Do I Care? -- Tell Me What's Changed on the Web

Brian Starr **Mark S. Ackerman** **Michael Pazzani**
Information and Computer Science
University of California, Irvine
{bstarr, ackerman, pazzani}@ics.uci.edu
http://www.ics.uci.edu/CORPS/ackerman.html

Abstract

We describe the Do-I-Care agent, which uses machine learning to detect "interesting" changes to Web pages previously found to be relevant. Because this agent focuses on changes to known pages rather than discovering new pages, we increase the likelihood that the information found will be interesting. The agent's accuracy in finding interesting changes and in learning is improved by exploiting regularities in how pages are changed. Additionally, these agents can be used collaboratively by cascading them and by propagating interesting findings to other users' agents.

Introduction

After one has discovered Web resources, a problem remains: How do you know when to *re*visit those resources for new material? Consider some common post-discovery situations:

- ❑ You know the page where a colleague has his publications. You would like to obtain *interesting* new papers as they are posted.
- ❑ You have obtained the page for a low cost clothes merchandiser. You would like to know when there is a sale on jeans or Hawaiian shirts.

These are examples of change events that occur somewhat unpredictably and offer no explicit notification. Today, interested parties must occasionally check by hand. This can be burdensome and tedious, and one may forget to do it on a timely basis. While there are a number of agents that will notice that *some* modification occurred, we would like an agent that noticed when *interesting* changes occurred. The user may not want to know when the author changed the font, some minor text, or added insignificant material.

This paper provides an overview of an agent, called Do-I-Care, that determines when interesting changes have occurred on previously discovered pages. It works by applying machine learning to regularities, including social norms of use, on the Web.

Locating information on the Web

There has been considerable effort devoted to the resource discovery problem (Schwartz et al. 1992). We note two classes of systems that facilitate information and resource location, and discuss some of their limitations.

Active browsers

Active browser systems, such as Web Watcher (Armstrong et al. 1995), treat the Web as a graph that can be searched using evaluation functions and heuristics derived from a model of users' goals and interests. While users browse the Web, Web Watcher offers navigational suggestions that direct them along paths that maximize these functions. User models are obtained directly from users while they are browsing, and require their active participation. If the user model is accurate, the advice given will presumably guide users to useful Web locations.

Because searching is in real time, given the size and complexity of the Web, agent suggestions can approximate hill climbing. Thus, the choice of starting points is particularly important, and some form of preliminary catalog search may be necessary. More seriously, active search requires human participation, time, and attention, which are generally scarce resources.

A variation described by Balabanovic and Shoham (1995) is an agent that wanders the Web independently from the user. It reports interesting pages to the user, who can then rate these suggestions. Therefore, the system uses relevance feedback, yet only occasional human participation is required. However, this agent is not optimized to detect changes to existing pages.

There are also a number of agents that monitor Web pages for *any* modification, such as URL-minder. However, many of the changes detected may be irrelevant.

Index creation and lookup

Index creation and lookup systems, such as ALIWEB (Koster 1994), utilize an information retrieval metaphor, where the Web is treated as a vast database of information to be cataloged and indexed.

These systems make the Web accessible to users who need not know its organization. However, users must articulate their information goals in well-formed queries. Also, due to the Web's rapid, uncoordinated growth, precision is increasingly important (Pinkerton 1994). Finally, users interested in new information must resubmit their queries if notification is not supported.

The SHADE and COINS matchmaker systems (Kuokka and Harada 1995) utilizes a knowledge sharing standard such as KQML to match user requests with information provider's "advertisements", and can notify users when it receives updates from providers about interesting changes to their knowledge-base. However, this system must rely on provider honesty, since there are no technical checks against suppliers that "cheat" (Brown et al. 1995).

The Do-I-Care agent

We propose a third type of system, optimized for reoccurring information needs. Do-I-Care was designed to periodically visit Web pages and detect interesting changes. We use machine learning to identify by example what changes are interesting, and how often they occur.

System architecture

The Do-I-Care agent has five major functions. It must:

- Periodically visit a user-defined list of target pages.
- Identify any changes since the last time it visited.
- Extract discriminatory attributes and decide if the changes were interesting.
- Notify the user if the change was interesting.
- Accept relevance feedback on the interestingness of the change.

Exploiting domain specificity

This problem has a number of unique features that we exploit to improve precision (percentage of retrieved items that are "interesting") and recall (the percentage of potentially interesting items that are actually retrieved). Precision, recall, and relevance feedback profit from improvements in attribute collection made possible by domain knowledge and simplifying assumptions. First, we assume that users already have lists of Web pages that will have interesting information, and that they feel these lists are adequate for their needs. By limiting our search to a small subset of Web pages specifically selected because they are likely to contain interesting information, precision is greatly improved. The increased precision allows us to increase recall without penalty.

Second, once a Web page stabilizes, authors seldom radically change them. Change is usually incremental, and new things are usually added in ways that are consistent with what has been done in the past. This is true because of the additional work involved in radical alterations, and because authors tend to follow styles consistent with the other pages of similar content and purpose. In a shared environment like the Web, forms and meanings tend to become socially constructed (Berger and Luckmann 1967). Therefore, we assume that Web pages generally change incrementally. We represent change as document segments that have been added or altered since the last time the page was visited. Deleted segments are ignored. These changes are evaluated, and those found to be relevant are reported to the user. By accepting or rejecting these suggestions, the user refines the types of changes that are interesting.

However, different pages may be interesting for different reasons. For example, the "interesting" attributes in pages with links to new papers may be different than for pages with sale items. Therefore, users may create multiple agents, each one trained to retrieve differing types of changes. These agents may have different user models and notification requirements.

Identifying interesting changes

Since the user provides feedback on whether changes are interesting and we are concentrating on additions, we can view the process of determining whether a change is interesting as a classification task, and we can view the process of learning a user profile as a supervised learning task.

We are currently attempting to use machine learning for this task. We represent each addition by a relatively small number of features. First, we find words that are informative, by computing the mutual information (e.g., Quinlan 1986) between the presence of a word in the document and the relevance feedback. In addition, we use attributes about the changed segment, such as its size. Once the examples are represented as feature vectors, any of a number of standard learning algorithms might be used to determine whether a change is interesting. We are experimenting with a simple Bayesian classifier (Duda and Hart 1973). Informally, we have found this method to be 80-90% accurate after 5-10 training examples.

Notifying the user

Agents are capable of notifying users about interesting changes by e-mail. E-mail notifications list the agent's name and Web page, as well as a link the to page that changed. The remainder of the message contains those attributes values that caused the segment to be considered interesting, and optionally the full text of the change.

All changes are also listed in the agent's associated Web page, which users can browse at their leisure. This Web page is also used for relevance feedback.

Training and configuring

As noted in the previous section, each agent has an associated Web page to report results back to the user. This Web page also serves as the user interface to the Do-I-Care agent. The agent's Web page contains a link to a form allowing the user to edit the agent's name, description, notification e-mail addresses, and pages to monitor for changes. The attribute values that define an interesting change are also listed, but are not editable. Within the body of the page, every change is listed, along with the segment change information described previously.

Every listed change has two associated menus. One menu selects whether to display the full text of a change or a link. The second allows the user to provide relevance feedback, and has three choices: I do care, I do not care, and unrated. When new change is discovered, the associated feedback menu's value is unrated, and the full text is displayed. When the user indicates his preferences via the feedback menu, the agent reinforces or de-emphasizes the attributes it used to evaluate the change. The full text of "rated" changes are not displayed by default, since the user has already seen them.

Scheduling visits

For each target page, the agent keeps track of when the page was last visited, how often the page should be visited, when the next visit will be, and when the last change was detected for that page. The last attribute allows the agent to adjust the visit periodicity based on past activity; there is no reason to visit a page once a day if it hardly changes.

Collaborative use

Although this paper has described using the Do-I-Care agent for individual use, the system was also designed to be used collaboratively. Because all communication takes place through a list of e-mail addresses and a Web page, any number of users can share the same agent. Simple Unix and WWW server access control mechanisms can be used to adjust who can view and modify an agent's activity and configuration. Change notifications can also be sent to distribution lists and news groups. Finally, an agent's Web page may be included other agents' list of pages, facilitating the construction of group or organizational "what's new and interesting" Web pages.

Our future plans include exploring the collaborative aspects of this agent. We feel that it provides a low cost method to facilitate group or organizational sharing of information by capturing and exploiting naturally occurring behavior.

Summary

Do-I-Care attempts to solve the post-discovery problem: when should the user revisit a known site for interesting new information? Doing this by hand is too laborious, and existing agents do not address this problem.

To address this issue, we have attempted to use machine learning and other techniques to discover interesting changes. Because the user has already identified relevant locations, precision is improved. Regularities in how Web pages are usually changed improves attribute extraction.

The Do-I-Care system represents an interesting variation on previous information location systems. Unlike most existing systems that focus on answering immediate queries, Do-I-Care attempts to satisfy the user's ongoing need to remain informed and up-to-date about those parts of a changing world that most interest him.

References

Armstrong, Robert, Dayne Freitag, Thorsten Joachims, and Tom Mitchell. 1995. WebWatcher: A Learning Apprentice for the World Wide Web. *Proceedings of the AAAI Spring Symposium Series on Information Gathering from Heterogeneous, Distributed Environments*: 6-12

Balabanovic, Marko, and Yoav Shoham. 1995. Learning Information Retrieval Agents: Experiments with Automated Web Browsing. *Proceedings of the AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments*: 13-18

Berger, Peter L., Thomas Luckmann. 1967. The social construction of reality; a treatise in the sociology of knowledge, New York : Doubleday.

Brown, Carol, Les Gasser, Daniel E. O'Leary, and Alan Sangster. 1995. AI on the WWW: Supply and Demand Agents. *IEEE Expert* : 50-55.

Duda, R., and P. Hart. 1973. *Pattern classification and scene analysis*. New York: John Wiley and Sons.

Koster, Martijn. 1994. ALIWEB - Archie-Like Indexing in the WEB. *Computer Networks and ISDN Systems* 27(2): 175-182.

Kuokka, Daniel, and Larry Harada. 1995. Supporting Information Retrieval via Matchmaking. *Proceedings of the AAAI Spring Symposium Series on Information Gathering from Heterogeneous, Distributed Environments*: 111-115.

Pinkerton, Brian. 1994. Finding What People Want: Experiences with the WebCrawler. *Proceedings of the Second International WWW Conference.*, http://info.webcrawler.com/bp/WWW94.html

Quinlan, J. R. 1986. Induction of decision trees. *Machine Learning*, 1: 81-106.

Salton, Gerard, and Michael J. McGill. 1983. *Introduction to Modern Information Retrieval.* New York: McGraw-Hill.

Schwartz, Miachel F., Alan Emtage, Brewster Kahle, and B. Clifford Neuman. 1992. A Comparison of Internet Resource Discovery Approaches. *Computing Systems* 5(4): 461-493.