2002

Automated Ontology Learning for a Semantic Web

Christopher N. Hammack
University of Nebraska - Lincoln, chammack@cse.unl.edu

QingFeng Lin
J. D. Edwards & Company

Hai Huang
University of Nebraska - Lincoln

Stephen Scott
University of Nebraska - Lincoln, sscott2@unl.edu

Sharad C. Seth
University of Nebraska - Lincoln, seth@cse.unl.edu

University of Nebraska-Lincoln, Computer Science and Engineering
Technical Report # TR-UNL-CSE-2002-0005

Automated Ontology Learning for a Semantic Web

Christopher N. Hammack[†], QingFeng Lin[‡], Hai Huang[†],
Stephen D. Scott[†], and Sharad C. Seth[†]

[†]Dept. of Comp. Sci. & Eng.
University of Nebraska
Lincoln, NE 68588-0115
{chammack,sscott,seth}@cse.unl.edu

[‡]J. D. Edwards & Company
One Technology Way
Denver, CO 80237
qingfeng_lin@jdedwards.com

February 8, 2002

Abstract

By expressing web page content in a format that machines can understand, the *semantic web* provides huge possibilities for the Internet and for machine reasoning. Unfortunately, there is a considerable distance between the present-day World Wide Web and the semantic web of the future. The process of annotating the Web to make it semantic web-ready is quite long and not without resistance. In this paper one mechanism for semanticizing the Web is presented. This system is known as AutoSHOE, and it is capable of categorizing pages according to one of the present HTML semantic representations (Simple HTML Ontology Extensions) by Heflin et al. We are also extending this system to other semantic web representations, such as the Resource Description Framework (RDF). The AutoSHOE system includes mechanisms to train classifiers to identify web pages that belong in an ontology, as well as methods to classify pages within an ontology and to learn relations between pages with respect to an ontology. The modular design of AutoSHOE allows for the addition of new ontologies as well as algorithms for feature extraction, classifier learning, and rule learning. This system has the promise to help transparently bridge traditional web technology to the semantic web using contemporary machine learning techniques rather than tedious manual annotation.

1 Introduction

Established web-search methods employ expression matching to identify candidate pages, and prune or reorder these by further analysis based on the traditional information retrieval techniques or the web topology [4, 3]. It would be considerably more useful if the knowledge contained on the web could actually be expressed in some way that machines could understand, and that this information could be interpreted. This is the promise of the *semantic web* [7, 6]: that all information that is available to humans on the web could be expressed in a way that machines can interpret.

One way to express this semantic information is SHOE [11] (Minipage, Section B). SHOE (as well as other semantic web models) uses a set of HTML tags to describe semantic information. Unfortunately, SHOE does not address the problems of efficiently adding these tags to a page and of making novice web-site designers aware of SHOE ontologies. Manually coding these tags into a page is very tedious and time-consuming and the toolset knowledge required is significantly more than many web site developers possess. Further, many developers do not see any benefit to placing the ontologies on their pages.

It would be extremely useful to be able to automatically generate these semantic tags using machine learning techniques (Minipage, Section C). This is the purpose of AutoSHOE. Given a web page that belongs to a certain category of pages (ontology), it can classify pages according to subclasses and infer the relationships between certain types of pages. Using similar technologies, AutoSHOE can also detect whether or not pages belong to a particular ontology prior to further classification. This allows for rapid collection of data known to be relevant to a particular ontology.

Machine learning techniques have been successfully applied to information extraction, text document classification, relation rule learning and many other information processing areas (e.g. [17]). However, machine learning systems for ontologies can be difficult to build due to lack of background in machine learning, the tedious work of gathering and processing labeled training data, and the need to try several feature selection methods to find one that works best for a specific ontology. (On the other hand, once training is done, annotating new pages is relatively easy and cheap.) A major design goal of AutoSHOE was to mitigate these difficulties in several ways.

- It obtains trained learners through the Internet to annotate unlabeled web pages.
- It simplifies the training process and the sharing of training data and trained learners.
- It integrates machine learning systems to learn SHOE ontologies in a seamless way.

With this framework, the users can collect online SHOE-annotated pages as training data, experiment with different feature selection methods and learning algorithms to find the best approach for learning a particular ontology, and automatically annotate new web pages with SHOE's annotations. As a framework, AutoSHOE is highly extensible, sharable and customizable, and it is extendible to other semantic web representations, such as the Resource Description Framework

(RDF). AutoSHOE can be accessed[1] at http://autoshoe.unl.edu and a more detailed description of its architecture is available from Lin [14] and Lin et al. [15].

The rest of this paper is organized as follows. In the next section we take the reader through a step-by-step demonstration of AutoSHOE. We then describe AutoSHOE's architecture in more detail in Section 3 and conclude in Section 4. In addition, minipages give background on ontologies, SHOE, and machine learning.

2 AutoSHOE Walk-Through

We will illustrate the functionality of AutoSHOE by means of the cs-dept-ontology [9]. First, labeled data is collected for training a classifier to determine if a new (arbitrary) web page is an instance of an ontology (i.e. if it belongs in any of its classes). Then this data is further used to train other classifiers to classify data within an ontology (i.e. this classifier will, when trained, predict a category for a new page known to be in its ontology) and to identify relationships between pages within this ontology. Then these classifiers can be used to annotate new web pages. When a new web page is encountered for annotation, AutoSHOE first decides if the page is relevant to the ontology. If it is, further analysis of the page's contents determines the annotation tags and the discovered relationship of the page to other web data.

The user of AutoSHOE accomplishes these tasks by a step-by-step process that we now describe by means of screen shots of the actual system.

2.1 Obtaining Training Data

Figure 1 shows the AutoSHOE form that the user fills out to collect a set of web pages as training data. The user provides a set of starting URLs and the recursive depth to which the WebGrabber should descend for collection of web pages. Further constraints on the retrieved training set can be specified by the user: an upper bound on the number of pages retrieved from any starting URL; the maximum file size (to prevent grabbing documents that may not be useful for classification); and the hostnames to visit. The last option is useful, for example, to limit the search to a single university's web site.

The modular framework of AutoSHOE allows a sophisticated user to choose a learning algorithm for classification while providing a default option for those not well-versed in machine learning.

After the form is filled out, the data collection process begins. As each page is visited, the features (see Minipage on Machine Learning) of the page are harvested and stored for training the classifier.

[1] While a complete prototype of AutoSHOE has been built, the final port to its server is still in progress.

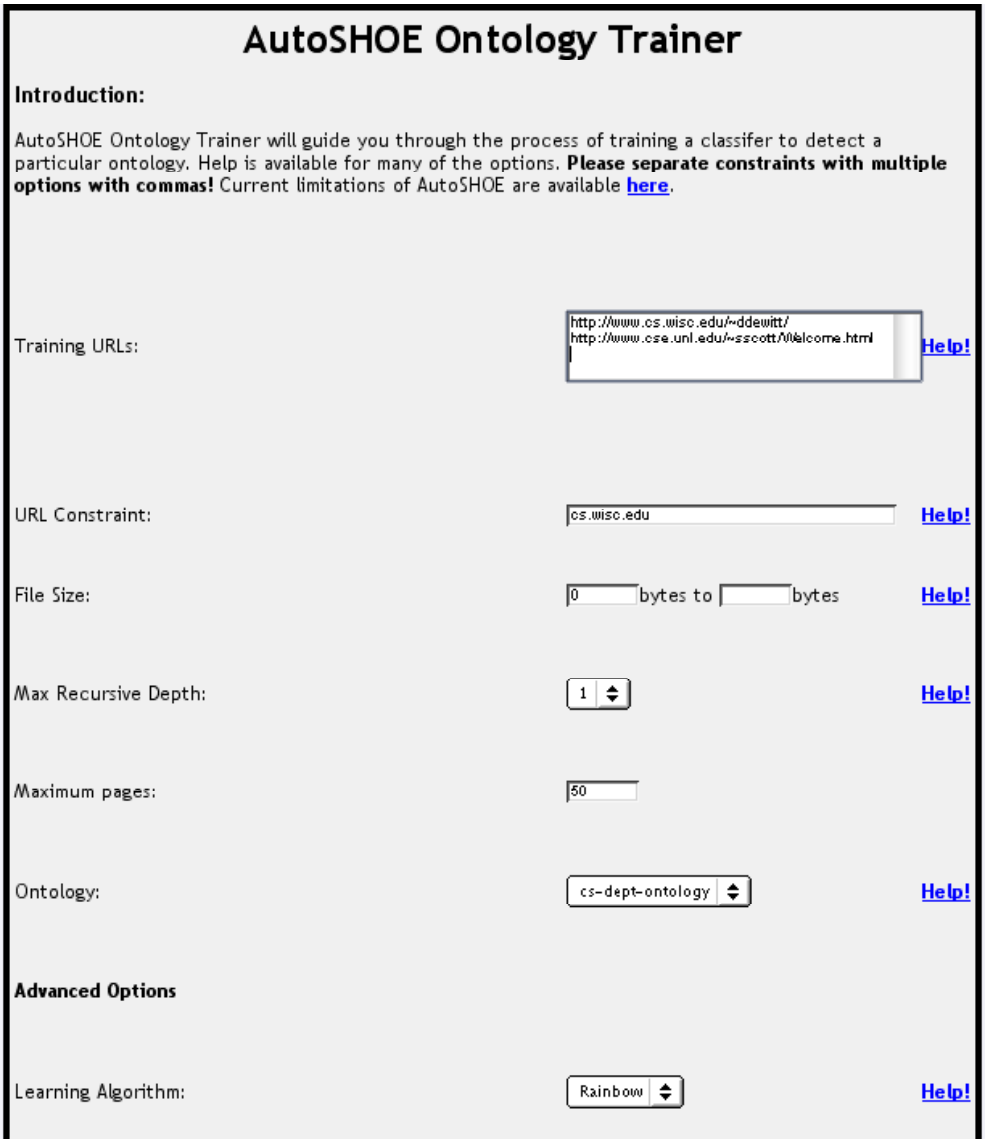


Figure 1: AutoSHOE form for specifying training data.

2.2 Learning if a Page Belongs to an Ontology

At the end of the data collection process, the user is presented with a list of the pages found (Figure 2). Since learning requires labeled data, the user must provide the labels. This is done by placing a check mark next to each page that is a positive example (i.e. each page that is relevant to the ontology). The unchecked pages are used as negative examples (pages irrelevant to the ontology). After the user finishes labeling the examples, the learning algorithm determines the features that best characterize classification in the ontology. The default learning algorithm package is Rainbow: a naïve Bayes algorithm (see machine learning Minipage).

2.3 Learning Classification Rules for SHOE Ontologies

For a page that belongs in an ontology, it is also necessary to learn the predefined categories in the ontology. For example, inside the cs-dept-ontology, a page might belong to the csFaculty class.

The training process for this phase is essentially identical to learning if a page belongs in the ontology—web sites are collected and extracted, the user labels the pages with their classes in the ontology, (e.g. cs.Student), and the learning algorithm builds a classifier. As in Section 2.2, the default learning algorithm package is Rainbow.

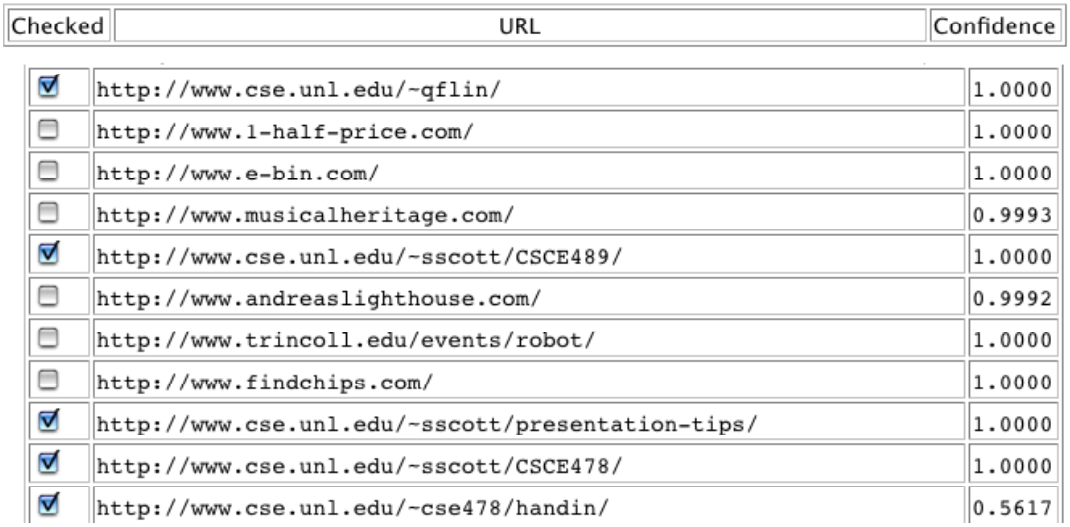
After learning the classification rules, AutoSHOE can be trained to recognize binary relationships between pages. For example, one relationship in the cs-dept-ontology is 'cs.member', which establishes a relationship between pages belonging to a CS Department and one of its members. AutoSHOE asks the user to identify pairs of web pages and the binary relationship that exists between these pairs. Then the algorithm FOIL is applied to learn the relations. As relational learning often takes a long time, an option is provided to enter an e-mail address and be notified when the process has been completed.

2.4 Testing Auto-Annotation

After the training is done, we can use the learned classifiers and rule sets to annotate new web pages. This interface is nearly identical to the training interface (Figure 1) aside from the addition of a feature to select which learned classifier to use. Although the annotation process is similar to training, this is the most likely feature for users to utilize, so we give a detailed example. First we collect a set of new HTML documents that we wish to label with respect to the ontology that we trained our classifier for. In this example, we are collecting a set of web pages from Dr. Stephen Scott. (In order to provide some negative examples to make this example more interesting, no URL constraint is used. If only pages from the University of Nebraska were desired, a constraint such as 'unl.edu' could be used.) A recursive search of depth 2 is used, meaning that two levels of links will be visited from the starting URL.

The ontology classifier then attempts to determine if the pages collected belong to the ontology, so that only pages which belong to the ontology are further classified. The system will return the list of collected URLs, with its classification presented through the check box next to each URL. The user should now verify that the pages returned correctly belong to the ontology. If the page is incorrectly classified as being in the ontology, the user should deselect the check box, and vice-versa. In this example (Figure 2), 47 pages were returned, 28 of which were selected as belonging to the ontology. In the screen shot below, pages such as http://www.musicalheritage.com/ were identified by AutoSHOE as not belonging to the ontology, while several course related pages did belong.

Next the system will extract features from each page (this is similar to the procedure used in training the classifiers) to represent the HTML document as a feature vector. This feature vector will



Checked	URL	Confidence
☑	http://www.cse.unl.edu/~qflin/	1.0000
☐	http://www.1-half-price.com/	1.0000
☐	http://www.e-bin.com/	1.0000
☐	http://www.musicalheritage.com/	0.9993
☑	http://www.cse.unl.edu/~sscott/CSCE489/	1.0000
☐	http://www.andreaslighthouse.com/	0.9992
☐	http://www.trincoll.edu/events/robot/	1.0000
☐	http://www.findchips.com/	1.0000
☑	http://www.cse.unl.edu/~sscott/presentation-tips/	1.0000
☑	http://www.cse.unl.edu/~sscott/CSCE478/	1.0000
☑	http://www.cse.unl.edu/~cse478/handin/	0.5617

Figure 2: Specifying membership in SHOE ontologies.

be used as an unlabeled instance. The unlabeled instances are then inserted into the database. In order to annotate the web pages with SHOE's classification rules, AutoSHOE sends the unlabeled instances from the database to the classifier. The classifier then labels these instances. In order to annotate the web pages with SHOE's relation rules, we extract the rule sets from the database and send them to a deduction program such as Prolog. The deduction program will discover the target relation according to the rule sets.

AutoSHOE presents the results (Figure 3) in two parts: the web pages classified with confidence intervals and the SHOE mark-up. In this example, Dr. Scott's research page is classified as belonging to the cs.Faculty class with a 99% confidence value. A suggestion box on one of his course pages is identified as a cs.Course page with 45% confidence. A list of courses on one of his course pages is incorrectly identified as a cs.Student page with 89% confidence. This is not unusual since this sort of page is similar to what many students may have on their web site explaining what courses they are currently taking. The second part of the results page, the SHOE mark-up, lists all of the classifications in SHOE syntax. This information could be stored in a new database indexed by URL to be used for various semantic web applications.

Ontology Classification Results

Label	URL	Confidence
cs.Student	http://www.cse.unl.edu/~sscott/CSCE496-CB/courselist.html	0.8917
cs.Course	http://www.cse.unl.edu/~sscott/CSCE496-CB/suggestion-box.shtml	0.4550
cs.Faculty	http://www.cse.unl.edu/~sscott/./research/grants.shtml	0.9963
cs.Faculty	http://www.cse.unl.edu/~sscott/./awards.shtml	0.9468
cs.Faculty	http://www.cse.unl.edu/~sscott/./profservice.shtml	0.9908
cs.Faculty	http://www.cse.unl.edu/~sscott/./univservice.shtml	0.9877
cs.Student	http://www.cse.unl.edu/~qflin/	0.9975
cs.Student	http://www.cse.unl.edu/~dchawla/	0.9544

SHOE Markup

```
<INSTANCE KEY="chammack1">
<USE-ONTOLOGY id=cs-dept-ontology URL="http://www.cs.umd.edu/projects/plus/SHOE/onts/

<CATEGORY FOR="http://www.cse.unl.edu/~sscott/CSCE496-CB/" NAME="cs.Course">
<CATEGORY FOR="http://www.cse.unl.edu/~sscott/CSCE970/" NAME="cs.Course">
<CATEGORY FOR="http://www.cse.unl.edu/~sscott/CSCE488/" NAME="cs.Course">
<CATEGORY FOR="http://www.cse.unl.edu/~sscott/CSCE496-ML/" NAME="cs.Course">
<CATEGORY FOR="http://www.cse.unl.edu/~cse478/handin/" NAME="cs.Course">
<CATEGORY FOR="http://www.cse.unl.edu/~sscott/CSCE478/" NAME="cs.Course">
```

Figure 3: SHOE's classification results.

3 AutoSHOE Architecture

AutoSHOE is not a simple software package but rather a machine learning *framework* since it has the following features:

- It is middleware: There are several general-purpose machine learning systems in existence. However, these usually require users to have machine learning background knowledge. In addition, these learning systems require different input formats and run on different platforms. Therefore, even knowledgeable users require time and effort to become familiar with them. On the other hand, SHOE's design is independent of any machine learning techniques. General-purpose learning algorithms cannot take the SHOE-annotated documents as training data without proper data preprocessing, feature selection, and format translation. As a middleware component between the end users and the learning systems, AutoSHOE hides the complex details of the learning systems. It provides a uniform and friendly user interface so that the AutoSHOE users can use the system without knowing the sophisticated machine learning techniques. In addition, AutoSHOE hides the background knowledge of SHOE from the learning systems. It provides a standard learner control protocol to communicate with different learning systems so that the learners can run without knowing SHOE's ontology and annotation.

- It is highly extendible: AutoSHOE doesn't have a pre-defined system boundary. The training data, learners, and feature selectors can be located anywhere so long as they are accessible via the web. New learners and feature selectors can be plugged into the framework by simply filling out an online registration form.

- It is highly sharable. Even though the details of learning are hidden from the user, training a learner to learn an ontology can be time-consuming. It requires AutoSHOE to grab a large amount of labeled training data, select the features from raw data, translate them into the format that learners can understand, and train the learners. It also requires a human to try many different feature selection methods and learning algorithms to find the best approach to learn the ontology. However, once training is finished, annotating new pages is relatively easy and cheap. In order to share the information among the AutoSHOE users, the training data, feature selectors, learners, intermediate processing results, trained classifiers and learned rule sets are stored in the database and can be accessed through the Internet.

- It is highly customizable: AutoSHOE is a database-driven system. The interfaces to the external entities, such as AutoSHOE users and machine learning systems, can be customized according to different requirements. AutoSHOE makes it easy to customize the user interface, the data output format, and even the communication protocol with different learners.

- It is online and interactive. Getting labeled training data can be expensive since human intervention is required. The accuracy of annotation will be poor when the training data are not sufficient to get a good classifier. In order to accumulate more labeled training data, AutoSHOE can run in an online interactive model, i.e. the users first submit a small

amount of training data and AutoSHOE learns from this primary set and annotates more pages. Then a human verifies the annotations. These newly-annotated pages can then be added as new training data for further use.

AutoSHOE is a three-tiered architecture (Figure 4). The first layer is the presentation layer, which is a set of HTML and Java Server Pages. The end users use the presentation layer to collect data, train classifiers, and annotate new web pages. The middle layer is the tools layer, which is a set of Java Objects. These tools are responsible for searching, collecting, filtering, storing and maintaining data. On the back end are the third party machine learning algorithms, which are used for feature selection and learning SHOE's classification and relation rules. These systems can be written in any language and can run on any server. They plug into the AutoSHOE framework by using system adapters. These adapters talk to the broker in AutoSHOE and map the training data to its format.

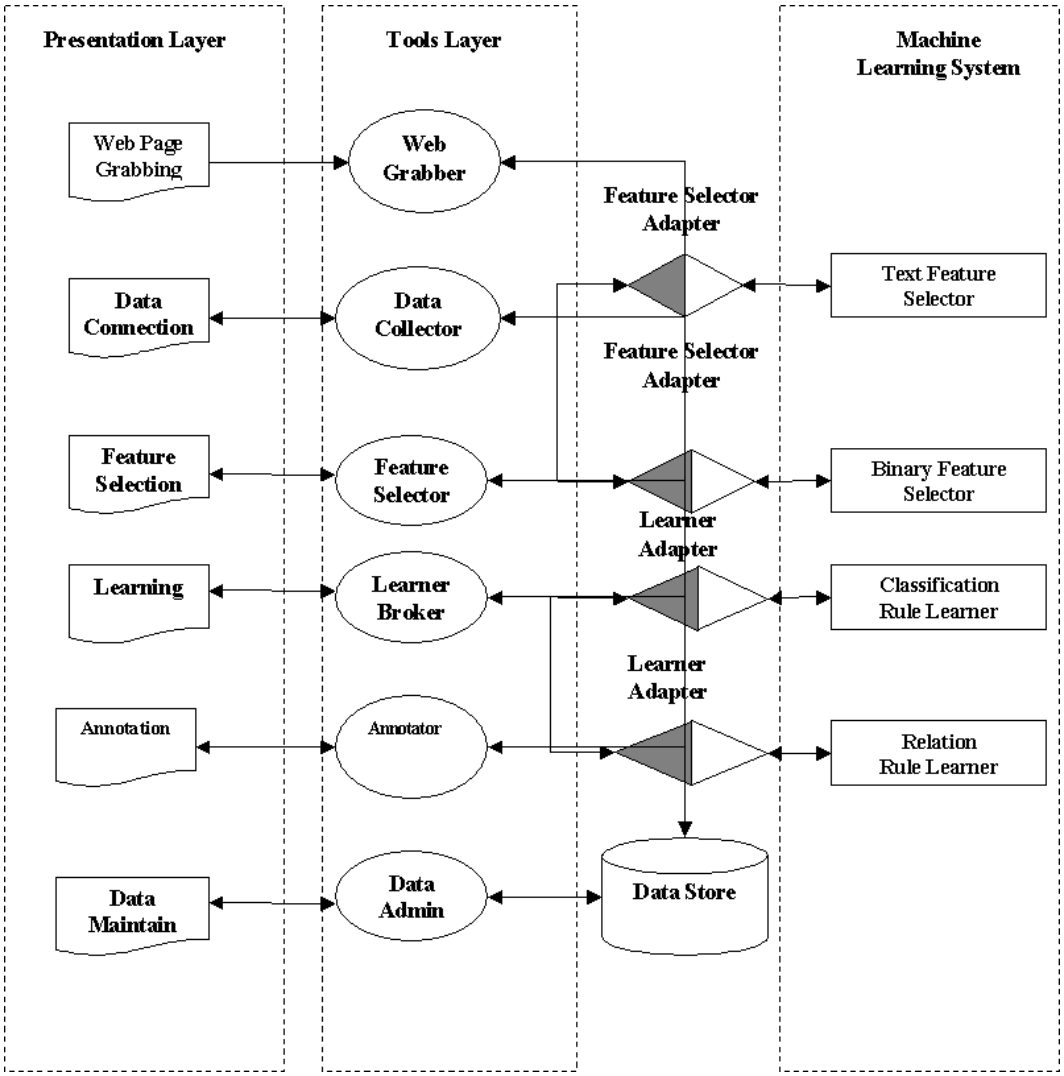


Figure 4: System architecture of AutoSHOE.

AutoSHOE builds SHOE ontologies by learning the characteristics of certain types of web pages. Web pages are harvested from the Internet using a module known as WebGrabber, and the *features* are extracted from the web page using a set of rules. These features provide key information to the learning algorithm for classification. This process is known as *feature extraction*. After the features have been extracted, the system uses various learning algorithms to classify the pages. AutoSHOE is designed in such a way that any type of learning algorithm should be usable using a pluggable interface, though the user is also provided built-in classifiers: a naïve Bayes classifier for web page classification [13, 16, 2] and an implementation of the FOIL algorithm [19, 21, 1] to induce relationships between pages.

After the training is done, we can use the learned classifiers and rule sets to annotate new web pages. The annotation process is similar to training. First we collect a set of new HTML documents that we wish to label with respect to the ontology that we trained our classifier for. Then we follow the same feature selection procedure to represent the HTML document as a feature vector. This feature vector will be used as an unlabeled instance. We can also parse the hyperlinks in the web pages, which will be used as the background relations to deduce the target relation. The unlabeled instances and the hyperlinks will be inserted into the database. In order to annotate the web pages with SHOE's classification rules, we send the unlabeled instances from the database to the classifier. The classifier then labels these instances. In order to annotate the web pages with SHOE's relation rules, we extract the rule sets and the background relations from the database and send them to a deduction program such as Prolog. The deduction program will discover the target relation according to the rule sets and the background relations.

4 Conclusions and Future Work

We have presented an online machine learning system that can automatically grab SHOE-annotated pages as training data and use machine learning techniques to learn classifiers and rule sets for a SHOE-defined ontology. As the middleware between the complex machine learning systems and the end users, AutoSHOE provides a uniform and easy-to-use environment so that even inexperienced users can use this framework to learn an ontology (i.e. learn classifiers to identify membership in and labels within an ontology) and annotate new pages. These labeled pages can then be manually verified and used to train new classifiers. As a framework, AutoSHOE is highly extensible: it allows new learning algorithms and feature selection algorithms to be dynamically added into the system. AutoSHOE is highly sharable: it allows the training data, feature selectors, learners to be allocated anywhere on the web, and the training results can be accessed from anywhere through web. AutoSHOE is also highly customizable: it allows users to tailor the interface. Finally, our AutoSHOE prototype successfully proved the concept of an online machine learning center.

Web classification is not a simple task, but even better results are not far down the road. Improved results are likely with better parsing, feature selection (e.g. hidden Markov model-based tools [8]), and training sets, all of which can be plugged into AutoSHOE. Another possible improvement would be to use multiple classifiers, using multiple sets of features.

The system is currently undergoing a period of rapid development and redeployment. In the next several months, we will improve the overall reliability and usability of AutoSHOE, while finishing the port of the original prototype. The system currently has some limitations in its HTML parser as well as difficulties parsing many kinds of dynamic web pages. These issues are currently being addressed. Additionally, we hope to publish formal specifications so that anyone may build an AutoSHOE module with little assistance from the AutoSHOE team, and use it for classification in the system.

In addition to extensions to the architecture, another possible improvement would be the inclusion of additional semantic representation parsers, particularly RDF [5]. RDF will become the W3C standard for semantic web representation. We plan to allow the user to choose between annotation in SHOE and RDF in the near future.

Acknowledgments

This work was supported in part by NSF grants CCR-9877080 (with matching funds from UNL-CCIS and a Layman Foundation grant) and CCR-0092761. QingFeng Lin performed this work at the University of Nebraska.

References

[1] FOIL6, 2001. `ftp.cs.su.oz.au/pub/foil6.sh`.

[2] MLC++ home page, 2001. `http://www.sgi.com/tech/mlc/`.

[3] Excite home page, 2002. `http://www.excite.com/`.

[4] Google home page, 2002. `http://www.google.com/`.

[5] Resource description framework (RDF), 2002. `http://www.w3.org/RDF`.

[6] Semantic web, 2002. `http://www.w3.org/2001/sw/`.

[7] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):34–43, May 2001.

[8] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to extract symbolic knowledge from the World Wide Web. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98)*, 1998.

[9] J. Heflin. The computer science department ontology, 2002. `http://www.cs.umd.edu/projects/plus/SHOE/onts/cs1.1.html`.

[10] J. Heflin. The SHOE knowledge annotator, 2002. `http://www.cs.umd.edu/projects/plus/SHOE/KnowledgeAnnotator.html`.

[11] J. Heflin, J. Hendler, and S. Luke. Reading between the lines: Using SHOE to discover implicit knowledge from the web. In *Proceedings of the AAAI-98 Workshop on AI and Information Integration*, 1998.

[12] J. Heflin, J. Hendler, and S. Luke. Applying ontology to the web: A case study. In *Proceedings of the International Work-Conference on Artificial and Natural Neural Networks (IWANN '99)*, 1999.

[13] Ken Lang. NewsWeeder: learning to filter netnews. In *Proc. 12th International Conference on Machine Learning*, pages 331–339. Morgan Kaufmann, 1995.

[14] Q. F. Lin. A machine learning framework for automatically annotating web pages with simple HTML ontology extension (SHOE). Master's thesis, Dept. of Computer Science and Engineering, University of Nebraska, May 2000.

[15] Q. F. Lin, S. Scott, and S. C. Seth. A machine learning framework for automatically annotating web pages with simple HTML ontology extension (SHOE). In *Proceedings of the International Conference on Intelligent Agents, Web Technology and Internet Commerce*, 2001.

[16] A. K. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering, 1996. `http://www.cs.cmu.edu/~mccallum/bow`.

[17] T. Mitchell. *Machine Learning*. McGraw-Hill Publishing Company, 1997.

[18] D. Mladenic. Text-learning and related intelligent agents: A survey. *IEEE Intelligent Systems*, 14(4):44–54, July-August 1999.

[19] J. R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.

[20] J. R. Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann, 1993.

[21] J. R. Quinlan and R. M. Cameron-Jones. FOIL: A midterm report, 3-20. In *Proceedings of the European Conference on Machine Learning*, 1993.

A Minipage: Web Ontologies

In order to let a computer "understand" the semantic meaning of web pages, we need a structured way to represent knowledge in the web pages. A common approach to representing the knowledge is using a *web page ontology* [8]. An ontology specifies the *categories* (or *classes*) and *relations* of interest. The ontology defines a hierarchy of categories and relations involving the categories. A relation may be defined between two categories or between a category and a constant datum (this kind of relation is also called an *attribute* or a *feature* of the category). A web page is an *instance* of a particular ontology if it belongs to one of the categories in this ontology.

Figure 5 is a simple university ontology. The boxes in the top part of the diagram represent categories, identified by their name in bold face. They are shown arranged in a hierarchy defined by the "is-a" relation. The relations involving each category appear underneath its name. Some relations describe the relationship between this category and other constant data. For example the *Department* category has the relation *NameOf* which associates a department with its name. Other relations involve two categories; for example, the *Faculty* category has the relation *TeacherOf*, which describes the relationship between a *Faculty* category and a *Course* ontology. At the bottom of the diagram are some web-page instances of this ontology. A web page that belongs to a particular category inherits all the relations of this category. The relations between two categories are shown by arrows in the diagram. For example, the *Faculty* web page is shown to have the relation *Teacher.Of* with the course web page, and so on. A category can also inherit the relations from its super category, thus the category *Faculty* will also have the relation *DeptOf*, inherited from *Person*.

Finding and labeling training data for building new classifiers brings up a somewhat controversial and subjective point. It is important to quantify exactly what qualifies as a match to the ontology. An important (and open) question is exactly what are the criteria for ontology membership—is a page membership classification merely based on the content of the page or is it a member through association?

For example, obviously the page of a CS professor describing her research should belong to the University ontology. However, is a page on the CS professor's website describing her children actually an instance of this ontology? Semantically, there is little information on that page that has to do with the description of the Computer Science Department. However, it can also be argued that by association, this information is related. Similarly, is a student's page that contains essentially no information really an instance of the University ontology? It can be argued that the addition of the page symbolizes the existence of the person in the ontology. On the other hand, since the page contains no semantic information, does it even matter if the page is not included? These are all important questions to consider when examining the "results" of AutoSHOE's classifications, when there is some ambiguity as to what exactly is a match.

B Minipage: SHOE

In order to help an intelligent agent "understand" the knowledge on web pages, Heflin et al. [11] proposed a language called Simple HTML Ontology Extension (SHOE). SHOE is a small extension of HTML that allows web page authors to annotate their web documents with respect to one or more predefined ontologies (see the Minipage on Ontologies). SHOE makes intelligent agent software on the web possible by annotating HTML using XML-style notation.

Using such annotations, a web page can subscribe to one or more ontologies and make assertions about categories under the relations defined in the ontologies. Further, SHOE annotations can be used to define categorization rules and relation rules for a new ontology. Figure 6 illustrates the use of SHOE's tags to construct the ontology shown in Figure 5. In this example, the name of the new ontology is declared at the beginning, followed by an indication that SHOE's

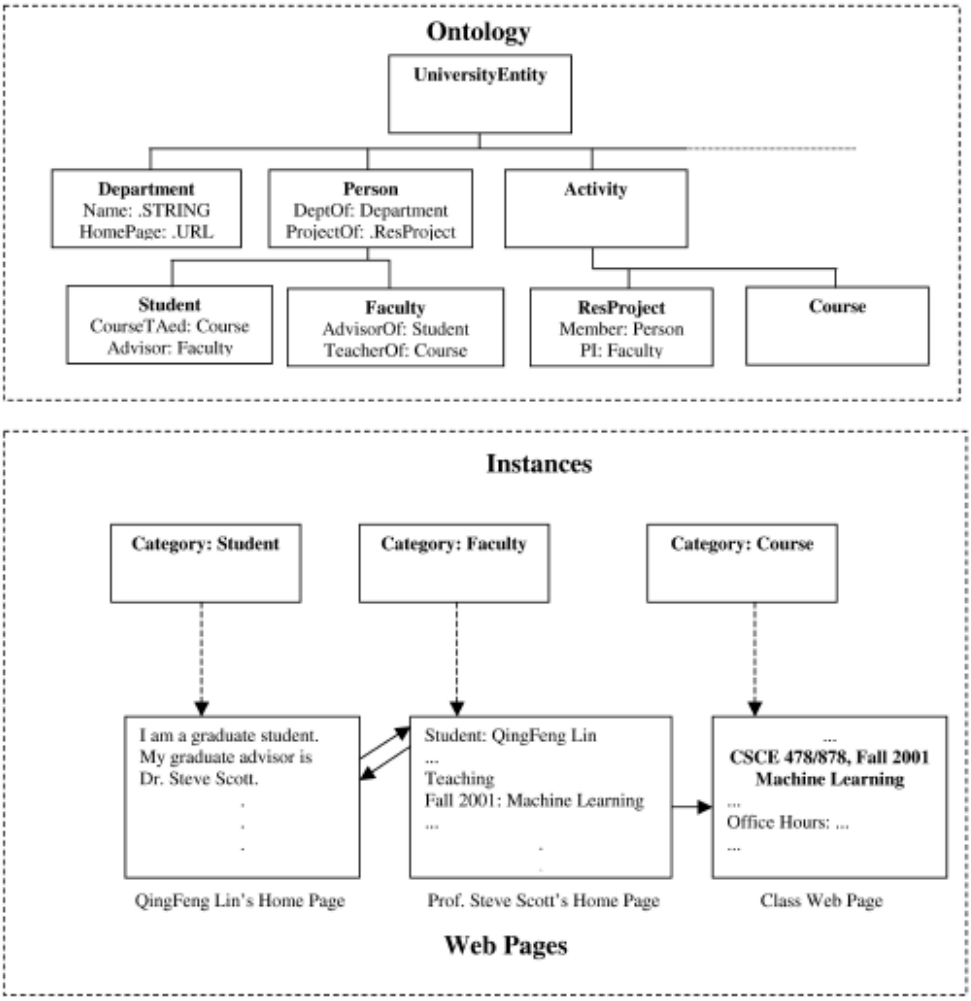


Figure 5: The university ontology.

“base-ontology” Version 1.0 is being borrowed. Next, the category hierarchy is defined by using SHOE’s “is-a” relation. Finally, the various relations are defined.

```
<!--Here we indicate that this document is conformant with SHOE 1.0 -->

 <META HTTP-EQUIV="SHOE" CONTENT="VERSION=1.0">
 <TITLE> Example SHOE annotations to define a new ontology </TITLE>
 </HEAD>
 <BODY>

 <!-- Here we declare the ontology's name and version -->

 <ONTOLOGY ID="An Extended Ontology" VERSION="0.0">

 <!-- Here we declare that we're borrowing from another ontology -->
 <USE-ONTOLOGY ID="base-ontology" VERSION="1.0" PREFIX="base"
URL="http://www.cs.umd.edu/projects/plus/SHOE/base.html">

 <!-- Here we lay out our category hierarchy -->

<DEF-CATEGORY NAME="UniversityEntity" ISA="base.SHOECategory">
<DEF-CATEGORY NAME="Person" ISA="base.SHOECategory">
<DEF-CATEGORY NAME="Department" ISA="UniversityEntity">
<DEF-CATEGORY NAME="Activity" ISA="UniversityEntity">
 ...
<!-- And now we lay out our relationships between categories -->

<DEF-RELATION NAME="MemberOf">
     <DEF-ARG POS="1" TYPE="Department">
     <DEF-ARG POS="2" TYPE="Person">
</DEF-RELATION>
 <DEF-RELATION NAME="AdvisorOf">
     <DEF-ARG POS="1" TYPE="Faculty">
     <DEF-ARG POS="2" TYPE="Student">
 </DEF-RELATION>
 ...
 <!-- Finally, we lay out our other relationships -->

<DEF-RELATION NAME="Name">
     <DEF-ARG POS="1" TYPE="Department">
     <DEF-ARG POS="2" TYPE=".STRING">
 </DEF-RELATION>
 </ONTOLOGY>
```

Figure 6: Defining categorization and relation rules using SHOE's construction tags.

As a first step for creating a machine-understandable Internet, SHOE is useful in many ways [12]. However, SHOE requires manual web-page annotations which can be a tedious and labor-intensive process. SHOE also has other weaknesses:

- SHOE depends on web page authors to annotate their pages and provides an interactive tool (knowledge annotator [10]) for this purpose. However, it is impractical to expect all web page authors to annotate their pages.
- The information provided by the SHOE annotation depends on the ontology that the web page author used. In many cases the ontology used to describe a given web page does not contain the information another user wants to extract. For example, a student may declare that he is a football fan on his web page. However, if only the university ontology is used for annotation, no useful information could be extracted from this declaration.
- The information on the web is dynamic and HTML documents are updated frequently. If the web page author updates the HTML document and forgets to update the SHOE annotation, then the annotation will be inconsistent.

These drawbacks (which also apply to other ontology-based systems on the web) are directly addressed by AutoSHOE.

C Minipage: Machine Learning

Machine learning[2] is a subarea of artificial intelligence in which programs automatically improve their performance with experience. Machine learning techniques have proven to be of great practical value in a variety of application domains. They are especially useful in data mining and in poorly understood domains where humans might not have the knowledge needed to develop effective algorithms.

A well-defined learning problem requires a well-specified task, performance metric, and source of training experiments. Designing a machine learning approach involves several design choices, including the type of training experiments, the function to be learned (called the *target function* or *target concept*), a representation of this function (e.g. a decision tree), and an algorithm for learning the target function from training examples.

The following terms are used frequently in machine learning.

- *Learning Algorithm* or *Learner*: The algorithm that does the learning. Example learning algorithms are C4.5 [20] (which represents its function as a decision tree), naïve Bayes [13] (which represents its function as a probabilistic model), and FOIL [19, 21] (which represents its function as a set of first-order rules). The latter is very useful in learning relations between web pages with respect to an ontology, while learning algorithms like

[2]For an excellent overview of machine learning, see Mitchell [17].

naïve Bayes are useful in learning to classify web pages within an ontology and in learning to predict if a page is relevant to an ontology.

- *Learning System:* The software package that implements a learning algorithm, e.g., Rainbow [16], MLC++ [2] and FOIL6 [1].

- *Attribute* or *Feature*: A variable that takes a value from a pre-defined domain. Examples of attributes are gender (male or female), color (red, green or blue), and temperature (real number). The presence of certain keywords in an HTML document can also serve as attributes, as well as various HTML tags used to emphasize words, e.g. `<b>...</b>` and `<i>...</i>`.

- *Instance* or *Example*: A list of attribute values. An instance is associated with an attribute schema, which defines the names and domains of the attributes. A *labeled instance* is an instance augmented with a special attribute, called a *label* or *category*. A label could be binary (e.g. "positive" versus "negative"), which is useful when learning to predict if a web page belongs to (is relevant to) an ontology. A label can also be multi-valued (e.g. "Student" versus "Faculty" versus "Course"), which is useful when learning to classify a page within an ontology.

- *Dataset:* A set of labeled instances, all associated with the same attribute schema. A *training set* is a dataset on which a learning algorithm is trained. A *test set* is a dataset on which a learning algorithm is tested.

- *Classifier* or *Categorizer:* A function that maps an unlabeled instance to a label. A learning algorithm induces a classifier from a training set.

Mladenic's 1999 survey on text classification [18] provides a good introduction to the field and some of the strategies that have been employed in existing classification packages. A number of packages that perform web classification are discussed. Mladenic also briefly discusses two areas that are particularly important to AutoSHOE: Learning algorithms and feature extraction. For learning algorithms, no particular learning algorithm is singled out as being identifiably the best algorithm for this task. Naïve Bayes and k-nearest neighbor are proposed to be two well-suited algorithms for the task. Interestingly, Mladenic draws the conclusion that feature extraction is likely more important to good performance than the choice of learning algorithm.