

Using Semantics to Identify Web Objects

Nathanael Chambers and James Allen and Lucian Galescu
Hyuckchul Jung and William Taysom

Florida Institute for Human and Machine Cognition
40 South Alcaniz Street
Pensacola, FL 32502
{nchambers,jallen,lgalescu,hjung,wtaysom}@ihmc.us

Abstract

Many common web tasks can be automated by algorithms that are able to identify web objects relevant to the user's needs. This paper presents a novel approach to web object identification that finds relationships between the user's actions and linguistic information associated with web objects. From a single training example involving demonstration and a natural language description, we create a parameterized object description. The approach performs as well as a popular web wrapper on a routine task, but it has the additional capability of performing in dynamic environments and the attractive property of being reusable in other domains without additional training.

Introduction

Searching the World Wide Web involves interacting with web objects such as links, search fields, buttons and forms. Each object is described in natural language, assisting the user in identifying the meaning and purpose of that object. The semantic knowledge in these natural language descriptions can be used to help automate everyday web browsing tasks. Unfortunately, approaches to automating knowledge extraction rely on the syntactic location of objects, and few (if any) semantic indicators are used. This paper describes **WebLearn**, a new approach to identifying web objects based on unifying the semantic content of a user's utterances with the semantic content of the written language surrounding web objects.

Web objects in this paper refers to the basic entities on a web page: text fields, links (anchors), buttons, etc. Most such objects are designed for humans to read, and so contain linguistic information that can be unified with the user's reasons for selecting the object. This gives important clues about the meaning of an object; such semantic information can then be used to identify it more reliably than purely syntactic information about the structure of the web page.

WebLearn's semantic approach also lends itself well to transfer learning. We will show how a user can teach a system where a *books tab* is located, and have the system find a *DVD tab* without extra training. This sort of transfer is possible because we can generalize on semantic parameters, as opposed to only syntactic constraints on page structure.

Copyright © 2006, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

The following section presents some background on web wrappers, followed by a description of PLOW, the larger procedure learning system that uses WebLearn. We then present WebLearn's approach to identifying web objects, followed by evaluations and a discussion.

Background

Web wrappers have been used since the mid nineties to identify web content. A wrapper is a program that turns a semi-structured information source into a structured source (Knoblock *et al.* 2001). This approach can be reliable on static, structured formats, but brittle and sensitive to any changes. Most wrappers define rules based on the syntax around a desired object, and fail if the page is restructured.

Many groups have investigated machine-learning to automatically create syntactic rules. (Lerman *et al.* 2004) developed a system that infers the page layout based on the repeated structure of the page, while also incorporating duplicate information on related "details" pages. (Hsu & Dung 1998) developed a wrapper induction algorithm using finite transducers and (Muslea, Minton, & Knoblock 1999) presented a technique using more expressive rules, transforming web pages into a tree-like structure. (Cohen 2003) discusses many of these wrapper approaches based on structure rather than content.

Some work has used ontologies instead of syntactic cues. (Embley *et al.* 1998) developed a system that formulated rules based on an ontology of words. (Davulcu, Mukherjee, & Ramakrishnan 2002) developed a technique to locate information related to the object taxonomies. These ontology-based approaches can be applied to unstructured web pages, however, their focus is on finding text patterns for the attributes in a given ontology, not on finding web objects.

User goals are largely ignored in the above. Our approach learns how web objects relate to the user's task; we are interested in how a desired object is described, not the structure of a web page.

Procedure Learning on the Web

WebLearn is part of a larger learning system called PLOW (Procedure Learning on the Web), whose purpose is to learn to perform common web tasks. PLOW is taught through natural language coupled with web action demonstrations. For

<p>let me teach you to buy a book go to this page <i>-enters the URL in the browser and hits enter-</i> click the books tab <i>-clicks the tab labeled 'books'-</i> select advanced search <i>-clicks the link labeled 'advanced search'-</i> put the title here <i>-types a book title into the title search field-</i> and put the author here <i>-types an author into the author search field-</i> now click the search button <i>-clicks the search button-</i> now select the page for the book <i>-clicks the link with the title of the book-</i></p>

Figure 1: Dialogue and accompanying browser actions the user performs as the system is instructed how to buy a book.

example, the user speaks, “we put the title of the book here” while simultaneously clicking on a text field and typing the title. The TRIPS dialogue system (Allen, Ferguson, & Stent 2001) controls the interaction with the user and monitors the state of a web browser. TRIPS uses a domain independent ontology and a lexicon of approximately 5000 unique words (excluding morphological derivations).

The system collaborates with the user to learn tasks by discussion. Natural language utterances are automatically parsed and interpreted into a semantic form that is combined with input from an instrumented Mozilla browser (**WebTracker**). When the central agent (**Plow**) receives a semantic representation of the user’s utterance, it must determine if that utterance corresponds to a new step in the procedure, a conditional, a retraction, etc. An example of a discussion with the user is in figure 1. With each new step, Plow asks the procedure learning module (**Palm**) if this next step makes sense, and if so, to add the step to the current procedure. Upon receiving a step and corresponding web action, Palm queries **WebLearn** for information about the web object. WebLearn learns to find instances of the object in the future by using (1) the current web object and (2) the description of the action. WebLearn’s use of the semantic description to identify the object is the focus of this paper.

The procedures and the web objects relevant to each step are learned from a single example performed by the user as he/she teaches the system, step by step, how to perform a specific web procedure. Since the PLOW system was designed for one-shot learning, WebLearn also learns how to identify web objects from a single example. WebLearn is key to procedure learning because few web tasks can be performed without robust object identification.

WebLearn

WebLearn’s main task is to associate the attributes of web objects with the semantic representations of a user’s utterances as he explains his actions.

Input Format

WebLearn’s input for learning is a 2-tuple: an action description and a web object ID. The action description is a frame-like feature-value structure based on a knowledge representation that is parsed automatically from the user’s utterances as he/she teaches PLOW, providing relevant semantic context for the web object. As an example, below is a description for the utterance, *then we click the books tab*.

(*CHOOSE :object (TAB :assoc-with BOOK)*)

WebLearn interacts with WebWatcher to retrieve information about the object by querying for the object and its children’s attributes in the Document Object Model (DOM) of the web page. The following is a list representation of our example object description returned by the browser.

(*object :type a :id a22 :atts ((href “www.amazon.com...”)) :children ((object :type text :id t31 :atts ((content “Books”))))*)

Semantic Relations

WebLearn searches the object description to find a semantic connection between its attributes and the current action. Each ontology class in the input description is expanded into its English realizations through the TRIPS ontology and lexicon. In our example, a relationship can be found between the third class, BOOK, and the text label of the link that the user clicked. BOOK has several linguistic realizations, including *hardcover*, *paperback* and *book*. We easily find that *Book* is a substring of the attribute’s value in the DOM tree, “Books” (the plural, *books*, is an exact match. This paper did not use morphological variations during learning, but instead relied upon them during search in the execution stage). WebLearn parameterizes both the object description and the action description where a relation was found, creating an Object Identification (ID) Rule to associate the action with the object; the action description is used as context for identifying the object in the future. Figure 2 shows a full representation of the learned knowledge in an Object ID Rule.

There are two parts to an Object ID Rule. The first is the action description, as explained above. The second is the object description. Each object description contains a list of parameter variables and string relations. The string relations describe the relationship between the parameters in the action description and the *linguistic realizations* of the semantic concepts in the object description. In figure 2, note that the variable *?target* has parameterized the *BOOK* concept in the task and the object. The relationship is a *substring* of a *BOOK* lexical item (based on the lexicon/ontology) with an *s* postfix. Currently, our system supports four types of string relations:

- **exact:** The task value exactly matches the object value.
- **substring:** The task value is a substring of the object value.
- **superstring:** The object value is a substring of the task value.
- **keyword:** The object value contains at least half the words in the task value, but not necessarily in order.

```
(IDrule :action (CHOOSE :object (TAB :assoc-with ?target))
:object (link :def (object :type a :atts ((href "http://www.amazon.com/...")))
:children ((object :type text :atts ((content ?target))))
:vars (?target (relation :type substring :prefix " :postfix 's')) :domain ("http://www.amazon.com/..."))
```

Figure 2: An Object Identification Rule. `:vars` is the relationship between the parameter in the `:action` and its location in the `:object`. The `:domain` is a list of webpages on which this rule was trained.

These string relations allow WebLearn to use linguistic information about the object that is not always visible to the user, but that may be present due to human creation of the web pages. For instance, form text fields often have several DOM values that are not rendered by the browser. In our book buying scenario, the user teaches the system to, “*put the author here.*” The object description of the text field is as follows:

```
(object :type input :id input4
:atts ((type "text") (size 40) (name "field-author")))
```

The `name` attribute of the field, which is only used by the browser for submitting the form, contains enough information for us to learn how to identify the correct author field using a substring relation on the lexical instance `author` of the `AUTHOR` concept. Since the string relations are in terms of abstract concepts, they can be applied to find objects that the system was not specifically trained on; for example, the Object ID Rule for finding this author field could be applied to find a title field (search for `field-title`). Such knowledge lends itself well to *transfer learning* and is discussed later.

Execution

Given a set of Object ID Rules, WebLearn can execute any rule that unifies with a semantic action description input. The interface to WebLearn is essentially a semantic one, freeing the calling agent from knowledge requirements on types of web objects and their properties.

The input action description is unified with all of WebLearn’s Object ID Rule actions. The unification is a partial unification in that all of the features do not have to be present, but a penalty is given for missing or altered features. As long as the variables unify, the actions are considered a match and the extra features are simply penalized in a similarity metric. This metric allows for flexible action inputs and if multiple actions match, the one with the least edit distance is chosen.

The variable bindings from the action unification are then bound to the object description in the matched Rule, producing a new instance of the object that is used to query WebWatcher for all matching objects. The variable (`?target`) is bound with the unified value using the constraints specified by the matching relation (exact, substring, superstring, keyword). For increased robustness, we make use of the TRIPS ontology and lexicon to further expand the range of natural language expressions. For each unified variable, WebLearn checks the value against the lexicon and ontology to see if it is present as a semantic class or lexical item:

- If a semantic class is found, all possible words under that class are included in the query to WebWatcher.

- If a lexical item is found, all morphological derivations of the word are used.

As an example, we use this text child from figure 2:

```
(object :type text :atts ((content ?target)))
```

If our input binds the action’s `?target` to the value, “*Hard Times*”, with the *exact* relation, the expansion algorithm inserts a regular expression search string into this object’s `?target` variable that would match, among other possibilities; *hard times*, *harder time*, *hard timing*, and *harder timed*. This approach of utilizing semantic input and linguistic knowledge provides for robust execution that finds semantically related objects with linguistically different realizations.

When multiple web objects are found, they are ranked according to a combined metric based on string similarity and tree similarity. The string similarity compares the unified values in the target object to the returned objects. We use a string edit distance score based on a modified version of the Generation String Accuracy (Bangalore, Rambow, & Whittaker 2000). Generation String Accuracy (GSA) is a metric from the Natural Language Generation community, designed to measure the similarity of two strings:

$$GSA = 1 - (I + D + S + M)/L \quad (1)$$

where I , D , S and M are respectively the number of word insertions, deletions, substitutions and movements. Movements are the same as one deletion and one insertion elsewhere in the string, but are only counted as one penalty rather than two. L is the word length of the target object. We created a new, modified version of this metric, the Lexical Generation String Accuracy:

$$LexGSA = GSA + \quad (2)$$

$$\frac{\lambda * \sum_{i=0}^L morph(x_i) - \sum_{j=0}^{I+D+S+M} disc(y_j)}{L} \quad (3)$$

where $morph(x_i)$ is a function on each word in the target string that returns 0 if no pre-processing morphology is needed and 1 if it is. The pre-processing step finds all morphological variants of words that could be changed to provide exact matches (e.g. ‘*timing*’ matches ‘*time*’). λ is a weight that determines the morphology penalty. We chose $\lambda = 0.2$ for our study. $disc(y_i)$ is a discount function that removes some of the penalty scores based on the part of speech (POS) of the word that was deleted, inserted, etc. Instead of penalizing string differences equally, we take into account the POS differences. Determiners and conjunctions return 0.75, prepositions and quantifiers 0.5, numbers and ordinals 0.25, and all other words 0. These numbers are chosen heuristically based on the semantic content of those types of words. Determiners do not add much content to a

phrase ('a tale' vs. 'the tale'), but ordinals do ('two cities' vs. 'three cities'). We present a string comparison example for illustration:

target: 'a counting of cristo'
found: 'the count of monte cristo'

$morph(count)=1$ to make *counting*. We substitute the word *the* for *a* and delete *monte* so that $S = 1$ and $D = 1$. Each action is also discounted (shown below in 4). The resulting lexGSA is then shown below in (5).

$$disc(y_0 = the) = 0.75 \quad disc(y_1 = monte) = 0 \quad (4)$$

$$lexGSA = 1 - \frac{0 + 1 + 1 + 0}{4} + \frac{\lambda * 1 - 0.75}{4} = 0.32 \quad (5)$$

In addition to the string similarity metric, a tree similarity score is calculated for each object against the ideal object from which WebLearn learned. We use a metric similar to GSA, counting the number of insertions, deletions, and substitutions that would make a given tree equal to the target tree.

$$treeSim = 1 - (I + D + S)/N \quad (6)$$

where I , D and S are respectively the number of node insertions, deletions and substitutions that must be made to the tree structure to be equivalent to the target tree. N is the number of nodes in the target tree. This is expensive to compute, but the objects we deal with are typically primitive web objects with trees that are usually less than 5 levels deep, so the computation is tractable.

The Lexical GSA is combined with this tree score and the highest object score is returned:

$$score = (w * lexGSA) + treeSim \quad (7)$$

The lexGSA is weighted higher than the treeSim using the weight w since the semantic values are the main indicators for correctness. The tree similarity measure affects the ordering of the objects when we have multiple objects with the same strings, or an object that has a vastly different tree structure than expected. Minor variations in the trees can be ignored by setting w relatively high. The lexGSA weight w was set to $w = 10$ for our study. Several values for w were tried during development, and a value emphasizing semantic values over syntactic structure performs better. The optimal value may be website specific. The system currently does not have a *score* threshold at which point it would give up and say the object is too dissimilar from the description, but instead always returns the highest scoring object.

Book Search Evaluation

Evaluations were performed on booksellers amazon.com (Amazon) and barnesandnoble.com (B&N). The overall procedure that the PLOW system used was a book search task in which a user teaches PLOW how to find a book. Our evaluation looks at the performance of WebLearn's execution.

Book Search

For training, the user demonstrates how to find a book's details page just once. The discussion and the user actions are shown in figure 1. The utterances in this evaluation are not

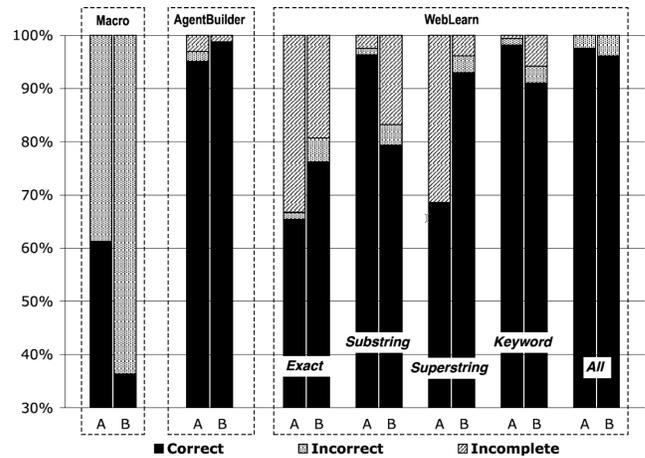


Figure 3: Book search results from (A) amazon.com (162 books) and (B) barnesandnoble.com (155 books). *Incomplete* indicates no web object was returned.

hardcoded into the system, but rather each utterance is automatically parsed and interpreted. Variations in the language commands are also possible. After learning the Object ID Rules for each training step, WebLearn assists in the book search procedure, given an author and a title.

Four evaluators randomly selected 165 book authors, titles, and ISBN numbers from an online ISBN database. The test set was then validated on each website to spot books that are not available for sale. After removing missing books, the set was reduced to 162 for Amazon and 155 for B&N.

The evaluation inserted each author/title pair into PLOW's learned procedure. A test is successful if at the end of the procedure, the system is on the correct book details page. Since the procedure PLOW learns is based on a 6-step procedure, the number of web objects that are tested is actually $6 * n$ where n is the number of books. We searched for 972 objects on Amazon and 930 on B&N.

A baseline evaluation was run as a macro-recorder against which to measure our results. The macro recorder ignores user utterances and simply records web actions. When the user selects a link (or textfield, form, etc.) during training, it records the index of that link on the page (e.g. the 23rd link). During execution, it acts a little differently than a basic macro in that it is parameterized by author and title.

We also compared the performance of WebLearn against a typical web wrapper, the AgentBuilder¹ system. Its GUI was used to build the same list of learned web objects from the procedure described above. The systems were trained separately on Amazon and B&N.

Book Search Results

The results for the title and author book search are shown in figure 3. The baseline macro-recorder performed at 61.3% and 36.4% on Amazon and B&N respectively. The results were much lower on B&N because that page dynamically

¹AgentBuilder is a commercial web wrapper, available at <http://www.agentbuilder.com/>

changes the content of its left selection pane, thus changing the sequential position of the information in the main body and causing the macro-recorder to fail.

WebLearn always found the objects in the first 5 of 6 steps (810 and 775 objects respectively). In essence, the results in figure 3 reflect the performance on the final step, identifying the book link from the search results. For illustration, we show the learned object description for the book link here:

```
(object :type link :atts ((href "http://..."))
 :children ((object :type text :atts ((content ?target))))
```

Note that the learned object has a variable in the text node of the link. Hence, for each book in the test set, we looked for a link with a specific title and used very little other information. As a result, the performance of finding this link was different depending on the type of relationship between the parameter's value and the task's value (the book title). Figure 3 shows the evaluation of each of the four string relations and their performance.

The exact match performed the worst on both Amazon and B&N, but the other three varied depending on the website. Substring does poorly on B&N because the search results use a different object structure when a subtitle is present. Our learned object was trained on a result without a subtitle, so fails to find the link. Superstring performed worse on Amazon because Amazon tends to include very verbose titles, and the shorter test titles fail on these cases. Keyword performs quite well on both sites, with a slightly lower performance on B&N because there tend to be more search results to choose from than on Amazon.

After seeing different results on both websites, and noting that most failures were *incomplete* runs (failure to find a matching object) and not *incorrect* choices, we ran a fifth evaluation that uses a disjunct of the four relationships, trying each relation until a link is found. The disjunct was in order from specific to general: exact, substring, superstring, keyword. The fifth column in figure 3 shows this **All** approach with correct results of 97.5% on Amazon and 96.1% on B&N, well over the 61.3% and 36.4% baseline.

These results are somewhat dependent on the behavior of the websites (Amazon and B&N). The titles and authors stored in Amazon are in a different format than those stored in B&N. To make matters worse, our test set was extracted from an independent ISBN search page with its own formatting. Many of Amazon's books do not include subtitles, while B&N includes them using a different html syntax. Sometimes our test set included subtitles and sometimes not. These exceptions notwithstanding, our approach performs quite well at 97.5% and 96.1% correct.

AgentBuilder's results are also shown in figure 3. On Amazon, WebLearn (using the **All** algorithm) outperformed 97.5% to 96% correct, while on B&N, AgentBuilder outperformed 99% to 96.1%. Both approaches appear effective during execution of their learned actions on this book task. One reason may be because the book search typically returns the correct match as the first choice. If the search results were more ambiguous, AgentBuilder would continue to choose the first link, while WebLearn would try to choose the most relevant link.



Figure 4: Some of the tabs at the top of books.com

While the lack of understanding is the most important drawback for wrapper approaches, it should be noted that they also require a significant amount of effort for training. Counting the number of steps (mouse clicks, text input, spoken utterances) to build our single training procedure, AgentBuilder required 96 GUI actions. Our natural language system required only 18 natural actions.

Transfer Learning Evaluation

WebLearn creates a parameterized description of the object, not a description of its location. Thus, Object ID Rules can be reused in different, yet similar tasks by providing different values for the parameters. The current PLOW system is not yet able to transfer entire procedures, but WebLearn itself is fully capable. We ran a few preliminary evaluations to test the feasibility of training WebLearn on one object and applying the Object ID Rule to find an object within a different context.

Transfer Learning

B&N includes a list of tabs across the top of its page (figure 4), letting the user refine his/her search by visiting a subtopic. By learning the context of a user's *book tab* choice, we can conceivably use this information to find other tabs, such as the DVD or Magazine tab. We performed a simple evaluation of this idea to see if WebLearn could find related links, given a single training example.

Repeating our example, Weblearn finds a relation between the tab type (*book*) and the actual tab on the page:

```
(CHOOSE :object (TAB :assoc-with ?target))
```

We created a test set of action descriptions that could be said by the user and compared WebLearn's returned objects against the desired links. For example, the command, *choose the magazine tab* would return the magazine tab from the list containing the learned books tab. This evaluation was applied on three menus: the left hand pane, the top list of tabs (figure 4), and the left hand pane on Amazon. For each of the three menus, the text of each link or graphical tab was included in the test set, as well as realistic subphrases that a user might use to describe it. For instance, the *Apparel & Accessories* link on Amazon may be described as just *Apparel* or just *Accessories*. Both were included in the set. Amazon's left pane included 80 descriptions (from 55 links), B&N's left pane had 55 (from 37 links) and its tabs list had 25 (from 12 tabs). Each was sent to WebLearn and the system was deemed correct if the expected link for the description was chosen.

	AZ Left	B&N Top	B&N Left
Correct	77 (96.3%)	25 (100%)	53 (96.4%)
Incorrect	2 (2.5%)	0 (0%)	2 (3.6%)
Failed	1 (1.2%)	0 (0%)	0 (0%)

Figure 5: Percent correct in the transfer learning test. *Correct* indicates the expected object was chosen.

Transfer Learning Results

The results for the three transfer learning tasks are shown in figure 5. **AZ Left** is Amazon’s left panel of links, **B&N Top** is B&N’s top list of graphical tab links, and **B&N Left** is B&N’s left panel of links.

Looking at just the correct hits, WebLearn performs extremely well on B&N’s two lists with 100% of the tabs and 96.4% of the links. The tabs are interesting because they are images that are unreadable by a computer without graphical processing. WebLearn finds a string relation using information that is normally hidden from the user, the *alt* attribute of images. WebLearn finds the semantic relationship:

```
(object :type link :atts ((href "http://..."))
:children ((object :type img :atts (.(alt ?target)..))))
```

By learning this very informative attribute, it is able to choose any of the tabs with 100% success.

The performance on Amazon’s left pane is interesting because 66 of the correct matches were not from the left pane, as one would expect. Instead, they were chosen from elsewhere on the page, but they still pointed to the correct web address. At the time of the evaluation, Amazon used javascript to create a duplicate list of links in a different location on the page (hidden from view until the user performs specific mouse gestures). It is from this list that WebLearn chose its results. A user of the system would be oblivious to whether or not the system chose the visually obvious choice. Although unexpected, the choice is still correct in that the effect is to navigate to the correct page. This behavior is possible precisely because WebLearn learns a description of an object, and not a description of the object’s location.

Given the success of transferring knowledge to similar links, we attempted to transfer the entire book search procedure to that of a musical album. This involved selecting a Music tab instead of a Book tab, filling in a search form with different labels, and choosing the CD from the search results based on its title. WebLearn was able to do all of the above based on a single training example about books.

Discussion

The performance of WebLearn is quite good in the book purchasing domain. We achieved performance results of 97.5% and 96% on two websites with limited book information, far above the performance of a parameterized macro recorder. This level of success was achieved with the user showing the system only *one* training example, accompanied by short and natural descriptions of why he/she was clicking on each web object. While the task of clicking a book title link may seem simplistic, it is the recognition that the title is relevant (and not the author) that makes these results exciting.

Our novel semantic approach also performed as well as a commercial web wrapper, AgentBuilder. This book search task is somewhat in a web wrapper approach’s favor in that the search engines of the book sites typically return the desired book as the first link. A wrapper that learns this location will be correct the majority of the time. WebLearn has the clear advantage if an ambiguous search returns a different ordering of results or on pages with dynamic content structure. Also, we showed that WebLearn and PLOW require far less effort to train than traditional wrappers.

Finally, we showed that a parameterized semantic approach can work beyond extracting the specific type of object on which it learned. WebLearn is extendable to other tasks outside of what the user initially intended. Not only did our transfer learning tests of web page menus prove very successful, but WebLearn was then able to search for a musical album based solely on its learning in a book domain. Using the semantic context of a user’s actions to learn cross-domain knowledge will prove critical to building advanced agents that can effectively collaborate with a human. Since this study, we are using WebLearn for other procedures in domains such as weather forecasting, travel planning, publication searches, and several office tasks.

Acknowledgements

This work was supported in part by DARPA grant NBCH-D-03-0010 under a subcontract from SRI International, and ONR grant N000140510314.

References

- Allen, J.; Ferguson, G.; and Stent, A. 2001. An architecture for more realistic conversational systems. In *Proceedings of IUI-01*.
- Bangalore, S.; Rambow, O.; and Whittaker, S. 2000. Evaluation metrics for generation. In *Proceedings of INLG*.
- Cohen, W. W. 2003. Learning and discovering structure in web pages. In *IEEE Data Eng. Bul.*, volume 26, 3–10.
- Davulcu, H.; Mukherjee, S.; and Ramakrishnan, I. 2002. Extraction techniques for mining services from web sources. In *IEEE Int. Conference on Data Mining*.
- Embley, D.; Campbell, D.; Smith, R.; and Liddle, S. 1998. Ontology-based extraction and structuring of information from data-rich unstructured documents. In *International Conference on Information and Knowledge Management*.
- Hsu, C.-N., and Dung, M.-T. 1998. Generating finite-state transducers for semi-structured data extraction from the web. *Information Systems Journal* 23-8.
- Knoblock, C. A.; Minton, S.; Ambite, J. L.; Muslea, M.; Oh, J.; and Frank, M. 2001. Mixed-initiative, multi-source information assistants. In *WWW-01*, 697–707.
- Lerman, K.; Getoor, L.; Minton, S.; and Knoblock, C. 2004. Using the structure of web sites for automatic segmentation of tables. In *ACM SIGMOD Conference*.
- Muslea, I.; Minton, S.; and Knoblock, C. 1999. A hierarchical approach to wrapper induction. In *International Conference on Autonomous Agents*.