application in Figure 13(a) is representative of many simple applications, whereas the obituaries application in Figure 13(b) is representative of more complex applications. Not only does the obituaries



(a) Car-ads application ontology



(b) Obituary application ontology

Figure 13: Graphical versions of application ontologies

<i>Application</i>	<i>Multiple-Record</i> <i>(semistructured/table/negative)</i>	<i>Single-Record</i> <i>positive/negative</i>	<i>Form</i> <i>positive/negative</i>
Car Ads	31/112/363	614/636	50/69
Obituaries	68/0/135	62/135	30/32

Table 3: Training data for car ads and obituaries

application have more object sets and relationship sets, but these object and relationship sets are also more complex. One relationship set is ternary, more than one object set is nonlexical, and several relationship sets are specializations—for example, both *Birth Date* and *Death Date* are specializations of *Date* as denoted by “: *Date*” following the name of these object sets.

For the car-ads application, we collected three sets of HTML documents: semi-structured HTML documents, HTML tables, and HTML forms. For the obituaries application, we collected only semistructured and HTML form documents. (Obituaries rarely, if ever, appear as tables having attributes such as *Deceased Name*, *Age*, *Death Date*, etc.) We divided the documents for each application into two sets: training documents and test documents. Table 3 shows the characteristics of the training data for car ads and obituaries obtained from the training documents. For each application, there were three sets of training examples, one each for single-record documents, multiple-record documents, and application-forms.

Table 4 shows the distributions of semistructured HTML documents, HTML tables, and HTML forms in the test documents. For the car-ads application, 10 of the semistructured HTML documents contained multiple-record car ads, and 10 of them contained single-record car ads. All the HTML tables contained multiple-record car ads. For the obituaries application, the semistructured HTML documents contained 20 multiple-record documents and 10 single-record documents. Among the 20 multiple-record obituary documents, 10 documents contained only partial obituaries. These 10 documents led to linked pages, some of which contained complete obituary records. The 10 HTML form documents for each application contained application forms but no single- or multiple-records of interest. The negative documents collected for each application contained documents with applications similar to those of the application ontologies. For example, we included forms to retrieve used auto parts, car reviews, and motorcycle sales to test the learner trained for car ads, and we included birth and marriage announcements, genealogy records and forms, and bibliographies to test the learner trained for obituaries. We assumed that if our filtering methodology could sort out differences among closely related applications, it could easily reject vastly different application documents (e.g. country descriptions, rail and flight schedules, university home pages, and molecular biology data). Indeed, the system made no mistakes rejecting obituaries and obituary like applications for car ads and rejecting car ads and car-ad like applications for obituaries.



<i>Application</i>	<i>Semistructured</i>	<i>HTML Table</i>	<i>HTML Form</i>	<i>Negative</i>
Car Ads	20	10	10	40
Obituaries	30	0	10	40

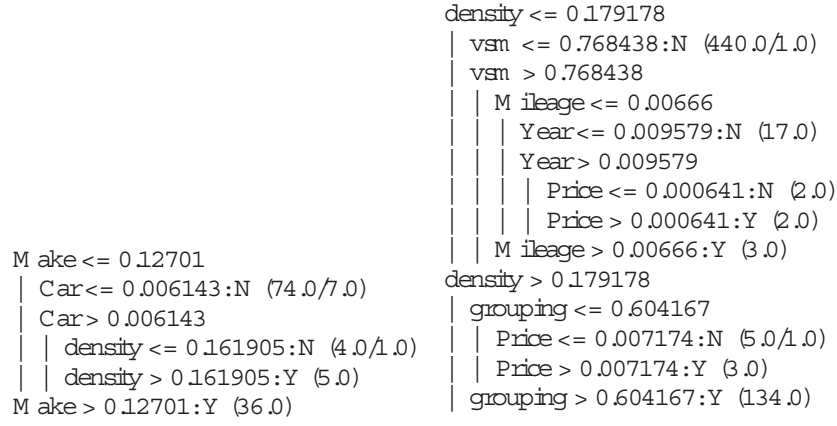
Table 4: Test documents for car ads and obituaries

Even though some of the semistructured HTML documents and HTML table documents in Table 4 contained additional irrelevant application forms (e.g. user registration forms), we expected that the learners would produce appropriate predictions based on the document text components that appear in the documents rather than the irrelevant form text components. For the HTML form documents, since they did not contain application records, we expected that the learners would produce the positive predictions using the application-form decision trees based on form text components that appear within forms.

## 6.2 Classification Models

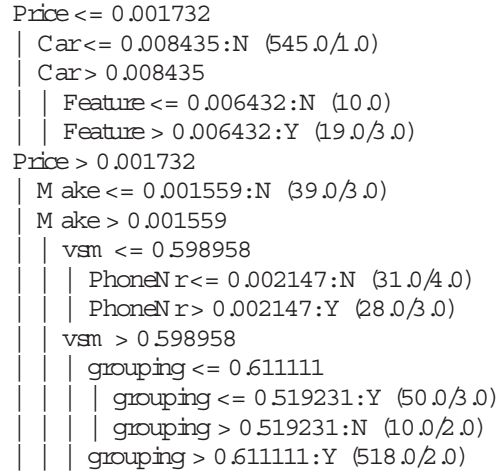
Figure 14 and Figure 15 respectively show the classification models the learners built for car ads and obituaries. Each classification model contains three decision trees for an application ontology  $O$ , which are for multiple-record documents, single-record documents, and application forms. Within one tree, a node denotes a predicate using heuristics measures. For example, “vsm” and “grouping” are two measures for a text component  $t_d$  computed based on the heuristics calculations of expected-values,  $Similarity(t_d, O)$ , and grouping,  $Grouping(t_d, O)$ . The “density” is a density measure  $Density(t_d, O)$  of the text component  $t_d$  with respect to the application ontology  $O$ —for either car ads or obituaries. The object-set names “Car” and “Deceased Person” denote the densities of the keywords specified for the object sets of interest in the two application ontologies. Other object-set names denote densities of lexical values and keywords for object sets in the two application ontologies. For example, *Make* denotes a density  $Density(t_d, Make)$  of the text component  $t_d$  with respect to the lexical object set *Make* in the application ontology for car ads. The parenthetical numbers  $(x/y)$  following “Y” and “N” for a decision-tree leaf  $L$  give the total number of training examples  $x$  classified for  $L$  and the number of incorrect training examples  $y$  classified for  $L$ .

Given the decision trees for the two applications in Figure 14 and Figure 15, we can see that the learners used different combinations of heuristics to check the relevancy. For both applications, however, the learners largely exploited density heuristics. Figure 14 shows that the learner trained for the car-ads application applied only the density heuristics to classify application forms. Figure 15 shows that “vsm”, the expected-values heuristic, was not useful for the learner of the obituaries application.



(a) Form decision tree

(b) Multiple-record tree



(c) Single-record tree

Figure 14: Classification model for car ads

<pre> FuneralDate &lt;= 0.002303   DeathDate &lt;= 0.022727     Deceased Person &lt;= 0.009384:N (34.0/4.0)     Deceased Person &gt; 0.009384:Y (5.0/1.0)   DeathDate &gt; 0.022727:Y (5.0) FuneralDate &gt; 0.002303:Y (18.0/1.0) </pre>	<pre> DeathDate &lt;= 0.005002   FuneralDate &lt;= 0.010844:N (119.0)   FuneralDate &gt; 0.010844:Y (9.0/4.0) DeathDate &gt; 0.005002   density &lt;= 0.118065:N (7.0)   density &gt; 0.118065:Y (68.0/5.0) </pre>
---	--

(a) Form decision tree

(b) Multiple-record tree

```

grouping <= 0.277778
| RelativeName <= 0.047308:N (109.0/1.0)
| RelativeName > 0.047308
| | DeathDate <= 0.00202:N (4.0/1.0)
| | DeathDate > 0.00202:Y (4.0)
grouping > 0.277778
| Deceased Person <= 0.023715
| | BirthDate <= 0.036281
| | | DeathDate <= 0.001617
| | | | FuneralDate <= 0.013723:N (7.0/1.0)
| | | | FuneralDate > 0.013723:Y (3.0)
| | | DeathDate > 0.001617
| | | | IntermentAddress <= 0.022565
| | | | density <= 0.070368
| | | | | Deceased Person <= 0.006863
| | | | | density <= 0.040834:N (3.0/1.0)
| | | | | density > 0.040834:Y (14.0)
| | | | | Deceased Person > 0.006863:N (3.0)
| | | | | density > 0.070368:Y (36.0)
| | | | IntermentAddress > 0.022565:N (4.0/1.0)
| | | BirthDate > 0.036281:N (3.0)
| | Deceased Person > 0.023715:N (7.0)

```

(c) Single-record tree

Figure 15: Classification model for obituaries

### 6.3 Experiments

For each application, we performed two sets of experiments. First, we measured the precision, recall, and the F-measure of our approach, including an investigation of how sensitive the performance is with respect to the analysis of linked pages and the application of the form-filling methods in [23, 27]. Second, we conducted studies to evaluate the contributions of heuristics: *Densities*, *Expected Values*, and *Grouping*. For all the experiments, we evaluated the performance of the learners on the test documents described in Table 4.

#### 6.3.1 Results

We evaluated the performance of our approach based on three measures: precision, recall, and the F-measure. Given (1) the number of relevant documents  $N$  determined by a human expert, (2) the number of correct relevant documents  $C$  selected by our approach, and (3) the number of incorrect relevant documents  $I$  selected by our approach, we computed the recall ratio as  $R = C/N$ , the precision ratio as  $P = C/(C + I)$ , and the F-measure as  $F = 2/(1/R + 1/P)$ . We report all these values as percentages in Table 5.

Application	Number Applicable Doc.'s	Number Correct	Number Incorrect	Recall %	Precision %	F-Measure %
Car Ads	40	39	2	98	95	96
Obituaries	40	38	2	95	95	95

Table 5: Results of the test algorithm in Figure 7

Observe that two negative documents and one relevant document for car ads were classified incorrectly (two incorrect positive responses and one incorrect negative response), and that two relevant documents and two negative documents for obituaries were classified incorrectly (two incorrect negative responses and two incorrect positive responses).

#### 6.3.2 Form Filling and Linked Pages

As explained earlier, in addition to the text components that appear in a document, we can also exploit auxiliary information such as linked pages or retrieved documents obtained by form filling. First, we applied a strategy that prefers a better precision ratio. The learner used the algorithm in Figure 11 to re-evaluate the application forms that the learner classified as positive responses using the test algorithm in Figure 7. Figure 16(a) shows a form to locate car dealers, which the algorithm in Figure 7 incorrectly classified as a form for car ads. By applying the methods to fill in forms, we retrieved the document in Figure 16(b), which contains dealer information rather than car ads. With this additional information, the learner caught the incorrect positive response for the document in Figure 16(a) and classified it as negative, irrelevant to the car-ads

## Dealer Locator

Enter The Zip Code, Select Either U.S.A Or Canada  
And Select A Manufacturer

Zip  


Country

Manufacturer



- BMW
- Quick
- Cadillac
- Chevrolet
- Chrysler
- Dodge
- Ford
- GEM
- GMC
- Honda
- Hyundai

(a) Dealer locator



Records 1 - 2 (2)

Click on the Location Name for details about the location.  
The  icon next to the Location Name indicates Special Events at that Location.

---

	<p>Name <a href="#">Bob Smith BMW</a></p> <p>Address 7050 Topanga Canyon Blvd., Canoga Park CA 91303</p> <p>Phone (818) 346-3144</p> <p>Country United States</p> <p>Specialty BMW</p>	 <a href="#">Map &amp; Directions</a>
---	--	---

---

	<p>Name <a href="#">Century West BMW</a></p> <p>Address 4245 Lankershim Blvd., Universal City CA 91618</p> <p>Phone 800 447-8871</p> <p>Country United States</p> <p>Specialty BMW</p>	 <a href="#">Map &amp; Directions</a>
---	--	---

(b) Retrieved document by filling in form in Figure 16(a)

Figure 16: An incorrect positive response for car ads

### UNIVERSAL VEHICLE SEARCH

YEAR
MAKE
MODEL
STATE

Fill in one or more of the above fields and click GO to search >>>

Leave a field blank to see all of those listings...

(a) Car form

### UNIVERSAL SEARCH RESULTS

Select a column to sort these results in alpha-numeric order.  
Uncheck the sort box to sort the results in reverse order

<input checked="" type="radio"/> YEAR	<input checked="" type="radio"/> VEHICLE	<input checked="" type="radio"/> LOCATION	<input checked="" type="radio"/> MILES	<input checked="" type="radio"/> PRICE
2001	Porsche 911 CARRERA 4 Cabriolet - LOADED!	NY Scarsdale	400	\$120000
2001	Mitsubishi GALANT GTZ	UT South Jordan	10000	\$22000
2001	Ford EXPLORER SPORT	TX Dallas	22000	\$18500
2001	Ford ESCAPE XLT	FL Hallandale	2000	\$24000
2001	Chrysler PT CRUISER	VA Salem	2500	\$20000
2001	Chevrolet IMPALA LS	CA Redding	12000	\$20500
2001	Chevrolet CAVALIER LS	VA Rustburg	2000	\$10000
2000	Volvo S70	AL Montgomery	36000	\$25500
2000	Volkswagen JETTA GLX	VA Falls Church	27000	\$21500
2000	Volkswagen CABRIO GLS Convertible	PA Huntingdon Valley	13000	\$18500
2000	Toyota TUNDRA	NC Winston Salem	25000	\$22500
2000	Toyota TUNDRA	CA Grass Valley	19000	\$21000

(b) Retrieved document by filling in form in Figure 17(a) (partial)

Figure 17: An incorrect negative response for car ads

application. With this strategy the learner improved its precision from 95% to 98%. Second, we applied a strategy that prefers a better recall ratio. The learner re-evaluated the negative responses using retrieved documents obtained by applying form filling. Figure 17 shows both a form in Figure 17(a) that the learner classified as irrelevant using the test algorithm in Figure 7 and the relevant document in Figure 17(b) obtained by filling in the form in Figure 17(a). By using the algorithm in Figure 11 to re-evaluate the form in Figure 17(a), the learner caught the incorrect negative response and improved its recall from 98% to 100% for the test set of the car-ads application in Table 4.

The other auxiliary information we use in our system is information on linked pages. Figure 18(a) shows a document that contains only partial obituaries. By applying the algorithm in Figure 9, the learner caught the incorrect negative response by considering the linked pages—for example, the relevant document in Figure 18(b) for the first link in Figure 18(a). Thus the

06/19: Kanderis, Mike M.

06/16: Torres, Josephine P.  
**Born:** 02/01/1924

06/11: Larsen, Rex Farrell  
**Born:** 04/06/1916

06/11: Oliver, Donna Joy  
**Born:** 09/15/1924

06/10: Anderson, Doug N.  
**Place of death:** Mayfield, Utah  
**Born:** 03/11/1920

06/10: Atkin, Lee Clawson  
**Place of death:** St. George, Utah  
**Born:** 12/11/1932

06/10: Brady, Brittany Kensie  
**Born:** 07/28/1983

06/10: Cass, Betty Louise  
**Place of death:** American Fork, Utah  
**Born:** 10/09/1923

### Obituary: Mike M. Kanderis

1925 ~ 2003

Mike M. Kanderis of Littleton, Co, preceded in death by his wife Grace; father of Shane, Dennis and Jennifer; brother of Stella Melonakis and Mary Spencer. Also survived by six grandchildren.

Active member of El Jebel Shrine, ROJ #138 and Sandblasters.

Visitation Wednesday, 4-7 p.m. Horan & McConaty Family Chapel, 31091 So. Wadsworth Blvd. Funeral services Thursday, 2 p.m. St Catherine Greek Orthodox Church, 5555 So. Yosemite, Greenwood Village, CO. Scottish Rite Rose Croix Services Friday 12 noon, at Municipal Cemetery, Grand Junction, CO.

Memorial remembrances to Shrine Hospitals for Children, 4625 W. 50th Ave, Denver, CO 80212

(a) Obituaries that require re-evaluation (partial)

(b) A linked page (partial)

Figure 18: An incorrect negative response for obituaries

learner improved its recall measure from 95% to 98%. Note that we included 10 HTML documents containing only partial obituaries similar to the HTML document in Figure 18(a) in the test documents. The test result shows that the learner for obituaries missed only one such page without analyzing linked pages. That means that the partial obituaries usually provided enough informative information for the learner to classify them as positive, relevant documents.

### 6.3.3 Other Incorrect Positives and Negatives

Figure 19 shows an incorrect positive response for the obituaries application, a marriage submission form that contains the keyword “obituary”. The form in Figure 19, however, requires that a user manually fill the text fields. Thus the full automatic classification procedure cannot catch this incorrect positive response.

Figure 20 shows an incorrect positive response, a motorcycle-sale page (Figure 20(a)), for the car-ads application; and an incorrect positive response, a bibliography of an American hero

Enter your Bride's Surname:

Enter your Bride's Given Name(s):

Enter your Groom's Surname:

Enter your Groom's Given Name(s):

Enter the date of the marriage dd Month yyyy : *Please format your dates.*

 [Date Format Help](#)

Enter your Michigan County:

Enter your Source (ie: license number, obituary, biography) in the space provided below.

Figure 19: An incorrect positive response for obituaries (partial)

(Figure 20(b), for the obituaries application. The motorcycle document contains data for *Year*, *Make*, *Mileage*, *Price*, and *PhoneNr*. The bibliography document contains data such as the person's name, birth and death dates, and relationships including his father and his daughter. Both documents include concepts that largely overlap those specified in the application ontologies. It is difficult for the learners to recognize that documents with significant overlap do not apply to the application ontologies.

The final incorrect response is a negative response for the document in Figure 21 for the obituaries application. The learner classified this document as irrelevant because the density measures obtained based on the entire document text component were not high enough to reach the thresholds defined in the single-record tree for obituaries. The reason is that a large amount of irrelevant data appears in the document. In order to catch this incorrect negative response, instead of evaluating the document text component that appears in the entire document, we can evaluate a subpart of the document text component that contains the information of interest. For example, we can select the text in Figure 21 that describes exactly the singleton obituary for *Bill Gilley*. To see what would happen if we were to select only the applicable part of the document, we manually revised the document and discarded the irrelevant text. The classification model built for obituaries gave a positive prediction for the revised document. Automatically selecting potentially relevant subcomponents of a document is challenging, and we have not yet resolved the issues and implemented a solution.



**AUTOMOBILES : MOTORCYCLES****'99 SUZUKI DS80**

\$1100; '97 Yamaha RT100, \$900; '85 Honda 70CC 3 wheeler, \$400; 410-282-1462

First date  
ad ran:  
**Jan 23 2000**

Mark ad ☐

**4 WHEELERS**

'96 HONDA 200, \$2300; '94 Polaris 400, \$2100; '98 Yamaha Blaster, \$2300; '87 Yamaha Banshee, \$2000; '95 Kwasaki Mojave 250, \$1900; '87 Majave 110, \$1200; '97 Kasea 50, \$700; '85 Honda 70CC 3 Wheeler, \$400; 410-282-1462

First date  
ad ran:  
**Jan 23 2000**

Mark ad ☐

**YAMAHA SNOWMOBILE**

540CC Less than 300 miles. First \$1500 410-287-6662

First date  
ad ran:  
**Jan 23 2000**

Mark ad ☐

**'84 HONDA Nighthawk**

S 11K mi. Exc cond. \$2200. 410-343-1448

First date  
ad ran:  
**Jan 22 2000**

Mark ad ☐


**'81 HONDA CR125,**

runs exc, rebt trans & mtr. \$750/OBO. 410-661-7193

First date  
ad ran:  
**Jan 22 2000**

Mark ad ☐

(a) Motorcycles (partial)



Captain John Smith


Jamestown  
REDISCOVERY

[Home](#) [History](#) [Smith](#)

Findings  
Exhibits  
History  
Visiting  
Publications  
Resources  
Contact  
Donations

APVA

*This portrait of Captain John Smith appeared on a 1616 map of New England. The image is colorized by Jamie May from an original engraving by Simon de Passe.*



Virginians know that Captain John Smith was one of the first American heroes. But because he was a proud and boastful man, it is difficult to know which parts of his life are fact and which are fiction. What many people may not know is that Smith's adventures started even before Jamestown.

Born in 1580 in Willoughby, England, John Smith left home at age 16 after his father died. He began his travels by joining volunteers in France who were fighting for Dutch independence from Spain. Two years later, he set off for the Mediterranean Sea, working on a merchant ship. In 1600 he joined Austrian forces to fight the Turks in the "Long War." A valiant soldier, he was promoted to Captain while fighting in Hungary. He was fighting in Transylvania two years later in 1602. There he was wounded in battle, captured, and sold as a slave to a Turk. This Turk then sent Smith as a gift to his sweetheart in Istanbul. According to Smith, this girl fell in love with him and sent him to her brother to get training for Turkish imperial service. Smith reportedly escaped by murdering the brother and returned to Transylvania by fleeing through Russia and Poland. After being released from service and receiving a large reward, he traveled all through Europe and Northern Africa. He returned to England in the winter of 1604-05.

(b) Captain John Smith (partial)

Figure 20: Two incorrect responses for car ads and obituaries



**Honored swim instructor Bill Gilley dies**

By BILL ZECHMAN  
State Correspondent

**McMINNVILLE, Tenn.** ?A longtime swim instructor has died just two days after the new McMinnville public swimming pool was dedicated in his honor.

Bill Gilley, who had headed the American Red Cross swimming program for Warren County since 1973, died early yesterday morning at age 76. He had an apparent heart attack a few minutes before he was due to enter Saint Thomas Hospital for an evaluation for a pacemaker.

**Today's Top Stories**

- \* Midstate anxiously awaiting Bonnaroo
- \* Teenagers are losing summer jobs to older workers
- \* Breiden signs lottery bill, expects March ticket sales
- \* Head-on crash kills 1, injures 2
- \* Two motorists perish as storm slaps Midstate
- \* 14% hike in tuition backed by chancellor

**We Want To Hear From You!**

The *Tennessean* wants your opinion on how we can improve [www.tennessean.com](http://www.tennessean.com). Please take a moment and respond to our brief Web site survey.

☐ E-Mail This Article  
☐ Printer-Friendly (text only)  
☐ Subscribe To The Tennessean

Figure 21: Single-record obituary (partial)

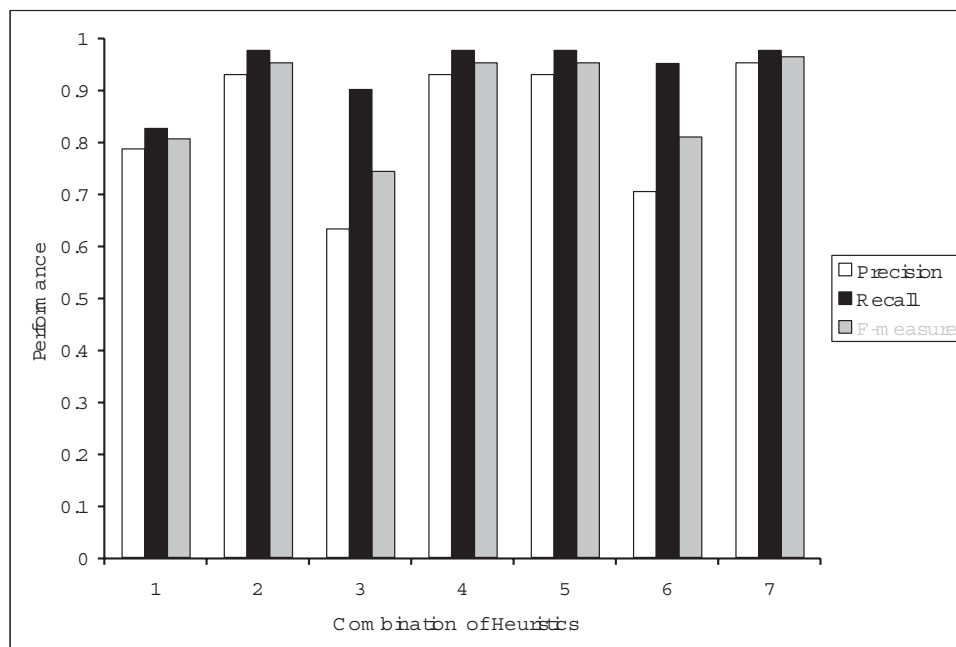
Combination	Densities	Expected-Values	Grouping
1			+
2	+		
3		+	
4	+		+
5	+	+	
6		+	+
7	+	+	+

Table 6: Combination of three kinds of heuristics

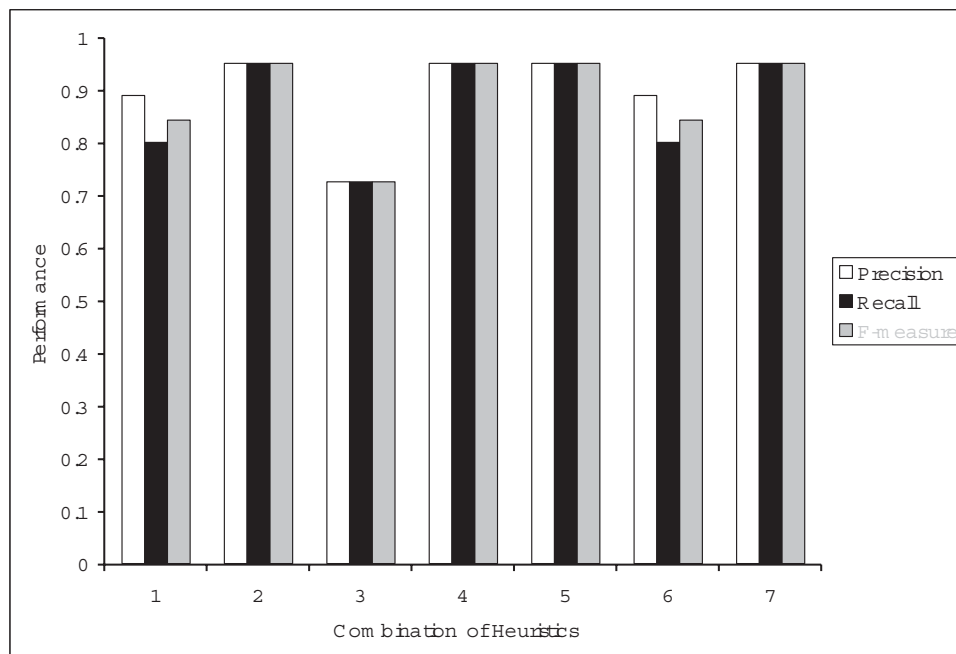
### 6.3.4 Contributions of Heuristics

We evaluated the performance by applying different combinations of heuristics using the test algorithm in Figure 7. Figure 22 shows the contribution of the three kinds of heuristics (densities, expected-values, and grouping) to the overall performance in the two applications. The x axis lists the seven combinations of the three kinds of heuristics. Table 6 shows the seven combinations, where “+” denotes the heuristic or heuristics in the corresponding column that are in use.

For both applications, Figure 22 shows that the density measures are important (Columns 2, 4, 5, and 7 are the best). When the learners exploited only density heuristics in the algorithm of Figure 7 to evaluate the test documents for the applications (Column 2), the learners achieved above 90% for all measures (precision, recall, and F-measure). Using the expected-values heuristic alone (Column 3), the learners achieved an F-measure of only about 70% for the two applications. Using the grouping heuristic alone (Column 1), the F-measure obtained by each learner was still less than 85%. Even when the learners used both the expected-values heuristic and the grouping heuristic together (Column 6), performance for neither application improved. Figure 22(a) shows that the learner of the car-ads application achieved the highest F-measure by applying all heuristics together.



(a) Cars ads application



(b) Obituaries application

Figure 22: Performance comparison of heuristics combinations

We know that the density heuristics are dependent on and sensitive to the specification of the application ontologies. The other two heuristics, expected values and grouping, are also mainly determined by the specification of the application ontology. Thus, when porting to a new application domain, as long as the application ontologies are well defined, our empirical evaluation shows that our approach should be able to recognize relevant HTML documents with high precision, recall, and F-measure. Moreover, as the application ontologies evolve, for example, more lexicons become available for *Make* or *Color* is added as a new object set, the performance of the approach will most likely to improve as well.

## 7 Related Work

Many papers about the broad areas of filtering and information retrieval have appear in recent years. (See recent surveys for filtering [31] and information retrieval [32].) Most of these papers are not relevant to our approach to filtering. The papers we do review in this section are those that are most recent and closest to our work.

User-profile-based filtering techniques have been extensively investigated in the context of content-based information filtering research. Most content-based information filtering systems are intended for unstructured text and typically use sets of keywords to represent user interests. The Stanford Information Filtering Tool (SIFT) [8] is a well-known content-based text filtering system for Internet news articles. A user subscribes to a SIFT server with one or more profiles, each of which includes a query supported either by a Boolean model or vector space model. Queries in both models are based on keywords. In contrast, our application ontology expresses information of interest in terms of concepts and relationships, which adds an enriching semantic description beyond keyword-based profiling. Moreover, the application domain of SIFT is only text documents, whereas our system works for HTML documents which, in addition to text, could include HTML tables, forms, and linked sub-documents.

In order to retrieve documents with higher precision, some researchers have resorted to enriching documents by adding meta information. WebKB [4], for example, is an ontology-based knowledge retrieval tool that interprets semantic statements stored in Web documents. WebKB allows the addition of meta-information, indexes, and constructed ontologies that subsume WordNet. With this added information, WebKB can evaluate user queries over the annotated documents that combine lexical, structural, and knowledge-based techniques to retrieve documents. Another example, [5], describes OWLIR as an approach to retrieve documents that contain both free text and semantically enriched markup. The OWLIR framework advocates the interdependency of search and inference for precise retrieval over semantic content. Both WebKB and the OWLIR framework are largely dependent on the accuracy of semantic markup for documents and queries, which is obtained based on information extraction techniques as well as ontologies.

Faithfully marking up documents, however, relates to the knowledge acquisition bottleneck faced in the AI research community of eighties, and there is little practical experience on which to rely [6, 7]. In contrast to WebKB and the OWLIR framework, we issue queries to filter application objects on the fly among unstructured Web documents without enriching Web documents by adding meta-information and without putting documents into repositories and indexing them.

TAP [33] provided a set of simple mechanisms for sites to publish data onto what it perceives the Semantic Web to be and for applications to consume this data via a query interface called *GetData*, a lightweight query representation language. A set of HTML scrapers dynamically locate and convert relevant pages in source sites into machine readable data and thus make them available on the "Semantic Web." TAP's semantic search augments traditional search results with relevant data aggregated from distributed sources on the semantic Web. In contrast, our approach works on the current Web without presumption that machine-understandable documents are supported.

With XML being used as a standard format to exchange data on the Web, filtering Web documents based on both content and structure has become more feasible. The XFilter system [34] and the YFilter [35] system are examples of XML filtering systems. The XFilter and YFilter engines use models based respectively on finite state machines and non-deterministic finite automata to locate and evaluate user profiles. With knowledge of structures and content of XML documents, users are able to express profiles in XPath [36]. In contrast, our approach is more widely applicable and scalable because it uses a fixed application ontology that works over a dynamic set of unstructured documents on the Web instead of specifying profiles basing on structures and metadata of a particular set of XML documents.

Within our research lab, we have used several approaches [37, 25, 26] to categorize multiple-record Web documents. A multiple-record Web document contains multiple unstructured records, one after another. The work reported in [25] and [26] respectively evaluates a multiple-record document by applying a statistical multivariate analysis and a logistic regression analysis. In contrast, we provide an evaluation model based on machine learning techniques. We reported some of these results in an initial report on our work [37]. In this paper we have expanded our earlier work (1) by also including single-record documents, forms, and linked subdocuments, (2) by providing a vastly expanded explanation of problematic pages, and (3) by improving our heuristics and running new experiments. Specifically, with regard to new heuristics, we improved our density heuristic by considering every attribute individually in addition to considering them all collectively. With this new density heuristic we generated new and better filtering rules. Overall, our results improved as well as our coverage, having added single-record documents, forms, and documents with linked subdocuments.

## 8 Conclusions and Future Work

We presented an approach for filtering HTML documents by application ontologies. Once an application ontology is created, we can use a machine learning algorithm over a set of heuristics to produce a classification model that accurately recognizes which documents apply to the ontology. Results for the tests we conducted showed that the recognition F-measure, precision, and recall were above 95% for both a car-ads application and an obituaries application. We also showed that we can further improve performance by considering linked pages and documents retrieved by submitting default forms.

Our approach is robust, flexible, and scalable. The heuristics, learning algorithms, and training documents used in the approach are extensible. If new heuristics appear useful, we can immediately use them without having to change our fundamental approach. The performance of a learner can be incrementally improved by adding more training documents. When porting to a new application domain, our approach is able to achieve high precision and recall if the application ontology represents the application domain well. As the application ontology evolves, the learner is likely to improve its performance as well.

Our future work can expand in several different directions. (1) We can test our approach on more applications. (2) We can investigate ways to enhance the heuristics. (3) We can do a deeper level analysis of an HTML document and check relevancy based on an appropriate subpart of the document rather than the entire document. (4) We can apply a meta-learning strategy to train the learners over several different classifiers including C4.5 as described in this paper, multivariate analysis [25], and logistic regression [26].

## References

- [1] G. Furnas, T. Landauer, L. Gomez, and S. Dumais. The vocabulary problem in human-system communications. *Communications of the ACM*, 30(1):964–971, March 1987.
- [2] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, Menlo Park, California, 1999.
- [3] N.J. Belkin and B.W. Croft. Information filtering and information retrieval: Two sides of the same coin. *Communications of the ACM*, 35(12):29–38, December 1992.
- [4] P. Martin and P. Eklund. Embedding knowledge in web documents. *Computer Networks*, 31(11–16):1403–1419, 1999.
- [5] U. Shah, T. Finin, A. Joshi, R.S. Cost, and J. Mayfield. Information retrieval on the semantic web. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management (CIKM 2002)*, pages 461–468, McLean, Virginia, November 2002.
- [6] S. Dill, N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingran, T. Kanungo, K.S. McCurley, S. Rajagopalan, A. Tomkins, J.A. Tomlin, and J.Y. Zien. A case for automated large scale semantic annotations. *Journal of Web Semantics*, 1(1):115–132, December 2003.

- [7] A. Kiryakov, B. Popov, I. Terziev, D. Manov, and D. Ognyanoff. Semantic annotation, indexing, and retrieval. *Journal of Web Semantics*, 2(1):49–79, December 2004.
- [8] T.W. Yan and H. Garcia-Molina. The SIFT information dissemination system. *ACM Transactions on Database Systems*, 24(4):529–565, December 1999.
- [9] J. Aslam, K. Pelekhev, and D. Rus. Using star clusters for filtering. In *Proceedings of the Ninth International Conference on Information and Knowledge Management (CIKM 2000)*, pages 306–313, McLean, Virginia, November 2000.
- [10] J. Callan. Learning while filtering documents. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 98)*, pages 224–231, Melbourne, Australia, August 1998.
- [11] J. Mostafa, S. Mukhopadhyay, W. Lam, and M. Palakal. A multilevel approach to intelligent information filtering: Model, system and evaluation. *ACM Transactions on Information Systems*, 15(4):368–399, 1997.
- [12] 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.
- [13] V.C. Storey, D. Dey, H. Ullrich, and S. Sundaresan. An ontology-based expert system for database design. *Data & Knowledge Engineering*, 28(1):31–46, October 1998.
- [14] Y. Wand. A proposal for a formal model of objects. In W. Kim and F.H. Lochovsky, editors, *Object-Oriented Concepts, Databases, and Applications*, pages 537–559. ACM Press, New York, 1989.
- [15] M.A. Bunge. *Treatise on Basic Philosophy: Vol. 3: Ontology I: The Furniture of the World*. Reidel, Boston, Massachusetts, 1977.
- [16] M.A. Bunge. *Treatise on Basic Philosophy: Vol. 4: Ontology II: A World of Systems*. Reidel, Boston, Massachusetts, 1979.
- [17] D.W. Embley, C. Tao, and S.W. Liddle. Automatically extracting ontologically specified data from HTML tables with unknown structure. In *Proceedings of the 21st International Conference on Conceptual Modeling (ER2002)*, pages 322–327, Tampere, Finland, October 2002.
- [18] D.W. Embley, C. Tao, and S.W. Liddle. Automating the extraction of data from HTML tables with unknown structure. *Data & Knowledge Engineering*, 54(1):3–28, July 2005.
- [19] G. Salton and M.J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, 1983.
- [20] J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, California, 1993.
- [21] Homepage for Bob Howard Honda, January 2002. <http://www.bobhowardhonda.com>.
- [22] World Wide Wheels classifieds of cars, May 2003. <http://www.wheels.com>.

- [23] 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.
- [24] S.W. Liddle, D.T. Scott D.W. Embley, and S.H. Yau. Extracting data behind web forms. In *Proceedings of the Joint Workshop on Conceptual Modeling Approaches for E-business: A Web Service Perspective (eCOMO 2002)*, *Lecture Notes in Computer Science (LNCS 2784)*, pages 402–413, Tampere, Finland, October 2002.
- [25] J. Tang. A probabilistic model for binary categorization of multiple-record web documents. Master's thesis, Brigham Young University, Provo, Utah, March 2001.
- [26] Q. Wang and Y. Ng. An ontology-based binary-categorization approach for recognizing multiple-record web documents using a probabilistic retrieval model. *Journal of Information Retrieval*, 6(3–4):295–332, September–December 2003.
- [27] S.H. Yau. Automating the extraction of data behind web forms. Master's thesis, Brigham Young University, Provo, Utah, December 2001.
- [28] D.W. Embley, N. Fuhr, C.-P. Klas, and T. Roelleke. Ontology suitability for uncertain extraction of information from multi-record web documents. In *Proceedings of the Workshop on Agenten, Datenbanken und Information Retrieval (ADI'99)*, Rostock-Warnemuende, Germany, 1999.
- [29] 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.
- [30] D.W. Embley, Y.S. Jiang, and Y.-K. Ng. Record-boundary discovery in web documents. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data (SIGMOD'99)*, pages 467–478, Philadelphia, Pennsylvania, May/June 1999.
- [31] A. Paepcke, H. Garcia-Molina, G. Rodriguez-Mula, and J. Cho. Beyond document similarity: Understanding value-based search and browsing technologies. *SIGMOD Record*, 29(1):80–92, 2000.
- [32] M. Kobayashi and K. Takeda. Information retrieval on the web. *ACM Computing Surveys*, 32(2):144–173, 2000.
- [33] R. Guha, R. McCool, and E. Miller. Semantic search. In *The Twelfth International World Wide Web Conference*, pages 700–709, Budapest Hungary, May 2003.
- [34] M. Altinel and M.J. Franklin. Efficient filtering of XML documents for selective dissemination of information. In *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB'00)*, pages 53–64, Cairo, Egypt, September 2000.
- [35] Y. Diao, M. Altinel, M.J. Franklin, H. Zhang, and P. Fischer. Path sharing and predicate evaluation for high-performance XML filtering. *ACM Transactions on Database Systems*, 28(4):467–516, December 2003.
- [36] J. Clark and S. DeRose. XML path language (XPath) version 1.0, W3C recommendation. <http://www.w3.org/TR/xpath>, November 1999.



- [37] D.W. Embley, Y.-K. Ng, and L. Xu. Recognizing ontology-applicable multiple-record web documents. In *Proceedings of the 20th International Conference on Conceptual Modeling (ER2001)*, pages 555–570, Yokohama, Japan, November 2001.