Appears in 'omm. of ACM, July '94

A Softbot-Based Interface to the Internet

Oren Etzioni and **Daniel Weld**
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195
{etzioni, weld}@cs.washington.edu

The Internet Softbot (software robot) is a fully-implemented AI agent developed at the University of Washington (Etzioni, Lesh, & Segal 1993). The softbot uses a UNIX shell and the World-Wide Web to interact with a wide range of internet resources. The softbot's effectors include `ftp`, `telnet`, `mail`, and numerous file manipulation commands. Its sensors include internet facilities such as `archie`, `gopher`, `netfind`, and many more. The softbot is designed to incorporate new facilities into its repertoire as they become available.

The softbot's "added value" is three-fold. First, it provides an integrated and expressive interface to the internet. Second, the softbot dynamically chooses which facilities to invoke, and in what sequence. For example, the softbot might use `netfind` to determine David McAllester's e-mail address. Since it knows that `netfind` requires a person's institution as input, the softbot would first search bibliographic databases for a technical report by McAllester which would reveal his institution, and then feed that information to `netfind`. Third, the softbot fluidly backtracks from one facility to another based on information collected at run time. As a result, the softbot's behavior changes in response to transient system conditions (*e.g.*, the UUCP gateway is down). In this article, we focus on the ideas underlying the softbot-based interface.

A Softbot-Based Interface

By acting as an intelligent personal assistant, the softbot supports a qualitatively different kind of human-computer interface. A person can make a high level request, and the softbot uses search, inference, and knowledge to determine how to satisfy the request. Furthermore, the softbot is able to tolerate and recover from ambiguity, omissions, and errors in human requests.

At its core, the softbot can handle goals specified in an expressive subset of first order logic. In particular, conjunction, disjunction, negation, and universal quantification can be composed to specify goals for the softbot. Since naive users are uncomfortable with logical notation, we have implemented a menu of request forms (Figure 1) which can be sent to the softbot via e-mail, from mosaic, or through an X-windows graphical user interface. A filled-in form is automatically translated into a softbot goal (Figure 2). In principle, any dialog modality (*e.g.*, natural language, speech, and pen interfaces) could be used to communicate with the softbot; we need only add a module to translate to and from the softbot's logical language.

Interface Design Principles

In designing our interface, we have de-emphasized the "look and feel" of the interface, and focused on how to leverage the softbot's AI capabilities to increase the interface's expressive power and flexibility. Specifically, our softbot-based interface embodies the following ideas:

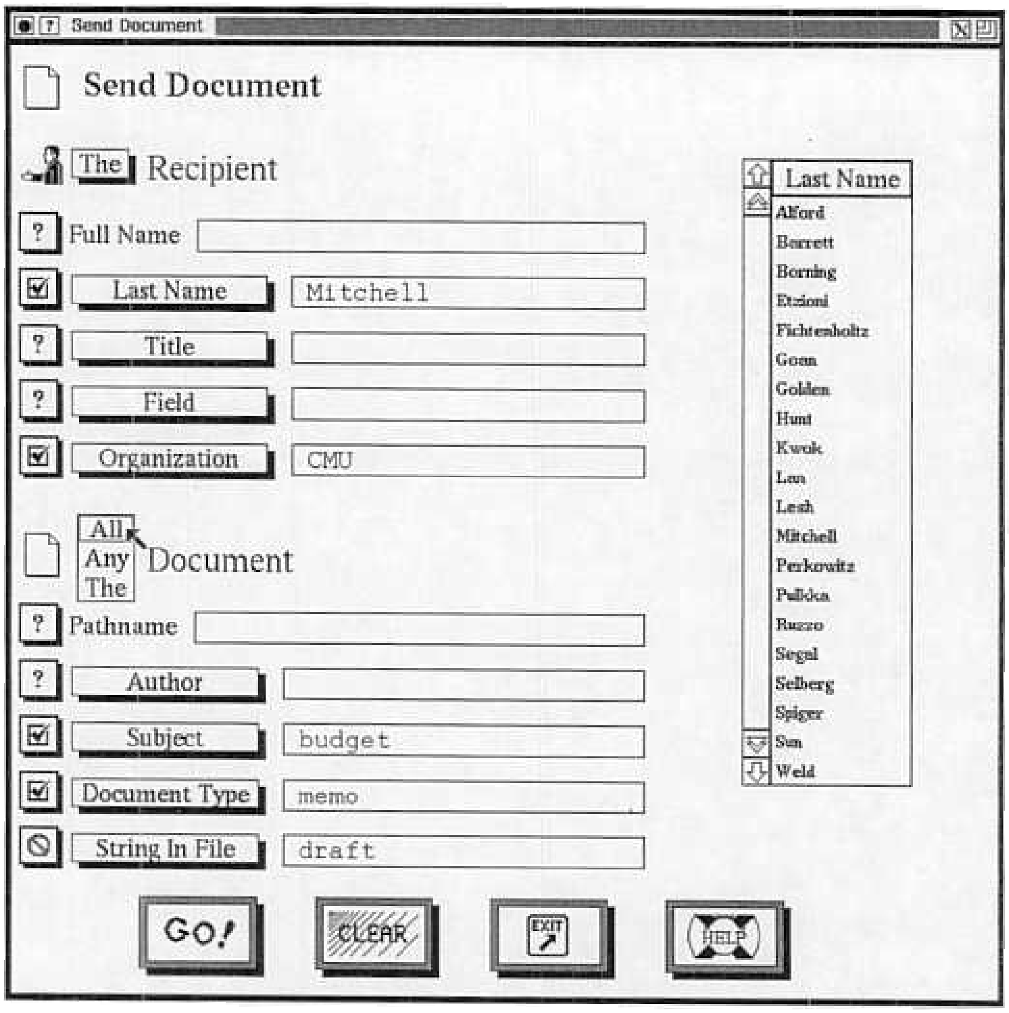
1. **Goal oriented:** a request indicates *what* the human wants. The softbot is responsible for deciding *how* and *when* to satisfy the request.
2. **Charitable:** a request is not a complete and correct specification of the human's goal, but a *clue* or a *hint* that the softbot attempts to decipher and then satisfy.[1]
3. **Balanced:** the softbot has to balance the cost of finding information on its own, against the nuisance value of pestering the human with questions.
4. **Integrated:** the softbot provides a single, expressive, and uniform interface to a wide variety of internet services and utilities.

The following scenario illustrates these ideas.

The Softbot in Action

Consider the task "Send the budget memos to Mitchell at CMU." A human assistant would handle this request with ease, but most existing software agents would not. Even if one solves (or circumvents) the problem of natural-language understanding, the agent still has to figure out:

[1]When the request is unethical or dangerous, the most appropriate response may be to alter or even refuse the request (Wilensky *et al.* 1988; Weld & Etzioni 1994).


Send Document

Send Document

The Recipient

? Full Name

✓ Last Name: Mitchell

? Title

? Field

✓ Organization: CMU

All / Any / The Document

? Pathname

? Author

✓ Subject: budget

✓ Document Type: memo

⊘ String In File: draft

Last Name
Alford
Barrett
Borning
Etzioni
Fichtenholtz
Goan
Golden
Hunt
Kwok
Lau
Lesh
Mitchell
Perkowitz
Pulkka
Ruzzo
Segal
Selberg
Spiger
Sun
Weld

Figure 1: The request form for sending a document. Note how the graphical interface supports the logical power of universal quantification (*all* documents) and negation ("draft" *not* present as a string in the file). In general, users need only provide a partial specification of the desired goal. The softbot disambiguates the request and plans how to achieve it, subgoaling and backtracking as required.

- Which Mitchell was intended?
- Which document should I send? (and where is it located?)
- How do I send the memos? (e-mail, fax, remote printing, etc.)
- What if the memos are confidential?
- What if Mitchell is out of town?

As this simple example illustrates, even mundane human requests are incompletely specified, potentially ambiguous, or even impossible to satisfy (what if there is no Mitchell at CMU?).

The softbot's first task is *disambiguation.* It has to decide what "objects" the request is referring to: for instance, who is the intended recipient of the memos? The request suggests that the memos ought to go to a person named Mitchell at CMU, but there may be several people at CMU that share the same last name. The softbot could adopt the policy of asking the human to specify the recipient more clearly whenever his full name is not provided, but this is inappropriate. A last name *could* potentially pick out a unique individual. For example, suppose the last name provided is "Satyanarayanan." In this case, the softbot's request for clarification would be gratuitous and annoying. In general, any description, however tenuous, might pick out a unique individual. Before asking questions, the softbot ought to *check* whether the given description is ambiguous.

The softbot could consult its knowledge base to see how many Mitchells it "knows" at CMU, but suppose that the softbot is familiar with only one, can it be sure that it is familiar with *all* the Mitchells at CMU?

```
(forall (?d :in files)
    (if (and (file.type ?d memo.document)
             (subject.of.document ?d "budget")
             (not (string.in.file "draft" ?d)))
        (delivered.to ?d ?obj341)))
```

Figure 2: The form shown in Figure 1 is automatically translated into an internal representation like the one shown here. During this process, the softbot executes actions as needed in order to find a unique person object (*i.e.*, `?obj341`) with last name "Mitchell" and institution "CMU". Next the softbot planner will determine how to achieve the delivery goal.

Since its knowledge of people on the internet is bound to be radically incomplete, the softbot cannot afford to make the *closed world assumption* made by many AI and database systems (Reiter 1978). Thus, it cannot automatically conclude that there *is* only one Mitchell at CMU from the fact that it is *familiar* with only one. Fortunately, it is easy to find all the Mitchells at CMU (by executing `finger mitchell@cmu.edu`). The softbot executes this command, records who are the various Mitchells at CMU, and (if necessary) prompts the human with a request to choose the intended one. The softbot also records, for future reference, that it is now familiar with all the Mitchells at CMU. Despite its incomplete knowledge, the softbot can recognize when it has complete information on a particular topic or locale (Etzioni, Golden, & Weld 1994).

To resolve ambiguity, the softbot could try to infer who the intended Mitchell is, based on the documents being sent and the context of the request (*e.g.*, did the human just receive an e-mail message from some Mitchell at CMU?). While plausible inference of this sort can be encoded within our softbot framework, our implementation is not that sophisticated, yet. Currently, the softbot attempts to find all individuals or objects on the internet matching a given description. If there is a single resource that provides this information, the softbot will access it (*e.g.*, `finger` in the above example). Otherwise, the softbot will form a plan to seek out matching individuals. If the description is not constrained appropriately, executing such a plan can be very expensive. For instance, suppose the human omits Mitchell's location in the above request. The softbot would be "tempted" to search the entire internet looking for Mitchells. However, the balance principle implies that the softbot would be better off asking the human to further constrain Mitchell's description. Thus, before disambiguating, the softbot estimates the cost of its disambiguation plan. When the cost is high, the softbot prompts the human for more information: "I am not sure which Mitchell you mean, can you tell me Mitchell's workplace, city, or field of interest?"

Once the appropriate Mitchell (and memos) have been determined, the softbot's second task is to actually send the memos to Mitchell. The softbot may decide to e-mail the memos, but first it has to find Mitchell's e-mail address, and reason about document format. For example, if a document contains figures, then sending the postscript version is more appropriate than sending the LaTeX source. Furthermore, if Mitchell is out of town, or if the memos are confidential, the softbot has to ensure that the memos reach their recipient in a timely and secure manner.

In general, after the softbot has figured out what the human wants, it considers how to satisfy the human request. The softbot solves this problem by invoking an automatic planning algorithm. We describe our softbot's planning capabilities in the next section.

Softbot Planning

To construct an integrated and goal-oriented interface, we use AI planning techniques. The softbot planner takes a logical expression describing the user's goal as input. After searching a library of *action schemata* describing available information sources, databases, utilities and software commands, the planner *dynamically* generates a sequence of actions that achieve the goal, backtracking and subgoaling as necessary.

Unlike standard programs and scripts which are committed to a rigid control flow determined *a priori* by a programmer, the softbot's planner automatically synthesizes and executes plans to achieve the goals which were input. This avoids the problematic task of writing programs that anticipate all possible changes in system environment, network status, and error conditions. In short, a softbot is worth a thousand shell scripts.

The softbot's planner accepts an expressive goal language, enabling the softbot to accept goals containing complex combinations of conjunction, disjunction, negation, and nested universal and existential quantification. Furthermore, the softbot's use of planning yields an integrated interface — users can write expressive goals, even when dealing with services that don't support them directly. Consider the task "Get all of Ginsberg's technical reports that aren't already stored locally." Through planning, the softbot can use `ftp` to handle this request, even though the `ftp` utility doesn't know which files are local, and does not handle this combination of universal quantification and negation. The softbot will determine which of Ginsberg's reports are not stored locally, and will issue `ftp` commands to obtain them.

The softbot planner is implemented as a search process over partially specified action sequences called *plans*. The planner is able to decompose complex goal expressions into their constituents and solve them with divide and conquer techniques. Interactions between subgoals are automatically detected and resolved. Space precludes comprehensive discussion of the algorithm (see (Weld 1994; Golden, Etzioni, & Weld 1994)); however, we note that modern planning algorithms are provably:

- Complete: if a plan exists, the planner will find it, and
- Sound: if the planner outputs a plan, that plan is guaranteed to achieve its goal (modulo certain explicit assumptions).

While these formal guarantees do not ensure that the planner is efficient, we have not found efficiency to be a problem in practice. The softbot planner accepts control heuristics, specified in a high-level, declarative language, which constrain the planner's search by instructing it to prefer certain options over others, avoid blind alleys, *etc.*. These heuristics can be hand-coded or generated automatically via machine learning techniques (Etzioni 1993a; Minton 1990).

We now consider the benefits derived from the softbot's use of a modern planning algorithm.

Subgoaling

If the softbot cannot satisfy its goal directly, it will automatically *subgoal* on an indirect way of satisfying the goals. For example, if looking up the phone number of a graduate student fails, the softbot will subgoal on identifying the student's office-mates and finding their phone number. The softbot relies on an inference rule which states that office-mates share the same telephone.

Resource Integration

The planner relies on a logical model of the available internet resources, which answers two questions: how can the softbot invoke or access the resource, and what is the effect of doing so? This sort of "resource model" can be viewed as a generalization of a Prolog inference rule to allow for multiple effects, nested universal and existential quantification, and state change. The precise syntax and semantics of our representation language are described in (Etzioni *et al.* 1992; Etzioni, Golden, & Weld 1994).

This declarative representation enables the softbot to integrate multiple, independent internet facilities in service of its goal. For instance, as mentioned in the introduction, the softbot's model of `netfind` (Figure 3) tells it that it has to know a person's institution (or city) before accessing the `netfind` facility. Thus when necessary, the softbot subgoals on finding this information by invoking different facilities (*e.g.*, it might access the `INSPEC` database or `grep` through local bibliographic databases). Furthermore, the softbot is poised to leverage new internet resources. When a new facility becomes available, we need only write the appropriate logical models and search control rules to update the softbot. We are also investigating the use of learning techniques to help automate this task.

Incomplete Specification → Search

The softbot accepts incompletely specified goals, and searches for missing information whenever possible. For example, if asked to print a file on "any free printer in the building," the softbot will find the printer list in a database, and will check the status of each printer until it finds one that is free. Similarly, if a human asks to be notified when Etzioni logs in to some machine at the University of Washington, the softbot will search for machines where Etzioni has an account and will monitor these machines until he appears. In general, the softbot's goal language allows the human to state three kinds of goals:

- Ground goals: notify me when Etzioni logs in to the machine called June.cs.washington.edu.
- Existentially quantified goals: notify me when Etzioni logs in to *some* machine.
- Constrained goals: notify me when Etzioni logs in to some computer-science machine at the University of Washington.

Empirically, we have found that constrained goals strike a useful balance between burdening the user with endless questions, and sending the softbot on a massive search of the internet.

The use of a modern algorithm for planning with incomplete information is one of the most distinctive features of our softbot. In contrast, much of the current work on intelligent software agents focuses on task-specific "bots" for visitor scheduling (Kautz *et al.* 1994), meeting scheduling (Dent *et al.* 1992), e-mail filtering (Maes & Kozierok 1993), white-page services (Droms 1990), etc. While each of these agents has its strengths, none share the benefits of the planning approach including a highly expressive goal language, automatic backtracking and subgoaling, and more.

Softbot Safety

We've argued that in order to provide an integrated, goal-oriented interface, one needs powerful tools such as the softbot. However, the softbot also requires safety features. Of course, this is true of any software tool (*e.g.*, the Macintosh refuses to reformat one's startup disk), but safety is more important for more powerful tools. Just as in the carpentry domain (where a table saw can slice through a tendon), internet power tools, like the softbot, are capable of inflicting damage when used indiscriminately.

The greatest danger lies in the softbot's very ability to plan how to achieve the user's goal. Note that using a sound planner (*i.e.*, one that only generates plans which are guaranteed to achieve the goal) is not sufficient, since there may be *many* such plans with different side effects. Consider the task "Reduce disk utilization below 90%." If the softbot succeeded by deleting irreplaceable LaTeX files without backing them up to tape, then users might prefer less "powerful" tools!

```
Name:      (netfind ?person)
Preconds:                                        Postconds:
   (current.shell csh)                              (userid ?person !userid)
   (isa netfind.server ?server)                     (person.machine ?person !machine)
   (firstname ?person ?firstname)
   (lastname ?person ?lastname)
   (or (person.city ?person ?keyword)
        (person.institution ?person ?keyword))
```

Figure 3: An action schema encoding the **netfind** utility; the softbot planner uses the schemata to determine when an internet service could help achieve a user goal. The disjunctive precondition encodes the fact that **netfind** requires the person's institution or city to constrain its search. The softbot subgoals on obtaining that information from a distinct information source. See (Etzioni *et al.* 1992; Golden, Etzioni, & Weld 1994) for the precise syntax and semantics of our representation language.

We believe that the softbot's safety mechanism should ensure the following qualities:

- **Safe:** the softbot should refuse to make destructive changes to the world.
- **Tidy:** the softbot should restore the world as close as possible to its original state (*i.e.*, recompress files after searches *etc.*)
- **Thrifty:** the softbot should limit its use of valuable resources.
- **Vigilant:** the softbot should block human actions that have unintended consequences.

See (Weld & Etzioni 1994) for a formalization of some of these ideas in a manner that supports computationally tractable implementation. Since the ideas reported therein are preliminary and as yet unimplemented, we acknowledge that softbot safety is an area that deserves substantially more investigation.

Conclusion

Software environments such as the internet are attractive testbeds for AI research (Etzioni 1993b). Softbots circumvent many thorny issues that are inescapable in physical environments. Furthermore, the cost, effort, and expertise necessary to develop and experiment with software artifacts are relatively low. Yet, in contrast to simulated worlds, software environments are readily available, economically important, and *real*. In the past three years, the focus of the Internet Softbots project has been on the AI problems of designing and building an agent capable of effectively exploring the internet. See (Etzioni *et al.* 1992; Etzioni, Golden, & Weld 1994; Golden, Etzioni, & Weld 1994) for a sample of technical AI results achieved in this context.

We are now leveraging the softbot's AI capabilities to develop an expressive, goal oriented, and charitable interface to the internet. In contrast to a loosely structured *browser* such as Mosaic, our long-term objective is to enable naive users to effectively locate, monitor, and transmit information across the net.

Acknowledgments

We would like to thank our co-softboticists Tony Barrett, Greg Fichtenholtz, Terrance Goan, Keith Golden, Cody Kwok, Aaron Pulkka, Mike Perkowitz, Richard Segal, Ying Sun, and Rob Spiger. We thank Alan Borning and Steve Hanks for helpful comments. This research was funded in part by Office of Naval Research Grant 92-J-1946, by a grant from the University of Washington Royalty Research Fund, and by National Science Foundation Grants IRI-9211045 and IRI-9357772.

Authors

Oren Etzioni received his bachelor's degree in computer science from Harvard University in June 1986, and his Ph.D. from Carnegie Mellon University in January 1991. He joined the University of Washington as Assistant Professor of Computer Science and Engineering in February 1991. In the fall of 1991, he launched the Internet Softbots project. In 1993, Etzioni received an NSF Young Investigator Award. Etzioni is the president of the AI Access Foundation, a nonprofit corporation devoted to the electronic dissemination of scientific results in AI. He is on the editorial and advisory boards of the Journal of AI Research, and has served on the program committees for AAAI and IJCAI. Most recently, he chaired the AAAI spring symposium on software agents.

Daniel Weld received bachelor's degrees in both Computer Science and in Biochemistry at Yale University in 1982. He landed a Ph.D. from the MIT Artificial Intelligence Lab in 1988 and immediately joined the Department of Computer Science and Engineering at the University of Washington where he is now Associate Professor. Weld received a Presidential Young Investigator's award in 1989 and an ONR Young Investigator's award in 1990. He is associate editor for the Journal of AI Research, was guest editor for Computational Intelligence, and has served on the program committee for the National Conference on Artificial Intelligence. Weld has published fifty refereed technical papers and has two books to his name.

References

Dent, L., Boticario, J., McDermott, J., Mitchell, T., and Zabowski, D. 1992. A personal learning apprentice. In *Proc. 10th Nat. Conf. on A.I.*, 96–103.

Droms, R. 1990. Access to Heterogeneous Directory Services. In *IEEE INFOCOM '90*, 1054–1061.

Etzioni, O., Hanks, S., Weld, D., Draper, D., Lesh, N., and Williamson, M. 1992. An Approach to Planning with Incomplete Information. In *Proc. 3rd Int. Conf. on Principles of Knowledge Representation and Reasoning.* Available via FTP from pub/ai/ at cs.washington.edu.

Etzioni, O., Golden, K., and Weld, D. 1994. Tractable closed-world reasoning with updates. In *Proc. 4th Int. Conf. on Principles of Knowledge Representation and Reasoning.*

Etzioni, O., Lesh, N., and Segal, R. 1993. Building softbots for UNIX (preliminary report). Technical Report 93-09-01, University of Washington. Available via anonymous FTP from pub/etzioni/softbots/ at cs.washington.edu.

Etzioni, O. 1993a. Acquiring search-control knowledge via static analysis. *Artificial Intelligence* 62(2):255–302.

Etzioni, O. 1993b. Intelligence without robots (a reply to brooks). *AI Magazine* 14(4). Available via anonymous FTP from pub/etzioni/softbots/ at cs.washington.edu.

Ginsberg, M., ed. 1987. *Readings in Nonmonotonic Reasoning.* San Mateo, CA: Morgan Kaufmann.

Golden, K., Etzioni, O., and Weld, D. 1994. Omnipotence without omniscience: Sensor management in planning. In *Proc. 12th Nat. Conf. on A.I.*

Kautz, H., Selman, B., Coen, M., Ketchpel, S., and Ramming, C. 1994. An experiment in the design of software agents. In *Proc. 12th Nat. Conf. on A.I.*

Maes, P., and Kozierok, R. 1993. Learning interface agents. In *Proceedings of AAAI-93.*

Minton, S. 1990. Quantitative results concerning the utility of explanation-based learning. *Artificial Intelligence* 42(2-3).

Reiter, R. 1978. On closed world databases. In Gallaire, H., and Minker, J., eds., *Logic and Data Bases.* Plenum Press. 55–76. Reprinted in (Ginsberg 1987).

Weld, D., and Etzioni, O. 1994. The first law of robotics (a call to arms). In *Proc. 12th Nat. Conf. on A.I.* Available via FTP from pub/ai/ at cs.washington.edu.

Weld, D. 1994. An introduction to least-commitment planning. *AI Magazine.* Available via FTP from pub/ai/ at cs.washington.edu.

Wilensky, R., Chin, D., Luria, M., Martin, J., Mayfield, J., and Wu, D. 1988. The Berkeley UNIX Consultant project. *Computational Linguistics* 14(4):35–84.