Web Agents That Work

Sean Luke
seanl@cs.umd.edu
http://www.cs.umd.edu/~seanl/

James Hendler
hendler@cs.umd.edu
http://www.cs.umd.edu/~hendler/

Department of Computer Science
University of Maryland
College Park, MD 20742

There are two kinds of information-seekers currently wandering the World-Wide Web. First there are us humans, the web-surfers for whom the Web was designed. Second, there are increasing numbers of automated systems, *Web agents*, which gather information from the Web on our behalf. At the present time, humans far outnumber web agents, but this could soon change: as the sheer volume of information on the Web increases, and the ratio of junk to useful information continues to grow, we will increasingly rely on agents to dig through all that muck to find our gems for us.

Web agents come in all shapes and sizes. There are "off-line" agents which gather all the information they can possibly find on the Web, then later let users query this information according to their needs. The most successful off-line agents to date have been text-indexing search engines such as Lycos or AltaVista. There are also "on-line" agents which search the Web with a query in-hand. For example, ShopBot [1] comparison-shops vendor's web pages to find the best price for products the user has requested. Similarly server-push mechanisms, such as screen savers displaying Web headlines, search for and display information according to the user's interests. Lastly, there are "guide" agents which work alongside the user, helping him focus his browsing in real-time as he searches for the information on his own.

In many respects, the World-Wide Web seems ideal for automated information-gathering. The Web has a standardized way of describing where information is found (URLs), a simple way to access it (HTTP), and an application-independent way to describe information (HTML). But automated information-gathering so far hasn't lived up to its promise. Some web agents can intelligently gather information, but only in simplistic, narrow domains (ShopBot being a good example). Others are more general-purpose, but fail because they're not very smart: for example, search engines which return hundreds of thousands of useless query results because they don't really "understand" the content of the data they have gathered. Instead, many of the most successful "agent" companies (like Yahoo!) are those which employ human beings, not computer systems, to gather information.

Why is it so hard to design a reasonably intelligent, general-purpose web agent? Ironically, although HTML is designed for computers to *ma-*

nipulate, it stores information in a way meant only for *humans* to understand. HTML data is mostly in a human-readable text (usually English), laid out for human visual understanding (tables, frames, headlines, visual lists) and interspersed with human-readable pictures and graphics. As bandwidth increases and multimedia continues to make inroads on the Web, this picture will only get worse: if you think that English sentences are difficult for a computer to understand, wait until computers try to comprehend the content of MPEG movies.

As a result, despite the latest advances in networking, knowledge representation, pattern recognition, and natural language processing, web agents are still unable to provide reasonable answers to most simple web queries. Consider the following queries, all of which can all be easily solved by a computer program given a structured database of relevant data. *None* can be solved by general-purpose web agents today:

> I'm doing a report on artist families. Can you find me a picture of a music album performed by someone but composed by a relative of his?
>
> A while ago I met a married couple, last name "Cook", who both work for the same company. The company was involved in Department of Defense initiative 123-45-6789. Find their home pages for me.
>
> I want to go to a school out-of-state but not too far from home. Is there a map of a university in a state bordering Virginia with a ROTC program, Japanese classes, and a Computational Biology major?
>
> Find five internet providers in my vicinity with the lowest rates and a better-than-average customer-satisfaction record.

SHOE: Making Web Agents Possible Today

Perhaps one day in the far future, computer technology will have progressed to the point where computer agents can comprehend Web content in the same way humans do. Until that day, however, we have been investigating an alternative approach to general-purpose, intelligent Web agents: rather than spend all that work designing Web agents that can understand human-only Web content, instead we should be spending time making the Web pages less agent-hostile.

To do this, we have developed a set of HTML extensions that enable Web content-providers to embed in their pages information that computer agents can understand. These HTML tags are collectively known as *SHOE* (Simple HTML Ontology Extensions). SHOE tags enable HTML authors to embed documents with computer-comprehendable information. For example, SHOE lets an author tell Web agents that her web page contains content about a woman whose name is "Helena Cook" and works for Yoyodyne corporation. SHOE could also let an author let agents know that some web page is the home page of a university in North Carolina, and that its course listings, majors and programs, and maps and photos are detailed (in SHOE) on other web pages as indicated. Agents wandering the web can comprehend SHOE data discovered on web pages without needing any natural-language or pattern recognition smarts at all.

SHOE has two parts. The first part is a set of tags for declaring *ontologies*, sets of rules which detail what kinds of claims web authors may make. As a simple example, imagine a popular root ontology about persons, places, and things, available from Ontology Inc. In SHOE, such an ontology would be written as a chunk of HTML code:

```
<ONTOLOGY "root" VERSION="1.0">
    <ONTDEF CATEGORY="Thing">
    <ONTDEF CATEGORY="Person" ISA="Thing">
    <ONTDEF CATEGORY="Place" ISA="Thing">
    <ONTDEF RELATION="name" ARGS="Thing STRING">
    <ONTDEF RELATION="relative" ARGS="Person Person">
</ONTOLOGY>
```

The first line declares the root ontology, version 1.0. The next three lines declare that by using this ontology, SHOE documents may categorize data as people, places, or things, and that people and places

are kinds of things. The next two lines declare that all things can have names, and that people can have other people as relatives.

SHOE recognizes that on the web, information is never static. For this reason, SHOE lets ontologies *extend* parent ontologies to provide for up-to-date, specialized domains. For example, Ontology Inc's Media Division wants to extend the previous ontology, adding facts about music and multimedia. The derived ontology would look like:

```
<ONTOLOGY "music" VERSION="1.5">
    <ONTOLOGY-EXTENDS "root" VERSION="1.0" PREFIX="r"
        URL="http://www.ontology.com/root.html">
    <ONTDEF CATEGORY="Album" ISA="r.Thing">
    <ONTDEF CATEGORY="Image" ISA="r.Thing">
    <ONTDEF RELATION="cover" ARGS="Album Image">
    <ONTDEF RELATION="performer" ARGS="Album r.Person">
    <ONTDEF RELATION="composer" ARGS="Album r.Person">
</ONTOLOGY>
```

This ontology adds albums and images to the list of classifications for SHOE data, declaring them to be "things", as defined in the root ontology. Furthermore, albums may have people as performers and composers, and images as album-covers.

The second part of SHOE consists of tags web authors use to actually mark up their web pages. For example, Bill Clinton might wish to put together a collection of data about himself, as part of his home page...

```
<HTML><HEAD><TITLE>Bill Clinton</TITLE>
    <META HTTP-EQUIV="Instance-Key"
        CONTENT="http://www.whitehouse.gov/bill.html">
    <USE-ONTOLOGY "root" VERSION="1.0" PREFIX="r"
        URL="http://www.ontology.com/root.html">
</HEAD><BODY>
    <CATEGORY "r.Person">
    <RELATION "r.name" 1=me 2="Bill Clinton">
<P> Hi, I'm Bill Clinton. Welcome to my web page...
```

In this example, Bill declares a unique key for himself; this key is based on the URL of his web page. Additionally, he indicates that he will draw on the root ontology to make declarations in this web page, categorizing himself as a person and giving himself a name. It so happens that Bill wants to discuss his brother Roger, who doesn't have a web page of his own. To do this, he can declare Roger as a *subentity* appearing on his home page with the additional code:

```
<INSTANCE "http://www.whitehouse.gov/bill.html#roger">
    <CATEGORY "r.Person">
    <RELATION "r.name" 1=me 2="Roger Clinton">
    <RELATION "r.relation" 1=me
        2="http://www.whitehouse.gov/bill.html">
</INSTANCE>
```

This gives Roger a unique key, and declares some facts about him (including the fact that he is related to Bill).

Web pages aren't restricted to discussing local information; they can also include relations with data entities from other places. If a music company had a record performed by the President but written by his brother, it can describe this information as so:

```
<P> Welcome to the Music Company!
<USE-ONTOLOGY "music" VERSION="1.0" PREFIX="g"
        URL="http://www.ontology.com/music.html">
<INSTANCE "http://www.music-company.com/Bill.html">
    <CATEGORY "g.album">
    <RELATION "g.r.name" 1=me
        2="Bill Clinton: The Saxophone Sessions">
    <RELATION "g.cover" 1=me
        2="http://www.music-company.com/Bill.gif">
    <RELATION "g.performer" 1=me
        2="http://www.whitehouse.gov/bill.html">
    <RELATION "g.composer" 1=me
        2="http://www.whitehouse.gov/bill.html#roger">
</INSTANCE>
```

All this may seem like a lot of work, but in reality annotating a web page like this is easier than one might think. We have developed a Java applet, the *Knowledge Annotator*, which helps a web author graphically edit his pages, modifying the SHOE data without ever having to write a single line of HTML code. Figure 1 shows the Knowledge Annotator in the process of modifying Bill Clinton's

Figure 1. The Knowledge Annotator.

home page.

As this example shows, SHOE has category and relation capabilities common to many database languages. But SHOE provides more than this. It also offers inferences to help cut down on the information web pages must spell out: if Bill and Roger are relatives, and Bill and Hillary are relatives, Roger shouldn't have to list Hillary as a relative; such a thing should be inferable. Ontologies can give inferential rules in the form of simple logical clauses. For example, the inference about relatives would be written as relative(x,z) relative(x,y), relative(y,z), $x \neq z$. In SHOE, the inference is written in an ontology declaration as:

```
<ONTDEF INFERENCE>
    <ONTIF RELATION="relative" 1="x" 2="y">
    <ONTIF RELATION="relative" 1="y" 2="z">
    <ONTIF SPECIAL="notEqual" 1="x" 2="z">
    <ONTTHEN RELATION="relative" 1="x" 2="z">
</ONTDEF>
```

While it provides some knowledge-representation semantics, it's worth noting that SHOE intentionally does not have *all* the capabilities of a modern knowledge-representation language such as KIF [2]. The powerful expressivity found in these languages comes at the cost of high computational complexity, and given the massive size of data on the web, the amount of time to process a query would be unacceptably long. Additionally, SHOE must be robust in the face of web information that may be incorrect, unavailable, inconsistent, or incomplete. Full knowledge-representation expressivity is difficult if not impossible in such a situation. Finally, the semantics of such languages are often very difficult to grasp. SHOE needs to be understandable by the Web population as a whole.

Exposé: An Off-line SHOE Agent

Given access to web pages embedded with this sort of information, it is relatively easy for a web agent to gather the necessary data to answer a query like "Find an album performed by someone and composed by a relative of his." In fact, we have built a SHOE agent, Exposé, that can do exactly this. Exposé uses as its knowledge-base engine PARKA [3], a high-performance knowledge-representation system. The web is a big place, with a lot of potential data. PARKA's horsepower allows Exposé to ask sophisticated queries over huge amounts of highly interconnected data.

Exposé is an "off-line" agent in two parts. The first part searches the web as directed to gather ontologies, adding these ontologies as part of its internal PARKA ontology. The second part roams the web, searching for SHOE data to interpret according to its internal ontology. Exposé uses its existing knowledge-base to interpret and store any interesting new data it finds, and to find new web sites to visit.

Once Exposé has gathered knowledge from the web, it can answer sophisticated queries about this data. After visiting www.music-company.com for example, Exposé interns facts about albums stored there, interpreted according to the music-album ontology. After subsequently visiting www.whitehouse.gov and other places to learn more about various performers, it will have learned about Bill and Roger. At this

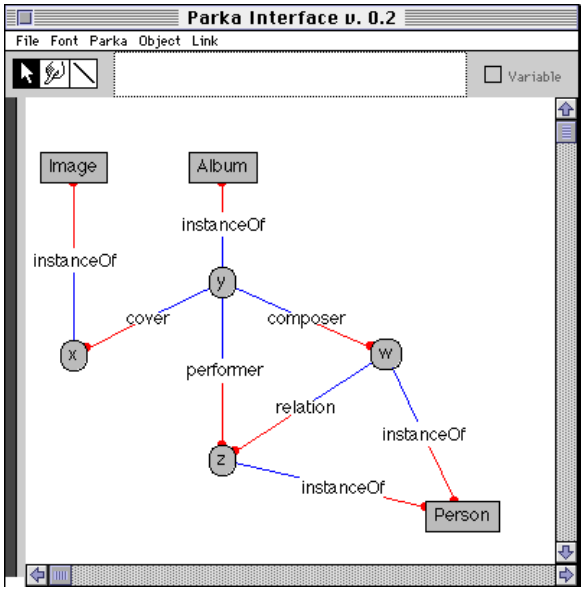

Figure 2. Querying Parka graphically to "Find an album performed by someone and composed by a relative of his".

point, Exposé now knows enough that we can issue the previous query: "Find an album performed by someone and composed by a relative of his." Figure 2 shows this query laid out in PARKA's graphical Java browser. This is the equivalent of asking PARKA the logic request:

Find the URL for X such that

X is an image, Y is an album, and Z and W are people where

composer(Y, W), performer(Y, Z),

relation(W, Z), cover(Y, X).

After the user has submitted this query, Exposé will find the data entity that represents the proper album cover, then fetch the picture of a cover and display it in a web browser, as shown in Figure 3.

The Truth Is Out There

There is a lot of information out there on the Web, and finding exactly what you need is difficult, especially when it concerns the non-textual data found in multimedia. SHOE gives web authors the tools to annotate documents with exact information about their contents and relationships, and in doing so it carefully takes into consideration issues special to the World-Wide Web. The result is a mechanism that permits web agents to do real, useful work helping users gather this information without being hobbled by the need to comprehend web pages like humans do.

As an off-line agent, Exposé nicely demonstrates the power SHOE can offer to searching the Web. But we think that SHOE will really shine when "online" or "guide" agents use SHOE not only as raw data to digest, but also as a guide to help them make better decisions about where to search next. For example, a not-so-bright natural-language web agent in search of pictures of fruit might come across the home page of Apple Computer, Inc., and mistakenly figure Apple for an apple-growing company. This could be a devastating decision, as Apple has a gigantic web server that contains surprisingly few pictures of fruit. With SHOE, the Apple home page could better assist this poor agent by precisely indicating that Apple is a company whose market is computer technology. More importantly, Apple's web page can do more than just ward off prospective fruit-searchers; SHOE data can detail various facets of the company and of its web site. This would be a boon for custom web agents in search of, say, recent press releases or laser printer spec sheets.

Similarly, an "intelligent browser" agent could benefit from SHOE. Imagine browsing images from the vast archives of the National Gallery of Art; as you surf the museum's web pages, the browser displays SHOE data about each image. It would tell you that a particular work of art was done in France in 1787, that it was last sold at auction for $2 million, and that it was done by a female artist. The web surfer indicates various points of interest in these facts; the browser then searches for art tagged with roughly similar information. After seeing this new art, the surfer continues refining his interests. In this way the browser-guide and user work together to zoom in on art of interest much more rapidly than the user can on his own.

We think that these applications are the future direction for the World-Wide Web, and for the internet

in general. But without the ability to gather and understand exact information about multimedia data and documents, making these applications a reality will be difficult. In the future, we may have the technology necessary to read human-oriented multimedia documents. In the mean time, by making the Web less agent-hostile we can take great strides towards that goal today.

Further Reading

More information about SHOE, including published papers, specifications, and examples, can be found at the SHOE home page, http://www.cs.umd.edu/projects/plus/SHOE/

References

1. Doorenbos, R. B., O. Etzioni, and D. S. Weld. 1997. A Scalable Comparison-Shopping Agent for the World-Wide Web. In *Proceedings of the First International Conference on Autonomous Agents (AA97).* W. L. Johnson, Editor. New York: Association for Computing Machinery. 39–48.

2. Genesereth, M. R., and R. E. Fikes, Eds. 1992. *Knowledge Interchange Format, Version 3.0 Reference Manual*. Technical Report Logic-92-1. Computer Science Department, Stanford University. URL: http://www-ksl.stanford.edu/knowledge-sharing/papers/kif.ps

3. Evett, M.P., W.A. Andersen, and J.A. Hendler. 1993. Providing Computational Effective Knowledge Representation via Massive Parallelism. In *Parallel Processing for Artificial Intelligence*. L. Kanal, V. Kumar, H. Kitano, and C. Suttner, Eds. Amsterdam: Elsevier Science Publishers. URL: http://www.cs.umd.edu/projects/plus/Parka/parka-kanal.ps See also: http://www.cs.umd.edu/projects/plus/Parka/

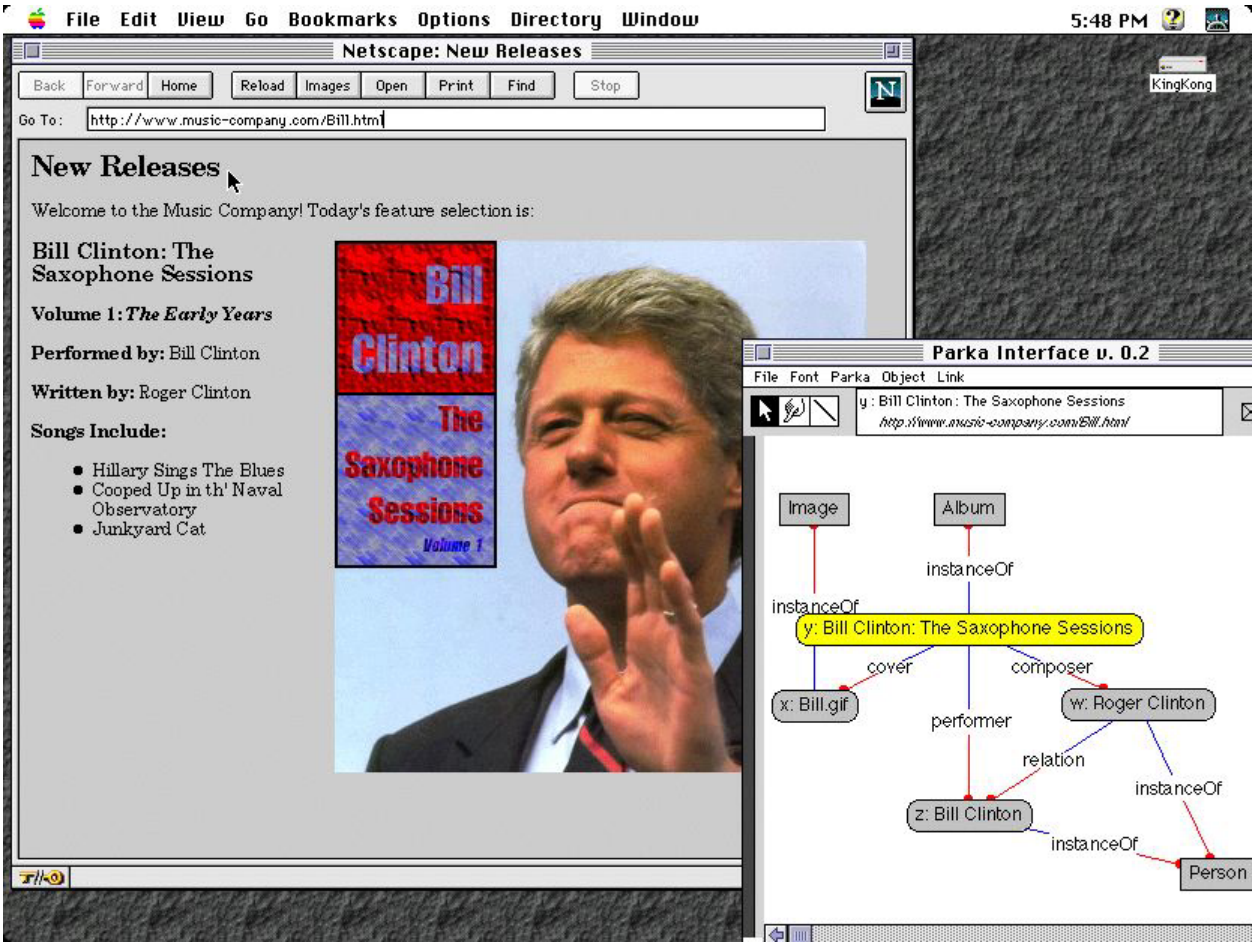

Figure 3. Displaying the results of the query issued in Figure 2.