ARACHNID: Adaptive Retrieval Agents Choosing Heuristic Neighborhoods for Information Discovery

Filippo Menczer
Computer Science and Engineering Department
University of California, San Diego
La Jolla, CA 92093–0114, USA
fil@cs.ucsd.edu

Abstract

ARACHNID is a distributed algorithm for information discovery in large, dynamic, distributed environments such as the World Wide Web. The approach is based on a distributed, adaptive population of intelligent agents making local decisions. The behavior of the algorithm is analyzed using a simplified model of the Web environment. This analysis highlights an interesting feature of the Web environment that bodes well for ARACHNID's search methods. The performance of the algorithm is illustrated using both simulated environments and preliminary experiments in which prototype agents search real Web environments. Interactions are also discussed between unsupervised learning by individual agents and evolution at the population level, along with the role played in both by user relevance feedback.

1 INTRODUCTION

Imagine that you are looking for some just-released product on the Web. You probably have a list of starting points, provided by your favorite search engine or by browsing some digital library. At this point you are probably going to browse manually through the most promising links, in a fashion resembling best-first-search, until you are satisfied that further browsing will not lead you to much more useful information.

Users invest a large amount of time in such manual searches. This would not be necessary if search engines could perfectly identify the sets of pages relevant to their queries, but we all know from experience that this is not possible. The growth of the Internet is posing new challenges to disciplines like information retrieval that have provided useful techniques for search problems like these. This paper focuses on several key characteristics that make the Web different. This environment is *large*, and growing at a fast pace; *distributed*, imposing differential costs on accessing documents; *heterogeneous*, with respect to document format, style, and content; and *dynamic*, with documents being added, deleted, or rearranged constantly. These features suggest the use of algorithms based on large populations of distributed agents, each capable of making local decisions. Further, the availability of powerful learning techniques and "relevance feedback" information generated by browsing users suggests that these agents could adapt their behavior with experience.

This paper describes the use of adaptive intelligent methods to automate on-line information search and discovery in the Web or similar networked environments. Stated concisely, the problem consists in locating as many documents as possible that are relevant to some user query, in as little time as possible. All the system can do is analyze document text and follow hyperlinks between documents. I introduce a simple algorithm, inspired by prior work on agents adapting in environments sharing many characteristics with the Web (Menczer and Belew 1996a, 1996b), and analyze its efficiency in an abstracted model of the Web. The algorithm is tested on simulated environments, and results of preliminary experiments on an interesting piece of the Web are outlined.

2 BACKGROUND

The traditional approach to retrieving information on the Web is an extension of the solution of the same problem for static, non-distributed collections. In both cases, the user submits the query to a centralized server, which in turn uses an index to retrieve a list of documents matching the query. In the classic case, the algorithm that constructs the index scans the whole corpus once. But for Web search engines, the index is usually built and updated on-line in an incremental fashion. Such work is performed by automatic programs, called *robots* (or *crawlers,* etc.), that attempt to visit exhaustively every existing document. This solution, however, does not scale equally well in the two domains. In the centralized case, if the collection changes the index must be rebuilt entirely. However, this is rare and thus the cost of building the index is amortized over many queries. In the distributed case, the cost is much larger because robots do not know when and where the changes occur, and so must revisit the same documents periodically. Transferring the documents over the Internet is also more expensive. Robots impose a high load on network resources, although these costs are hidden when the user submits a query. More active robots would improve accuracy at enormous costs in network load. Hence the need to complement index-based search engines with intelligent browsing at the user's end.

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). Several machine learning techniques have been suggested to produce effective information agents, yielding for example agents that perform lookahead searches and provide suggestions to the user on the basis of reinforcement learning (Armstrong et al. 1995). Techniques such as weighted keyword vector representations and relevance feedback, in conjunction with genetic algorithms and/or paradigms inspired by artificial life, have been applied to information retrieval and filtering (Sheth and Maes 1993; Kraft et al. 1995; Balabanović and Shoham 1995; Moukas 1996). Such approaches have led to multi-agent information systems where both population-based evolution and individual-based learning occur. We have proposed the application of these machine learning methods to the problem of information discovery (Menczer et al. 1995).

Traditional genetic algorithms use selection schemes in which some global operation (e.g., ranking or averaging) is performed on the fitness of the population to decide which genotypes get to reproduce. As a result, the population typically converges to some solution, hopefully the optimum. This unimodal behavior is not the most appropriate in the information discovery domain, where as many relevant documents as possible need be retrieved, rather than just the most relevant one. Techniques dealing with multimodal optimization problems, such as fitness sharing (Horn 1996), have been suggested in the genetic algorithms literature, but they are not efficient in distributed environments because they still require global operations (Mahfoud 1995). The approach proposed here employs a genetic algorithm based on *local selection.* An agent's fitness is not compared with other members of the population. Rather, some measure of fitness (*energy*) is accumulated over time; selection occurs in a distributed fashion, comparing energy with a fixed threshold. This selection scheme is inspired by artificial life models with endogenous fitness (Menczer and Belew 1996b). The result is the same type of non-converging behavior required for multimodal optimization, and therefore local selection is appropriate for the information discovery problem. A related method has been proposed by De Bra and Post (1994); in their Fish Search algorithm, however, all agents are identical clones following fixed, exhaustive search strategies.

```
0. situate, initialize agents with E = ε/2
while there are alive agents:
   1. pick random agent a
   2. pick link for a to follow
   3. fetch new document D_a
   4. E_a ← E_a − cost + { r(D_a)   if D_a new
                         { f(D_a)   otherwise
   5. mark D_a as visited
   6. learn by reinforcement
   7. if (E_a > ε)
         a' ← mutate(clone(a))
         E_a, E_a' ← E_a/2
      else if (E < 0)
         die(a)
```

Figure 1: Pseudo-code for the ARACHNID algorithm.

3 THE ALGORITHM

The ARACHNID algorithm is shown in Figure 1. To keep the fundamental algorithm schema as general as possible, several representation-specific details are left as black-boxes until later subsections. Notice that the threshold ϵ is a constant, so the decision whether an agent should reproduce is local (independent of other agents), making ARACHNID a distributed algorithm.

Off-line, the user may assess the relevance of (some of) the documents visited by the algorithm up to a certain point. Such assessments can alter the behaviors of online agents. These interactions occur through the computations of energy payoffs r and f from new and visited documents, respectively (step (4)). The relevance feedback mechanism also depends on the agent representation. In general, the function f will be correlated with the user's assessment of a document's relevance.

The user initially provides a list Q of keywords and a list of starting points, in the form of a bookmark file.[1] In step (0), the population is initialized by pre-fetching the starting documents. Each agent is "positioned" at one of these documents and given a random behavior (depending on the representation) and an initial reservoir of "energy." In step (2), each agent "senses" its

[1]Obtained for example by consulting a search engine.

local neighborhood by analyzing the text of the document where it is currently situated. This way, the relevance of all neighboring documents —pointed to by the hyperlinks in the current document— is estimated. Based on these link relevance estimates, an agent "moves" by choosing and following one of the links from the current document. In step (4), the agent's energy is updated. Energy is needed in order to survive and continue to visit documents on behalf of the user. Agents are charged costs for the use of network resources incurred by transferring documents.[2] Agents are also rewarded with energy if the visited documents appear to be relevant. That is the role of the r function, which an agent uses to estimate the relevance of documents. Steps (4) and (5) of the algorithm embody the principle of conservation of resources. Agents visiting marked documents do not get energy from them (unless by user's feedback). This mechanism is implemented via a cache, which also speeds up the process by minimizing duplicate transfers of documents. Caching documents is a form of communication, and thus a bottleneck for the parallelism of distributed algorithms. However, since network communication is the most expensive resource for the algorithm, the performance improvement warranted by the use of a cache should outweigh any such degradation. Instantaneous changes of energy can be used, in step (6), as reinforcement signals. This way agents can adapt during their lifetime, if their representation allows for it. An agent can modify its behavior based on prior experience, by learning to predict the best links to follow. In step (7), an agent may be killed or be selected for reproduction. In the latter case the offspring is situated at the same document location as the parent, and mutated to provide evolution with the necessary variation. Evolutionary adaptation differs from learning in two important respects. First, it does not change the reproducing agents, but biases the population towards successful individuals. Second, since selection is based on energy accumulated over many steps, evolution averages out short-term energy fluctuations and can thus rely on better assessments of agent behaviors. The output of the algorithm is the flux of documents visited by the population.

3.1 NAIVE REPRESENTATION

Let us first describe one particularly simple representation that allows for very elementary behaviors and limited relevance feedback interactions with the user. Its simplicity, however, lends itself to characterizing, statistically and by way of simulations, what information environments are best suited for this approach.

[2]Costs can be a function of used resources, for example transfer latency. For simplicity, a constant cost is assumed here for new documents, and no cost for cache hits.

A representation consists of the genotype, which is passed on to offspring at reproduction, and of the actual mechanisms by which the genotype is used to implement behaviors. In this first, "naive" representation an agent's genotype consists of two parameters $\beta, \gamma \in \Re^+$. Roughly, the former represents the degree to which an agent trusts the descriptions that a page contains about its outgoing links, and the latter represents the degree to which an agent trusts the user's relevance assessments. Behaviors are implemented in the algorithm of Figure 1 as follows (numbers in parentheses refer to algorithm steps).

Initialization (0) $\forall a$:

$$\begin{aligned} \beta_a &\leftarrow U[0, \beta_{max}] \\ \gamma_a &\leftarrow U[0, \gamma_{max}] \end{aligned}$$

where U is the uniform random distribution.

Action selection (2) The agent first computes relevance estimates for each link from the current document by weighted sums: $\forall l \in D_a$

$$\lambda_l = \sum_{k:\text{keywords} \in D_a} \frac{match(k, Q)}{dist(k, l)}$$

where

$$\begin{aligned} match(k, Q) &= \begin{cases} 1 & \text{if } k \in Q \\ 0 & \text{otherwise} \end{cases} \\ dist(k, l) &= \text{number of links in } D_a \\ & \text{separating } k, l \end{aligned}$$

Then, the agent uses a stochastic selector to pick a link with probability distribution:

$$\Pr[l] = \frac{e^{\beta \lambda_l}}{\sum_{l' \in D_a} e^{\beta \lambda_{l'}}}$$

Relevance feedback (4) If the user has assessed the relevance of the current document as $\phi(D_a) \in \{-1, +1\}$, agent a receives energy

$$f(D_a) = \gamma_a \phi(D_a)$$

Relevance computation (4) The agent estimates the relevance of a document by computing the cosine matching function between the document and the query, normalized to an appropriate energy range:

$$r(D_a) = \sqrt[4]{\frac{\sum_{k \in D_a} match(k, Q)}{|\text{keywords} \in D_a|}}$$

Learning (6) None occurs with this representation.

Mutation (7) If a' is an offspring of a:

$$\begin{aligned} \beta_{a'} &\leftarrow U[\beta_a(1 - \kappa_\beta), \beta_a(1 + \kappa_\beta)] \\ \gamma_{a'} &\leftarrow U[\gamma_a(1 - \kappa_\gamma), min\{\gamma_a(1 + \kappa_\gamma), \gamma_{max}\}] \end{aligned}$$

where $0 \leq \kappa_\beta, \kappa_\gamma \leq 1$ are parameters. While β's can grow without bound, the γ distribution is clipped to γ_{max} to discourage "stationary" behaviors.

3.2 VECTOR REPRESENTATION

Let us next consider an alternative representation that may better be suited to detect useful features of the information environment, and internalize them into adaptive behaviors —learned or evolved. Since information is most easily available and analyzable as text, the features we want to focus on are words. Therefore each agent's genotype will contain a vector of keywords, W. Feed-forward neural nets will also be employed as adaptive computational devices. Therefore genotypes will also contain a vector of real-valued weights. The keywords will represent an agent's opinion of what terms best discriminate relevant documents from the rest. The weights will represent the relative —possibly negative— importance of such terms. Let us now sketch how the algorithm of Figure 1 will be working on the basis of this representation.

Initialization (0) Each agent's word vector is initialized with the query terms; its neural net architecture is given as many inputs as query terms, and provided with random initial weights distributed uniformly in a small interval around 0.

Action selection (2) An agent estimates the relevance associated with each link by considering the words around the link similarly to the naive representation, except that the agent's keyword vector is used in place of the original query. For each link l, the weighted frequencies of terms $w \in W$ in the current document are fed to the agent's neural net, that computes the relevance estimate λ_l. As for the naive representation, a stochastic selector is used to explore the consequences of alternative moves.

Relevance feedback (4) Energy is obtained from relevance assessments as in the naive representation. In addition, a "feedback list" of encountered words is maintained. Each word in this list, k, is associated with a signed counter ω_k that is updated each time a document D is assessed by the user: $\forall k \in D$

$$\omega_k \leftarrow \begin{cases} \omega_k + 1 & \text{if } \phi(D) = +1 \\ \omega_k - 1 & \text{if } \phi(D) = -1 \end{cases}$$

Relevance computation (4) A newly visited document D yields energy:

$$r(D) = \tanh\left(\sum_{k \in D} freq_{k,D} \cdot I_k\right)$$

where $freq_{k,D}$ is the frequency of term k in document D and I_k is the TFIDF[3] weight of term k based on relevance feedback:

$$\begin{aligned} I_k &= \omega_k \cdot \left[1 + \log\left(\frac{1}{C_k}\right)\right] \\ C_k &= \text{fraction of cache documents containing } k \end{aligned}$$

Learning (6) This representation lends itself naturally to learning by way of connectionist Q-learning (Lin 1992). After an agent visits document D, $r(D)$ is used as the reinforcement signal. The agent's network weights are adjusted by back-propagation of error, using teaching input:

$$r(D) + \mu \cdot \max_{l \in D}\{\lambda_l\}$$

where μ is a discount factor.

Mutation (7) Weights are mutated by adding random noise. The offspring's word vector is also mutated, replacing term $\arg\min_k(|I_k|)$ by $\arg\max_k(freq_{k,D} \cdot |I_k|)$, where D is the document of birth.

4 THE ENVIRONMENT

It was stated that performing useful tasks in networked information environments is difficult because they are large, dynamic, and non-centralized. Let us now characterize such environments a bit more closely. Information providers impose a structure on the distributed database via hypertext, as is the case for the Web. The environment can thus be represented as a graph (V, E), with $N = |V|$ nodes corresponding to documents, and edges corresponding to hyperlinks.

Statistical features such as word frequencies, and physical properties such as network organization and protocols, are of course important dimensions of the environment that should be exploited. I want to argue that the "semantic" structure imposed upon the organization of documents by manually constructed hyperlinks provides for another useful resource. This is more clear in the case of well-structured information environments, where editors ensure that documents are organized according to meaningful classification criteria. But even in unstructured information spaces, such as the Web, authors tend to cluster documents about related topics by letting them point to each other via hyperlinks. I refer to such hypothetical property as *semantic topology.* Whether this structure can be detected and exploited by intelligent algorithms seems a worthwhile question to investigate.

The idea is to identify conditions under which an ecology of information agents will outperform non-adaptive search algorithms, such as breadth-first-search or random-walk. Let us begin by comparing ARACHNID with an algorithm in which, at each time step, a random document is retrieved. If correlations

[3] Actually, this is a variation of the "term frequency–inverse document frequency" index weighting scheme, using algebraic term frequencies.

in relevance across links can be detected, the ARACHNID population should exploit the locality of the document space by way of local reproduction and by following edges from current locations; let us evaluate this advantage over random search. To simplify the analysis, assume that (i) each agent follows a random edge out of its current node; and (ii) interactions among agents are negligible. These assumptions are reasonable at the beginning of an ARACHNID run, and become less reasonable as time passes. *Precision* is defined as the fraction of returned documents that are considered relevant by the user. Since we are interested in the time-course of the retrieval process, let us instead employ as a measure of performance the *rate* at which relevant documents are retrieved.

Define a relevance measure $\rho : \{v \in V\} \mapsto \{0, 1\}$ with respect to some query. Then characterize the *generality* of the query by

$$G \equiv \Pr[\rho(u) = 1 | u \in V] = \frac{1}{N} \sum_{v \in V} \rho(v).$$

The constant expected rate at which relevant documents are retrieved by the random algorithm is given by $\nu_{random} = G$. If one knows that a certain document is relevant, one may expect that some of its neighbors will be as well. Define the *relevance autocorrelation* of a graph as the conditional probability

$$R \equiv \Pr[\rho(v) = 1 | \rho(u) = 1, u, v \in V, (u, v) \in E].$$

G and R are two parameters by which I will characterize the search graph with respect to a query. Once the algorithm starts running, one can define the fraction $\eta_t \equiv g_t/p_t$, where p_t is the current size of the population and g_t is the number of agents in the current population which are located at relevant nodes. The expected rate at which relevant documents are retrieved by the ARACHNID algorithm is then

$$\nu_{spider}(t) = \eta_t R + (1 - \eta_t)G,$$

yielding a performance improvement

$$\delta(t) \equiv \frac{\nu_{spider}(t)}{\nu_{random}} = \frac{\eta_t R + (1 - \eta_t)G}{G}. \quad (1)$$

It is easy to see that $\delta(t) > 1$ if and only if

$$R > G. \quad (2)$$

The hypothesis of a useful correlation between relevant documents is expressed by condition (2), which I will call *semantic topology conjecture* for realistic queries and information environments.

η_t can be derived recursively, given G and R, to estimate performance over time. The derivation is omitted here for lack of space. The recursions for η_t have been solved numerically and equation (1) has been used to obtain performance estimates for ARACHNID with respect to random search. One way to proceed is to see

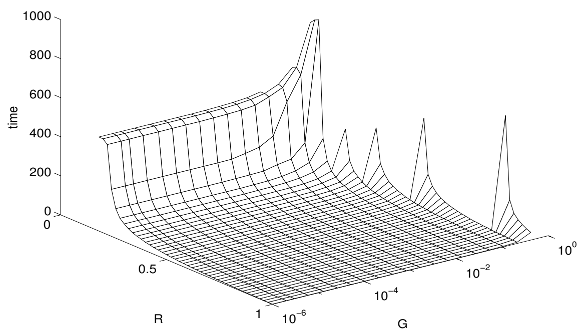

Figure 2: Time necessary for ARACHNID to reach a frequency of retrieval of relevant documents double that of a random algorithm, as a function of generality G and relevance autocorrelation R.

how many nodes must be accessed until some improvement is obtained. Such times are plotted in Figure 2, using $\delta(t) \geq 2$, for different parametric characterizations $(G, R|R > G)$. The time required to outperform random search by 100% increases sharply as $G \to 1$, a situation in which most documents are relevant, as would be the case for example in the presence of noise words in the query. Performance degrades similarly as $R \to G$, as expected; when two relevant documents are no "closer" to each other than any two random documents, there is no advantage in local search. But over a wide range of reasonable parameterizations, the ARACHNID algorithm is well-behaved.

Bibliometric studies suggest that the Web environment possesses a well-structured semantic topology (Larson 1996). This observation bodes well for the ARACHNID approach. The semantic topology conjecture can also be tested directly; if $R > G$, ARACHNID should outperform random search under the assumptions above. Realistic estimates of $R(G)$ can only be determined by gathering statistics from the actual Web. I have performed some preliminary measures of R for a set of realistic queries corresponding to a range of different G values. Ten queries were fed to WAIS engines searching in two hypertext collections: Lycos, a large index of actual Web pages, and Britannica Online, a better-structured article database (Lycos; EB). Each retrieved set was then used to approximate the relevant set and collect relevance autocorrelation statistics. R was obtained by counting the fraction of links, from documents in the relevant set, pointing back to documents in the set, and then plotted against the size of the relevant set, normalized by the size of the collection. The results (Figure 3) indicate that information is in fact structured in such a way as to support the semantic topology conjecture.

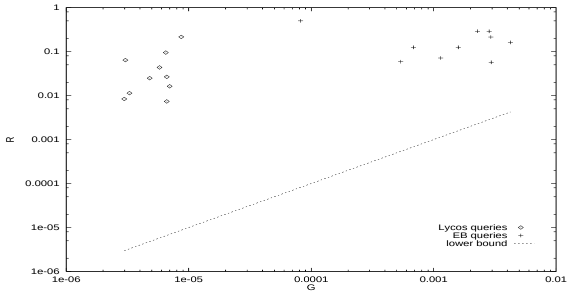

Figure 3: Semantic topology measures from the Lycos and EB search engines. G values are normalized by $N \sim 10^5$ for EB and $N \sim 10^7$ for Lycos. The lower bound is $R = G$.

5 SIMULATIONS

Before comparing ARACHNID with other search algorithms on real data, I have run experiments with simulated models of the Web. Graphs are constructed at run-time, with relevance and topology information (N, G, R, etc.) built *ad-hoc* in accordance to user-specified parameters. While this Web model must be validated by experiments with actual data, it allows one to efficiently explore algorithmic behaviors with respect to these parameters. The ARACHNID simulator is implemented with the naive agent representation described in Section 3.1. Both ARACHNID and other search algorithms can cache visited documents. The number of *new* documents retrieved is used as a measure of time.

5.1 COMPARISON WITH BFS

Breadth-first-search (BFS) is used as a baseline for performance evaluation. This is a typical exhaustive search algorithm that will visit every node in time $\Theta(N)$. BFS, too, can be viewed as a local algorithm, in the sense that new nodes are always visited by following edges from previously visited ones. However, it cannot focus adaptively on promising areas. ARACHNID instead can adapt its exploration/exploitation behavior to the local conditions encountered by each agent, by selective reproduction and the evolving β parameter.

Figure 4 shows a comparison between BFS and ARACHNID in three simulation experiments. Here performance is measured by *recall,* defined as the fraction of relevant documents that are retrieved.[4] Let us observe the implications of different models of seman-

[4] Recall captures how many relevant documents are suggested to the user, and how many are not. Precision would also be a valid measure. It can be obtained — modulo the generality constant — dividing each recall level by the corresponding value along the x axis.

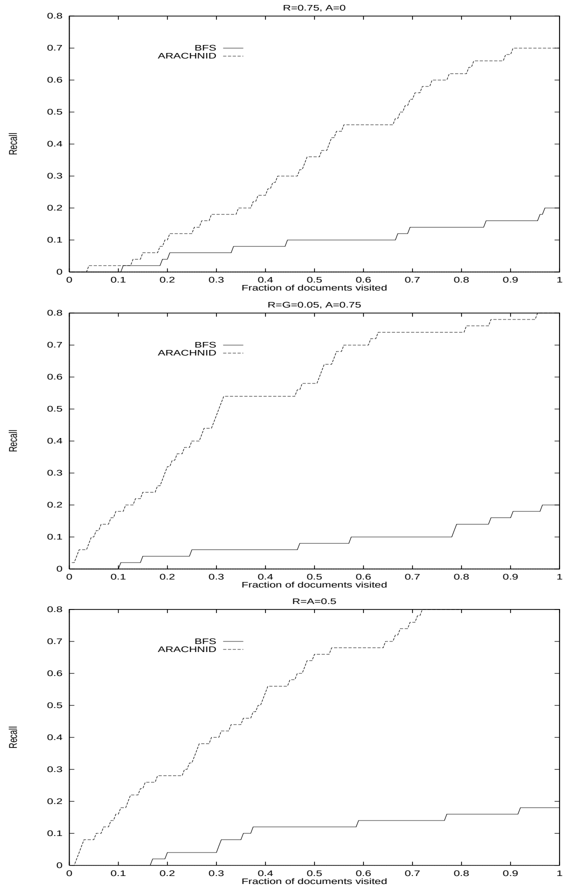

Figure 4: ARACHNID versus BFS on three simulated graphs with $N = 1000$, $\langle\text{fan-in}\rangle = \langle\text{fan-out}\rangle = 10$, and $G = 0.05$. The plots show recall as a function of the fraction of documents retrieved, out of a total of 200.

tic topology and link estimation accuracy. Semantic topology is as usual modeled by R. Link estimation *accuracy* is modeled by the quantity $A \equiv \Pr[\lambda_{u,v} = \rho(v)]$ from which the simulator draws link estimates.

In the first experiment, the semantic topology counts ($R = 0.75$) while link estimates are at noise level ($A = 0$). Evolving β values are of no consequence. This result shows that semantic topology is indeed a sufficient condition for ARACHNID to outperform exhaustive algorithms. In the second experiment the situation is reversed: there is no semantic topology ($R = G$), but the links provide useful hints for the agents ($A = 0.75$). These can be exploited by strategies with evolving β values. The cues provided by this environment allow ARACHNID to secure an even better performance. The third experiment attempts to model a more realistic situation in which the semantic topology is somewhat detectable (less than in the first experiment) and the link estimates are somewhat reliable (less than in the second experiment). Once again,

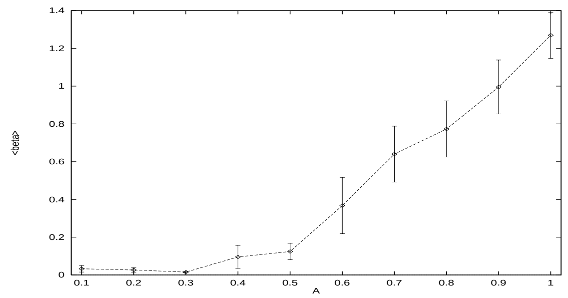

Figure 5: Population averages of evolved β values for a range of link estimation accuracies (means and standard errors over repeated simulation runs).

ARACHNID significantly outperform BFS.

5.2 INTERNALIZATION

Does evolution allow agents to detect important features of the information space and adapt their strategies accordingly? Given the naive behaviors implemented by the simulator, a partial answer to this question can be obtained by analyzing the β distribution over the population of agents.

I have run ten simulations with $G = 0.1$, $R = 0.75$, and the other parameters as in Figure 4. In each run β is initialized with uniform distribution in the range $[0, 5]$ and measured after 750 document accesses. As shown in Figure 5, β does in fact evolve to match the reliability encoded into the links: inaccurate links (low A) drive β to low (random-walk) values, while accurate links (high A) push β toward higher (best-first) values. The positive correlation between A and β indicates that the population has "internalized" the environmental feedback about the accuracy with which agents estimate link relevance.

5.3 LOCAL SELECTION

Through the simulator, support can be sought for the claimed superiority of ARACHNID versus a more traditional machine learning approach —standard genetic algorithms— for this class of problems. The main feature discriminating between the two is local versus global selection. This difference can be captured by considering the reproduction threshold ϵ (cf. Figure 1); to model global selection, ϵ is the average energy level of the current population, rather than a constant.

I carried out a series of experiments with different simulated environments, each a random graph with link accuracy $A = 0.5$ and generality $G = 0.1$. The relevant nodes are grouped into separate components, or niches. One can think of such niches as local groups of documents, altogether making up the relevant set

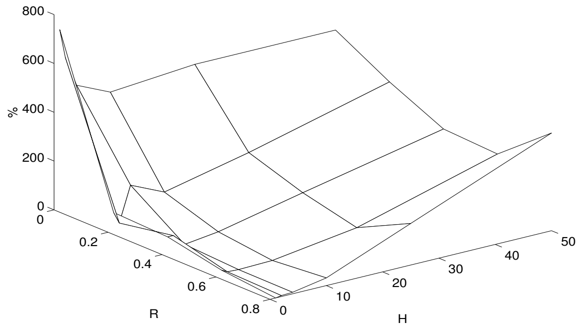

Figure 6: Recall improvement (%) of local selection over global selection, on different graphs parameterized by semantic topology (R) and multimodality (H).

corresponding to multiple or ambiguous queries. Environments are parameterized by semantic topology (measured by relevance autocorrelation, R) and multimodality (measured by number of niches, H). R and H are varied to test the hypothesis that, as environments becomes more complex, local selection provides an advantage over global selection schemes used in genetic algorithms. For each environment parameterization (R, H) both algorithms ran for the same amount of time (200 nodes visited). The results are shown in Figure 6. The plot shows, along the z axis, the recall improvement of local over global selection. With the single exception of the $(R = 0.8, H = 1)$ environment, global selection leads the population to premature convergence on a subset of the relevant documents, and the consequent extinction makes complete recall impossible. On the contrary, ARACHNID consistently achieves 100% recall. Local selection shows the greatest improvements when relevance is poorly correlated (small R) and/or highly multimodal (large H); in these conditions the recall level achieved by global selection degrades due to over-exploitation of information.

6 ON THE WEB

For validation and testing purposes, I have implemented two client-based prototypes of ARACHNID to search the actual Web on-line. Agents employ information retrieval tools such as a filter for noise words and a stemmer based on Porter's (Frakes and Baeza-Yates 1992). An efficient representation of visited documents is stored in a shared cache.

Some very preliminary experiments have been performed using these ARACHNID prototypes. Evaluation is a main problem in the Web due to the difficulty of collecting relevance assessments. A "subset" of the Web, the Encyclopaedia Britannica corpus (EB),

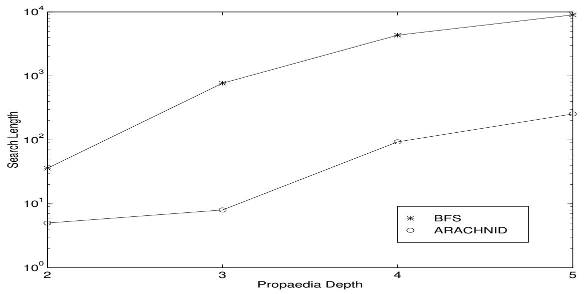

Figure 7: Search length of ARACHNID and BFS for queries at different depth in the EB Propaedia tree.

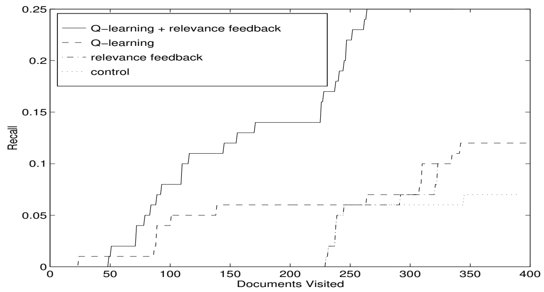

Figure 8: Effects of Q-learning and relevance feedback on ARACHNID performance. While a recall level of 25% may appear quite low, it must be pointed out that less than 3% of the documents have been seen.

has been therefore chosen. In this environment relevant sets of articles, associated with a large number of queries, are readily available. Articles are in fact organized into a hierarchical topical tree, called *Propaedia,* which is manually built and updated by skilled human editors. The category title of any Propaedia node can be used as a query, and the articles associated with the subtree rooted at that node as its relevant set. In addition, each article has hyperlinks to one or more Propaedia categories and possibly to other, semantically related articles, forming a graph. In the two experiments outlined below, the search is limited to a subtree of the Propaedia containing approximately 700 category nodes and 11,000 document nodes.

The first prototype implements the naive representation of Section 3.1, with the limitations on the range of behaviors and relevance feedback that have already been discussed. It has been used to measure *search length,* defined as the number of non-relevant documents visited until the first relevant document is found. Search lengths of a single ARACHNID run and of BFS are compared for four queries, each at a different depth in the Propaedia tree. Notice that since ARACHNID is a randomized algorithm, a single run provides a rather rough measurement of search length. The results, shown in Figure 7, are nevertheless promising: ARACHNID finds the first relevant document between one and two orders of magnitude more quickly than BFS.

The second prototype implements the more complex vector representation of Section 3.2, enabling richer forms of adaptation by Q-learning and relevance feedback. It has therefore been used to observe the effect of learning and relevance assessments on ARACHNID performance. In a control run, neither learning nor relevance feedback takes place. In another run, the weights of each agent's neural net are adjusted by Q-learning after each move. In yet another run, after every ten documents visited, the ones in the relevant set are assessed by $\phi = 1$. In the last run, both learning and relevance feedback occur simultaneously. All runs have identical starting conditions,[5] so that the difference between the four outcomes is to be attributed to learning and/or relevance feedback. Figure 8 plots recall versus documents accessed. Learning causes an acceleration in the discovery of relevant documents. Relevance feedback improves recall by a similar degree, but of course only after the first relevant documents have been found. The combination of both learning and relevance feedback results in the best performance, because relevance feedback can take advantage of the earlier discoveries elicited by learning.

7 THE FUTURE

I have discussed preliminary evidence for the claim that adaptive algorithms based on local decisions are appropriate machine learning methods for information discovery in distributed environments. The proposed algorithm has allowed for the characterization of one environmental condition —semantic topology— sufficient for the success of the approach. Work is under way to achieve a better theoretical analysis of the algorithm, allowing one to predict its behavior under wider environmental conditions, and accounting for interactions among agents (Carson and Impagliazzo 1997). Furthermore, more data must be collected before one can become reasonably confident that the $R > G$ conjecture holds in real environments.

The naive representation allows for only limited adaptation —namely, adjusting how much agents trust link estimates and user feedback. Link estimates in this representation cannot capture important features of the environment, such as words. Consequently, more satisfying evidence for the notion of internalization is to be sought within the vector representation framework. An agent's adaptive perception of what features

[5]This includes the seed of the random number generator, which is not used by Q-learning nor by relevance feedback.

are relevant to the user can be internalized into its keyword vector and neural net to guide the search process. Agents will then evolve behaviors internalizing features locally correlated with relevance; if the user seeks information about sports, agents should give different weights to the word "rock" in pages about "rock climbing" or "rock music." Testing will tell whether the additional communication costs involved in the vector representation (due to the centralized feedback list) will afford adequate performance improvements.

Another issue to explore through the vector representation is how interactions between evolution and learning affect the information discovery process. It can be argued that agents can internalize environmental cues at different time and/or space scales. Some words will retain their discriminating value in more general contexts than others (e.g., "sport" vs. "rock"). Local selection allows agents to internalize features specific to local contexts, but only at time scales corresponding to an agent's reproductive cycle. Reinforcement learning is the natural extension of local adaptation to shorter time scales. While local selection integrates energy intake over time, instantaneous energy changes are a valuable short-term feedback signal.

Simulation results suggest that ARACHNID is an effective algorithm, compared with both non-adaptive search strategies and genetic algorithms with global selection. Preliminary experiments with the EB corpus not only support these results, but also provide encouraging evidence that both unsupervised reinforcement learning and supervised adaptation by relevance feedback can significantly enhance system performance. Further validation must be sought by more extensive experimentation on this as well as other real, unstructured Web environments.

Finally, as more secure languages and protocols for distributed computation over the Internet become available, one can expect to see servers allowing mobile agents to use up some CPU cycles in exchange for improved bandwidth. Agents would then analyze documents on the server, and only transfer relevant information back to the client. Distributed algorithms for intelligent agents, such as ARACHNID, may provide the machine learning techniques to make such developments feasible.

Acknowledgements

Work supported in part by NATO and Apple ATG fellowships. Encyclopaedia Britannica made the BCD corpus available. Thanks to Mark Land and two anonymous reviewers for helpful comments, Dave Demers for suggesting the acronym, and Rik Belew for constructive criticism and discussions.

References

Armstrong, R., D. Freitag, T. Joachims, and T. Mitchell (1995). Webwatcher: A learning apprentice for the world wide web. In *AAAI SSS Info. Gathering from Heterogeneous, Distrib. Envts.*

Balabanović, M. and Y. Shoham (1995). Learning information retrieval agents: Experiments with automated web browsing. In *AAAI SSS Info. Gathering from Heterogeneous, Distrib. Envts.*

Carson, T. and R. Impagliazzo (1997). Personal communication.

De Bra, P. and R. Post (1994). Information retrieval in the world wide web: Making client-based searching feasible. In *1st Intl. WWW Conf.*, Geneva.

EB. http://www.eb.com/.

Frakes, W. and R. Baeza-Yates (1992). *Information Retrieval: Data Structures and Algorithms.* Prentice-Hall.

Horn, J. (1996). GAs (with sharing) in search, optimization and machine learning. In *Proc. FOGA-4.*

Kraft, D., F. Petry, B. Buckles, and T. Sadasivan (1995). Applying genetic algorithms to information retrieval systems via relevance feedback. In P. Bosc and J. Kacprzyk (Eds.), *Fuzziness in Database Management Systems.* Physica-Verlag.

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

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.

Mahfoud, S. (1995). A comparison of parallel and sequential niching methods. In *Proc. 6th ICGA.*

Menczer, F. and R. Belew (1996a). Latent energy environments. In *Adaptive Individuals in Evolving Populations: Models and Algorithms.* Addison Wesley.

Menczer, F. and R. Belew (1996b). From complex environments to complex behaviors. *Adaptive Behavior 4*, 317–363.

Menczer, F., R. Belew, and W. Willuhn (1995). Artificial life applied to adaptive information agents. In *AAAI SSS Info. Gathering from Heterogeneous, Distrib. Envts.*

Moukas, A. (1996). Amalthaea: Information discovery and filtering using a multiagent evolving ecosystem. In *Proc. Conf. Practical Applications of Intelligent Agent Technology.*

Sheth, B. and P. Maes (1993). Evolving agents for personalized information filtering. In *9th IEEE Conf. on AI for Applications.*