Is Agent-based Online Search Feasible?

Filippo Menczer
Management Sciences Department
University of Iowa
Iowa City, IA 52245, USA
filippo-menczer@uiowa.edu

Abstract

The scalability limitations of the current state of the art in Web search technology lead us to explore alternative approaches to centralized indexing, such as agent-based *online* search. The possible advantages of searching online, or agent-based browsing, are often down-played in the face of the apparently unquestionable loss of efficiency. This paper is an attempt to reach a more balanced view in which both the obvious and hidden costs of the two approaches are considered. The two approaches can then be compared more fairly, and possible complementarities and synergies are explored.

Model overview

This paper discusses an agent-based approach to building scalable information searching algorithms. For systems designed to let users locate relevant information in highly distributed and decentralized databases, such as the Web, we argue that *scalability* is one of the main limitations of the current state of the art.

The complexities emerging in networked information environments (decentralization, noise, heterogeneity, and dynamics) are not unlike those faced by ecologies of organisms adapting in natural environments. The capabilities of such *natural* agents — local adaptation, internalization of environmental signals, distributed control, integration of externally driven and endogenous behaviors, etc. — represent desirable goals for the next generation of *artificial* agents: autonomous, intelligent, distributed, and adaptive. These considerations, along the lines of the artificial life approach, inspired us to base our model upon the metaphor of an *ecology* of agents.

In this sense, the *multi-agent* system is not composed of a few agents with distinct and clearly defined functions, but rather by a (possibly very) large number of agents collectively trying to satisfy the user request. The number of agents is determined by the environment, in turn shaped by the search task. This does not mean that all agents responding to a specific search are identical; each will adapt to both the local context set by the environment and the global context set by the user.

Cooperation results from the indirect interaction among agents, mediated by the environment. If there are sufficient resources to sustain multiple agents in a given information neighborhood, then new agents will spawn and collaborate with the existing ones. If resources are scarce, on the contrary, the agents will compete and some of them will be eliminated.

The ecological metaphor thus induces a rational use of the information resource chain. Computational resources (CPU time) and network resources (bandwidth) are allocated in proportion to the recently perceived success of an agent, estimated from the relevance of the consumed information resources. Ideally, each agent would browse the Web neighborhood in which it is situated like its human master would, given her finite resources — time and attention.

The approach discussed in this paper assumes that, in exchange for improved bandwidth or payments, server hosts may allow trusted mobile agents to execute and possibly even spawn new agents using their CPU, memory, and disk storage in a controlled operating environment. If this assumption holds, agents can take advantage of the distributed nature of our algorithm. They can execute in a parallel, asynchronous fashion, resulting in a great potential speedup. Security and distributed systems issues are central to the implementation of such mobile agents. If the assumption fails, however, the agents can still execute in a client-based fashion. The algorithm becomes essentially sequential (although possibly multi-threaded), and thus simpler to implement. The decrease in performance can be partially offset by the use of a central cache.

State of the art and agent-based alternatives

Search engines

The model behind search engines draws efficiency by processing the information in some collection of documents once, producing an *index*, and then amortizing the cost of such processing over a large number of queries which access the same index. The index is basically an inverted file that maps each word in the collection to the set of URLs containing that word. It is normally built and updated incrementally. Additional processing is normally involved by performance-improving steps such as the removal of noise words, the conflation of words via stemming and/or the use of thesauri, and the use of word weighting schemes.

This model, albeit very successful, has important limitations. In fact it assumes that the collection is static, as was the case for earlier information retrieval systems. In the case of the Web the collection is highly dynamic, with new documents being added, deleted, changed, and moved all the time. Indices are thus reduced to "snapshots" of the Web. They are continuously updated by *crawlers,* yes at any given time an index will be somewhat inaccurate and somewhat incomplete. The problem is one of *scalability.* As a result, search engines' capability to satisfy user queries is hindered. Users are normally faced with very large hit lists, low *recall* (fraction of relevant pages that are retrieved), even lower *precision* (fraction of retrieved pages that are relevant), and stale information. These factors make it necessary for users to invest significant time in manually browsing the neighborhoods of (some subset of) the hit list.

A way to partially address the scalability problems posed by the size and dynamic nature of the Web is by decentralizing the index-building process. Dividing the task into localized indexing, performed by a set of *gatherers,* and centralized searching, performed by a set of *brokers,* has been suggested since the early days of the Web by the Harvest project (Bowman *et al.* 1994). The success of this approach has been hindered by the need for cooperation between information providers and indexing crawlers.

Search agents

Autonomous agents, or semi-intelligent programs making automatic decisions on behalf of the user, are viewed by many as a way of decreasing the amount of human-computer interaction necessary to manage the increasing amount of information available online (Maes 1994). Many such information agents, more or less intelligent and more or less autonomous, have been developed in the recent years. One important improvement on the quality of any search engine's performance was achieved by agents who submit queries to many different engines simultaneously and then combine the results. This technique, originally called *metasearch,* has indeed proven to increase recall significantly (Vogt & Cottrell 1998).

However, most search agents suffer from a common limitation: their reliance on search engines. The limited coverage and recency of search engines cannot be overcome by agents whose search process consists of submitting queries to search engines. Typical examples of such agents are homepage or paper finders that rely on centralized repositories to find information on behalf of the users (Bollacker, Lawrence, & Giles 1998; Shakes, Langheinrich, & Etzioni 1997; Monge & Elkan 1996)

A different class of agents are designed to learn user interests from browsing for recommendations purposes (Pazzani, Muramatsu, & Billsus 1996; Armstrong *et al.* 1995; Lieberman 1997; Chen & Sycara 1998). These agents learn to predict an objective function online and can track time-varying user preferences. However, they need supervision from the user in order to work; no truly autonomous search is possible.

Other systems, based on multi-agent paradigms, adapt a matching between a set of discovery agents (typically search engine parasites) and a set of user profiles (corresponding to single- or multiple-user interests) (Moukas & Zacharia 1997; Balabanović 1997). These systems can learn to divide the problem into simpler subproblems, dealing with the heterogeneous and dynamic profiles associated with long-standing queries. However they share the weak points of other agents who perform no active autonomous search, and therefore cannot improve on the limitations of the metasearch engines they exploit.

One last set of agent-based systems, inspired by ecological and artificial life models, actually relies on agents searching (browsing) online on behalf of the user. The first of such systems, Fish Search (De Bra & Post 1994), was hindered in effectiveness by the absence of any adaptability in the agents. One unfortunate consequence of its fixed search strategy was the possibility of load-unfriendly search behaviors, partially mitigated by the use of a cache. This factor, coupled with the growing popularity of search engines and the relatively slow performance of Fish Search, did not help to focus on the potential advantages of such models. The remainder of this paper illustrates a system that was developed in part to overcome the limitations of Fish Search and prove the strengths of this approach (Menczer, Belew, & Willuhn 1995).

Scalability

The scalability problem, limiting the effectiveness of search engines, is caused by the large size of the Web, its rapid growth rate, and its highly dynamic nature. The problem was quantified in a study that estimated the size of the Web at over 320 million pages and its growth rate at 1000% in a few years, attesting to the increasing complexity of environment (Lawrence & Giles 1998).

Lawrence and Giles also measured the *coverage* and *recency* of six among the most popular search engines. The coverage achieved by these search engines varied approximately between 3% and 34% of the Web's indexable pages. An estimate of recency was obtained by counting the fraction of returned hits corresponding to broken URLs, i.e., pages that had been deleted or moved.[1] Among the search engines considered, the one with highest coverage also had the most broken links (5%), and vice versa — the engine with lowest cover-

[1] This method did not account for URLs with changed content.

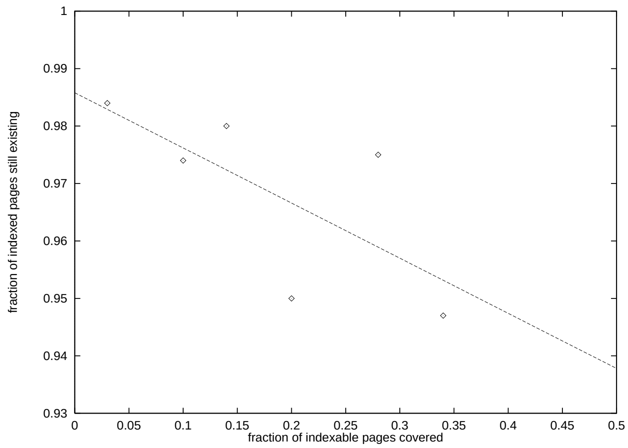

Figure 1: Scatter plot of coverage versus recency in six popular search engines: Alta Vista, HotBot, Northern Lights, Excite, InfoSeek, and Lycos. Data from (Lawrence & Giles 1998). Linear regression is also shown. The correlation coefficient is -0.7.

age was the one with highest recency. Such a trade-off between coverage and recency is illustrated in Fig. 1. Coverage and recency are indeed anti-correlated, as expected. Increasing the coverage of an index, given some limited bandwidth resource, imposes a search engine's crawler to "spread itself thin" and update pages less frequently, thus increasing the amount of stale information in the index.

In order to keep indices as up-to-date as possible, crawlers have to revisit documents often to see if they have been changed, moved, or deleted. Further, crawlers have to try to exhaustively visit every new document to keep indices as complete as possible. Such crawler behaviors impose significant loads on the net, as documents must be examined periodically. Heuristics are used to estimate how frequently a document is changed and needs to be revisited, but the accuracy of such statistics is highly volatile.

Agents browsing the Web *online* search through the *current* environment and therefore do not run into stale information. On the other hand, they are less efficient than search engines because they cannot amortize the cost of a search over many queries. Assuming that users may be willing to cope with the longer wait for certain queries that search engines cannot answer satisfactorily, it becomes imperative to determine what is the impact of online search agents on network load.

We have argued that, because of the scale effect, making an index more stale can free up sufficient network resources to completely absorb the impact of online searches (Menczer 1998). It turns out, in fact, that increasing a crawler's mean time between revisits to a document by a factor of $(1+\epsilon)$, while maintaining a constant amortized cost per query, we could refine the results of each query online using an amount of network resources scaling as

$$\frac{n}{q\tau}\frac{\epsilon}{1+\epsilon},$$

where n is the number of documents in the Web, q is the number of queries answered by a search engine per unit time, and τ is the mean time between visits to the same document by the engine's crawler. Given the size and the growth rate of n, this is an impressive amount of resources no matter what ϵ we choose. The question then becomes one of resource allocation: What ϵ achieves an appropriate balance between the network resources allocated to the upkeep of centralized search engines versus those usable by personal online agents?

Model details

The previous section suggests that users could increasingly rely on personalized tools in addition to global search engines, with the relative load of the network simply shifting from "dumb" crawlers to "smart" browsing agents. But for this vision to become reality, we must prove that an agent-based solution can indeed reach beyond search engines and effectively locate information unknown to them. We have begun to provide evidence of this possibility through experimentation of the *InfoSpiders* system, which is described in detail elsewhere (Menczer & Belew 1998a; Menczer 1998). In this section the salient features of the system are outlined.

Our approach is based on the idea of a multi-agent system. The problem is decomposed into simpler subproblems, each addressed by one of many simple agents performing simple operations. The divide-and-conquer philosophy drives this view. Each agent "lives" browsing from document to document online, making autonomous decisions about which links to follow, and adjusting its strategy to both local context and, possibly, the personal preferences of the user. Population-wide dynamics bias the search toward more promising areas.

In this framework, both individual agents and populations must *adapt* to capture the relevant features of the environment at the appropriate local scales (information neighborhoods) while covering heterogeneous areas at the global scale. To achieve this goal, agents adapt by both evolutionary and reinforcement learning.

Algorithm

InfoSpiders search online for information relevant to the user, by making autonomous decisions about what links to follow. Fig. 2 illustrates the high-level algorithm driving InfoSpiders. This is an instance of an evolutionary algorithm based on "local selection" (Menczer & Belew 1998b).

The user initially provides a list of keywords (query) and a list of starting points, in the form of a bookmark file. First, the population is initialized by pre-fetching the starting documents. Each agent is "positioned" at one of these document and given a random behavior

```
InfoSpiders(query, starting_urls, MAX_PAGES) {
   for agent (1..INIT_POP) {
      initialize(agent, query);
      situate(agent, starting_urls);
      agent.energy := THETA / 2;
   }
   while (pop_size > 0 and visited < MAX_PAGES) {
      foreach agent {
         pick_outlink_from_current_document(agent);
         agent.doc := fetch_new_document(agent);
         agent.energy +=  benefit(agent.doc) - cost(agent.doc);
         apply_Q_learning(agent, benefit(agent.doc));
         if (agent.energy >= THETA) {
            offspring := mutate(clone(agent));
            offspring.energy := agent.energy / 2;
            agent.energy -= offspring.energy;
            birth(offspring);
         }
         elseif (agent.energy <= 0) death(agent);
      }
   }
}
```

Figure 2: Pseudocode of the InfoSpiders algorithm.

(depending on the representation of agents) and an initial reservoir of energy. The user also provides a maximum number of pages that the population of agents are allowed to visit, collectively. This would depend on how long the user is willing to wait, or how much bandwidth she is willing to consume.[2]

In the innermost loop of Fig. 2, an agent "senses" its local neighborhood by analyzing the text of the document where it is currently situated. This way, the relevance of all neighboring documents — those pointed to by the hyperlinks in the current document — is estimated. Based on these link relevance estimates, the agent "moves" by choosing and following one of the links from the current document.

The agent's energy is then updated. Energy is needed in order to survive and move, i.e., continue to visit documents on behalf of the user. Agents are rewarded with energy if the visited documents appear to be relevant. The `benefit()` function is used by an agent to estimate the relevance of documents, but a documents yields energy only the first time it is visited. Agents are charged energy for the network load incurred by transferring documents. The `cost()` function should depend on used resources, e.g., transfer latency.

Instantaneous changes of energy are used as reinforcement signals, so that agents can adapt during their lifetime by reinforcement learning. By learning to predict the best links to follow, an agent can modify its behavior based on its prior experience.

Local selection means that an agent is selected for reproduction based on a comparison between its current energy level and a constant `THETA` that is independent of the other agents in the population. Similarly, an agent is killed when it runs out of energy. Offspring are also mutated, providing the variation necessary for adapting agents by way of evolution. Energy is conserved at all reproduction events.

[2] It is also possible for the user to provide the population with relevance feedback, but this paper does not discuss that aspect of the system.

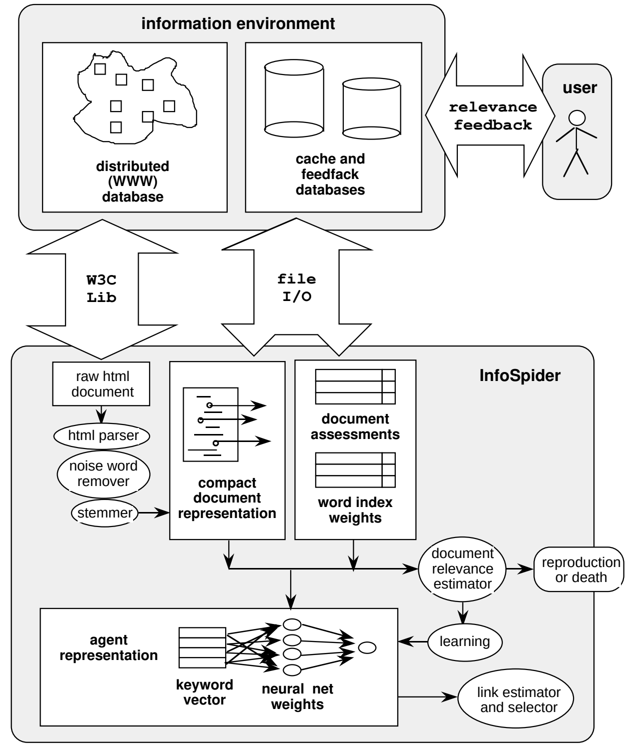

Figure 3: Architecture of an InfoSpiders agent.

The output of the algorithm is a flux of links to documents, ranked according to estimated relevance. The algorithm stops when the population goes extinct for lack of relevant information resources, visits `MAX_PAGES` documents, or is terminated by the user.

Representation

Figure 3 illustrates the architecture of each InfoSpiders agent. The agent interacts with the information environment, that consists of the actual networked collection (the Web) plus information kept on local data structures. The user interacts with the environment by accessing data on the local client (current status of a search) and on the Web (viewing a document suggested by agents).

Figure 3 highlights the central dependence of the InfoSpiders system on agent representation. The adaptive representation of InfoSpiders consists of the genotype, that determines the behavior of an agent and is passed on to offspring at reproduction; and of the actual mechanisms by which the genotype is used for implementing search strategies.

Each agent's genotype contains a list of keywords,

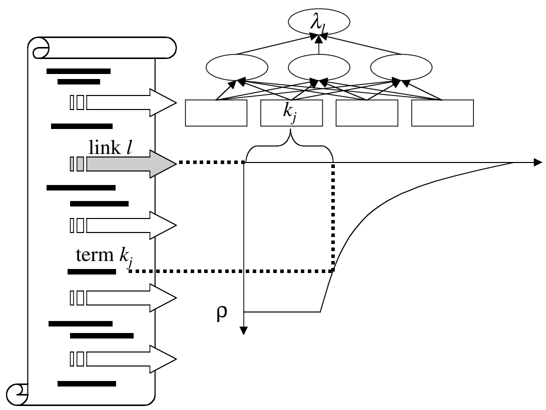

Figure 4: Estimation of a document's outlinks by InfoSpiders.

initialized with the query terms. Since feed-forward neural nets are a general, versatile model of adaptive functions, they are used as a standard computation device. Therefore genotypes also comprise a vector or real-valued weights. The keywords represent an agent's opinion of what terms best discriminate documents relevant to the user from the rest. The weights represent the interactions of such terms with respect to relevance. The neural net has a real-valued input for each keyword in its genotype and a single output unit.

An agent performs action selection by first computing the relevance estimates for each outgoing link from the current document. This is done by feeding into the agent's neural net activity corresponding to the small set of (genetically specified) keywords to which it is sensitive. Each input unit of the neural net receives a weighted count of the frequency with which the keyword occurs in the vicinity of the link to be traversed. A distance weighting function is used, which is biased towards keyword occurrences closest to the link in question. The process is illustrated in Fig. 4 and is repeated for each link in the current document. Then, the agent uses a stochastic selector to pick the next link.

After a link has been chosen and the corresponding new document has been visited and evaluated, the agent compares the (estimated) relevance of the new document with the estimate of the link that led to it. By using the connectionist version of Q-learning (Lin 1992), the neural net can be trained online to predict values of links based on local context. The value returned by the `benefit()` function is used as an internally generated reinforcement signal to compute a teaching error. The neural net's weights are then updated by backpropagation of error (Rumelhart, Hinton, & Williams 1986). Learned changes to the weights are inherited by offspring at reproduction. This learning scheme is completely unsupervised.

InfoSpiders adapt not only by learning neural net weights, but also by evolving all of the genotype components — the neural net and the keyword representation. The neural net is mutated by adding random noise to a fraction of the weights. The keyword vector is mutated by replacing the least useful (discriminating) term with a term that appears better correlated with relevance. Learning is charged with adjusting the neural net weights to the new keyword.

The evolution of keyword representations via local selection and mutation implements a form of *selective query expansion*. Based on local context, the query can adapt over time and across different locations. The population of agents thus embodies a distributed, heterogeneous model of relevance that may comprise many different and possibly inconsistent features. But each agent focuses on a small set of features, maintaining a well-defined model that remains manageable in the face of the huge feature dimensionality of the search space.

Engines vs. agents: compete or cooperate?

In previous work (Menczer 1998), the InfoSpiders system was evaluated on a limited and controlled chunk of the Web — a subset of the Encyclopaedia Britannica (Encyclopaedia Britannica, Inc.). For this dataset, a large number of test queries and relevant sets were readily available. Each query had a *depth* describing the minimum distance between the starting points and the relevant set. Depth was roughly inverse to *generality* — deeper queries were more specific and their smaller relevant sets added to their difficulty.

The collective performance of InfoSpiders was assessed and compared to *best-first-search* using a variation of the *search length* metric (Cooper 1968). We measured the total number of pages visited by InfoSpiders before some fraction of the relevant set was discovered. We found that for the more general queries, InfoSpiders had a significant advantage over best-first-search, while for the deepest queries the situation was reversed. Furthermore, both algorithms degraded in performance with increasing depth, i.e., they succeeded less frequently at locating the required fraction of relevant documents.

These results suggest using search engines to provide InfoSpiders with good starting points. If the starting pages were not too far from the target pages, agents could quickly locate relevant documents that are unknown to the search engine. To illustrate this hypothesis, we ran InfoSpiders as a front-end to a traditional search engine for a query that the search engine could not satisfy alone (Menczer & Monge 1999).

An *ad-hoc* query was constructed in such a way that the small relevant set (a total of four Web pages) was known *a priori*. Three of these pages had been recently published on the Web, so that none of the major search engines had yet indexed any of them. The remaining page was old and previously indexed by all of the ma-

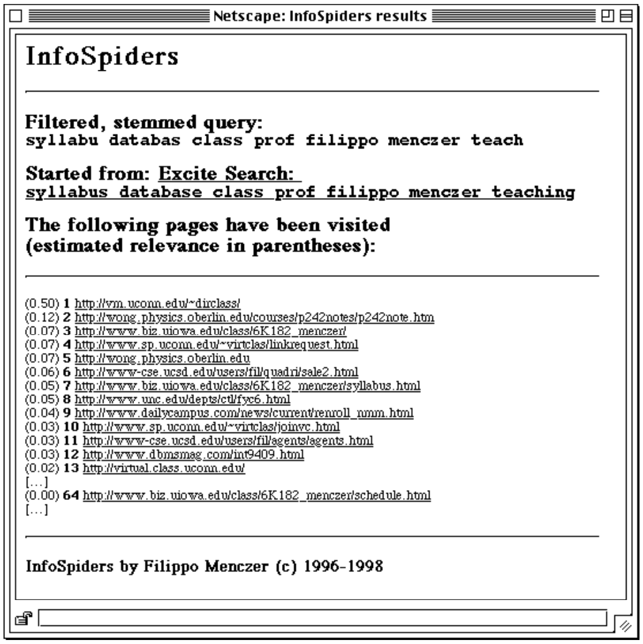

Figure 5: Report of the InfoSpiders search. The HTML document created by InfoSpiders is viewed through a browser.

jor search engines, but it had been recently updated to include, among its many links, an additional link to one of the other three relevant pages. The linked new page included, among others, links to the remaining two pages, so that all the relevant set was within two links from the indexed page.

As expected, the search engine returned the only relevant page indexed (a recall of 1/4 and a precision of about 3×10^{-7}). The hit was ranked first. A small population of 10 InfoSpiders was then initialized at the top 10 pages returned by the engine (one agent per page) and allowed to search online, adapting by evolution and reinforcement learning. After 9 minutes and 66 new pages visited, all of the relevant pages had been located (a recall of 1 and a precision of about 0.05 if we count the startup pages among those visited). Fig. 5 shows the output at the end of the InfoSpiders run. The visited pages are ranked by estimated relevance — in this case simply by their similarity to the query; the cosine matching score is listed next to each visited page. Two of the three new relevant documents (those containing query terms) were ranked in the top ten positions (3 and 7) while the third appeared later in the list.

Although this single run does not warrant any quantitative conclusions, Fig. 6 visualizes recall as a function of the network resources allocated to the online search. The purpose of this case study was merely to illustrate the potential synergy of combining the starting points provided by a search engine with the online search capabilities of InfoSpiders. The user could not have located all of the needed information with the search engines

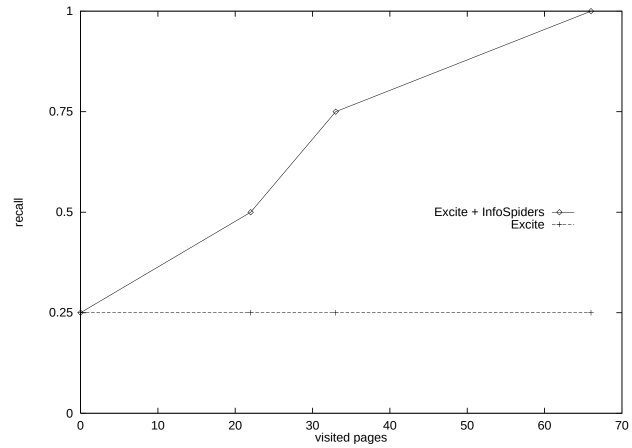

Figure 6: Plot of recall versus number of new pages visited by InfoSpiders browsing online from the results of a search engine. The recall level achieved by the search engine alone remains unaffected.

alone. The goal could have been achieved by manually browsing through the top pages returned by the search engine, but this is a very time-consuming activity — one better delegated to intelligent information agents!

Discussion

The issue of topology is now discussed, in support of the view that the two approaches of centralized search engines and online search agents are really complementary to each other, and either one alone is insufficient to achieve scalable search. We conclude with a look at the future.

Topological issues

Indexing can be described as the process of building a *word topology* over a document space. A search engine will show similar documents next to each other, effectively creating on the fly a topology based on their word statistics. This is a very useful model because the user can immediately make assumptions about the contents of retrieved documents, for example about the fact that they contain certain words.

However, hypertext information environments contain additional structure information, which can be used to provide browsing users (or agents) with helpful cues. It has been argued that this *linkage topology* — much of which is lost in the construction of indices — is indeed a very precious asset that can be exploited by browsing users in their navigation from document to document (Larson 1996; Chakrabarti *et al.* 1998; Huberman *et al.* 1998). Linkage topology is useful inasmuch as browsers have a better-than-random expectation that following links can provide them with guidance — if this were not the case, browsing would be a waste of time!

To quantify the notion of value added by linkage topology, we have conjectured that such value can be captured by the extent to which linkage topology "preserves" relevance (with respect to some query) (Menczer 1998). Imagine a browsing user or agent following a random walk strategy.[3] First define R as the conditional probability that following a random link from the current document will lead to a relevant document, given that the current document is relevant. We call R *relevance autocorrelation*. Then define G as the probability that any document is relevant, or equivalently the generality of the query.

For the random browser, the probability of finding a relevant document is given by

$$\nu = \eta R + (1 - \eta) G \ ,$$

where η is the probability that the current document is relevant. If linkage topology has any value for the random browser, then browsing will lead to relevant documents with higher than random frequency. In order for this to occur the inequality $\nu > G$ must hold, which upon simplifying for η is equivalent to $R > G$. We can then express the linkage topology value added by defining the quantity

$$\Theta = R/G - 1 \ .$$

We measured Θ for a few queries from a couple of search engines and found a positive value added by linkage topology: in Lycos (Lycos), for example, we found $\Theta = (9 \pm 3) \times 10^3 \gg 0$. Thus the confidence that browsing is a reasonable task for autonomous agents seems justified.

Linkage topology is not a sufficient condition for an effective search, however. For a given query q, it seems plausible that relevance autocorrelation decays rapidly for distances greater than some correlation distance $\Delta_{R,q}$. If an agent is farther than $\Delta_{R,q}$ links away from a relevant document, the search is blind. Since $\Delta_{R,q}$ is unknown, there is no way to estimate the necessary amount of energy with which to initially endow agents at any arbitrary starting point. If such amount is overestimated, resources will be unnecessarily wasted searching through unlikely neighborhoods. And if it is underestimated, extinction will ensue before any relevant document is located. This is consistent with the results outlined in the previous section. It appears then crucial to launch InfoSpiders from "good" starting points. That is, a starting point should be within a radius $\Delta_{R,q}$ of a target page.

Search engines can rely on word topology to provide agents with good starting points. Even if the index does not contain the target, it may contain pages within a radius $\Delta_{R,q}$ of it. If this hypothesis holds — something that must be verified empirically — then linkage topology can lead the agents in the right direction. We believe that the information encoded by word and linkage topologies are complementary. Search engines and browsing agents should work together to serve the user most effectively.

[3]The conservative assumption of random walk should yield a lower bound for the value added of linkage topology.

The future

This paper has discussed the scalability limitation of search engines and suggested a solution based on populations of adaptive information agents searching online. The case study in the previous section has illustrated the potential search scalability achievable through the synergy between search engines and online browsing agents.

The viability of adaptive information agents in achieving scalable Web search, however, cannot be demonstrated with anecdotal evidence. Quantitative confirmation of the ideas discussed in this paper must be sought through extensive testing on the Web. One experiment would have human browsers and InfoSpiders compete in locating documents not indexed by search engines. Another approach would be to create a new page and measure how long it takes, on average, until it is found by a crawler (provided it is not directly submitted to it by the author); this time can be compared to the average time it takes InfoSpiders to find the page, starting from appropriate queries and different hit lists derived from search engines.

The InfoSpiders prototype is under continuous development to allow for such experiments. Many aspects of the model also remain to be explored in the "real world," from the use of relevance feedback under long-standing queries to the role of recombination and from the effect of cache size to distributed implementations.

Beyond such explorations we envision that in the growing and increasingly complex Web of information, users will need to rely heavily on intelligent agents. If the feasibility of agent-based online search is proven, people will be able to delegate more and more of their tedious tasks to their personal agents.

Acknowledgments

Alvaro Monge was instrumental in the design and execution of the case study illustrated in this paper. The InfoSpiders project originated from a collaboration with Wolfram Willuhn and Richard K. Belew. This work is partially supported by a CIFRE grant from the University of Iowa.

References

Armstrong, R.; Freitag, D.; Joachims, T.; and Mitchell, T. 1995. Webwatcher: A learning apprentice for the world wide web. In *Working Notes of the AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments*.

Balabanović, M. 1997. An adaptive web page recommendation service. In *Proc. 1st International Conference on Autonomous Agents*.

Bollacker, K.; Lawrence, S.; and Giles, C. 1998. CiteSeer: An autonomous web agent for automatic retrieval and identification of interesting publications. In *Proc. 2nd International Conference on Autonomous Agents*.

Bowman, C.; Danzig, P.; Manber, U.; and Schwartz, M. 1994. Scalable internet resource discovery: Research problems and approaches. *Communications of the ACM* 37(8):98–107.

Chakrabarti, S.; Dom, B.; Raghavan, P.; Rajagopalan, S.; Gibson, D.; and Kleinberg, J. 1998. Automatic resource compilation by analyzing hyperlink structure and associated text. In *Proc. 7th International World Wide Web Conference*.

Chen, L., and Sycara, K. 1998. WebMate: A personal agent for browsing and searching. In *Proc. 2nd International Conference on Autonomous Agents*.

Cooper, W. 1968. Expected search length: A single measure of retrieval effectiveness based on weak ordering action of retrieval systems. *Journal of the American Society for Information Science* 19:30–41.

De Bra, P., and Post, R. 1994. Information retrieval in the world wide web: Making client-based searching feasible. In *Proc. 1st International World Wide Web Conference*.

Encyclopaedia Britannica, Inc. http://www.eb.com.

Huberman, B.; Pirolli, P.; Pitkow, J.; and Lukose, R. 1998. Strong regularities in world wide web surfing. *Science* 280(5360):95–97.

Larson, R. 1996. Bibliometrics of the world wide web: An exploratory analysis of the intellectual structure of cyberspace. In *Proc. 1996 Annual ASIS Meeting*.

Lawrence, S., and Giles, C. 1998. Searching the world wide web. *Science* 280:98–100.

Lieberman, H. 1997. Autonomous interface agents. In *Proc. ACM Conference on Computers and Human Interface*.

Lin, L.-J. 1992. Self-improving reactive agents based on reinforcement learning, planning, and teaching. *Machine Learning* 8:293–321.

Lycos. http://www.lycos.com.

Maes, P. 1994. Agents that reduce work and information overload. *Comm. of the ACM* 37(7):31–40.

Menczer, F., and Belew, R. 1998a. Adaptive information agents in distributed textual environments. In *Proc. 2nd International Conference on Autonomous Agents*.

Menczer, F., and Belew, R. 1998b. Local selection. In *Evolutionary Programming VII*, number 1447 in Lecture Notes in Computer Science. Springer.

Menczer, F., and Monge, A. 1999. Scalable web search by adaptive online agents: An InfoSpiders case study. In Klusch, M., ed., *Intelligent Information Agents: Agent-Based Information Discovery and Management on the Internet*. Springer. Forthcoming.

Menczer, F.; Belew, R.; and Willuhn, W. 1995. Artificial life applied to adaptive information agents. In *Working Notes of the AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments*.

Menczer, F. 1998. *Life-like agents: Internalizing local cues for reinforcement learning and evolution*. Ph.D. Dissertation, University of California, San Diego.

Monge, A., and Elkan, C. 1996. The WEBFIND tool for finding scientific papers over the worldwide web. In *Proceedings of the 3rd International Congress on Computer Science Research*.

Moukas, A., and Zacharia, G. 1997. Evolving a multi-agent information filtering solution in amalthaea. In *Proc. 1st International Conference on Autonomous Agents*.

Pazzani, M.; Muramatsu, J.; and Billsus, D. 1996. Syskill & Webert: Identifying interesting web sites. In *Proc. National Conference on Artificial Intelligence (AAAI96)*.

Rumelhart, D.; Hinton, G.; and Williams, R. 1986. Learning internal representations by error propagation. In Rumelhart, D., and McClelland, J., eds., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1. Cambridge, MA: Bradford Books (MIT Press).

Shakes, J.; Langheinrich, M.; and Etzioni, O. 1997. Dynamic reference sifting: A case study in the homepage domain. In *Proc. 6th International World Wide Web Conference*.

Vogt, C., and Cottrell, G. 1998. Predicting the performance of linearly combined ir systems. In *Proceedings of the ACM SIGIR Conference*.