Adaptive Information Agents in Distributed Textual Environments

Filippo Menczer and **Richard K. Belew**
Computer Science and Engineering Department
University of California, San Diego
La Jolla, CA 92093–0114, USA
{fil,rik}@cs.ucsd.edu

Abstract

Hypertext environments such as the Web are rich with both word and link cues that can be exploited by autonomous agents performing distributed tasks on behalf of the user. This paper characterizes such environments and identifies the features that are most useful and readily available. We describe the adaptive representation of an ecology of retrieval agents who attempt to capture important features of their surroundings, and base their behaviors upon them. We discuss how such a representation allows the agents to interact with the environments where they are situated. Agents can internalize words that are locally correlated with fitness, based on user feedback. They are shown to outperform non-adaptive search by an order of magnitude. Furthermore, each agent learns new strategies at local time and space scales, while the population evolves at a global scale.

1 Introduction

Imagine that you just submitted a query to your favorite digital library or search engine on the Web, and received a long list of "hits" as a response. At this point, you are probably going to browse manually through some of the links, giving higher precedence to those that appear more promising and backtracking when you feel that a branch is exhausted, until you are satisfied that further browsing will provide little further discovery of useful documents.

In many situations like this, users invest a large amount of time in the manual portion of the search. Yet the behavior of the browsing user in a case like this could be modeled with relative ease by an agent employing a "best-first-search" strategy, given an adequate evaluation function to predict the relevance of a page from looking at just a small portion of text surrounding a link to that page. There is, then, room and need for intelligent, adaptive machine learning methods to complement current search engine technology by automating such processes of personalized information discovery.

In fact, the paradigm of autonomous agents is receiving much attention because of the difficulty experienced by users in coping with the information overload caused by the increasing amount of information available on-line. 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 for the information management task [15].

The Web possesses many of the features making it an ideal target for adaptive agents. Many different agents in a population can adapt to the local characteristics of the different places where each is situated within the large, heterogeneous environment. Each agent individually, and the population as a whole, can adapt over time to the changes of the dynamic environment in which servers and documents are continuously being added, deleted, and moved. Finally, agents can execute in parallel on different server machines because the environment is distributed.

Several machine learning techniques have been suggested to produce effective information agents, yielding for example agents that perform look-ahead searches and provide suggestions to the user on the basis of reinforcement learning [9]. Techniques such as weighted keyword vector representations and relevance feedback, in conjunction with genetic algorithms and/or paradigms inspired by natural or economic systems, have been applied to information retrieval and filtering [21, 18, 1].

In these approaches, agents require some sort of supervision in order to adapt to the preferences of the user and/or to the external environment. The user may supervise the agents, for example, by allowing them to look over his shoulders, or by providing them with relevance feedback. In other systems, agents are completely unsupervised but cannot learn or adapt. For example, in the Fish Search algorithm [5] agents in a population of identical clones follow fixed, exhaustive search strategies.

We propose that agents should be able to perform *and* adapt in a completely autonomous fashion in the absence of supervision from the user, while making use of the user's feedback when this is available. In this paper we discuss the use of algorithms based on adaptive, intelligent, autonomous, distributed populations of agents making local decisions as a way to automate the on-line information search and discovery process in the Web or similar environments. The ARACHNID system was built to test the suitability of this approach [17]. In a recent paper we analyzed the high-level behavior of the algorithm, and its interaction with an abstraction of the environment, from a theoretical perspective [16]. Here we will focus on a representation of ARACHNID agents allowing for the exploitation of the wealth of statistical and topological cues present in distributed text

To appear in *Agents'98: Proceedings of the Second International Conference on Autonomous Agents*, Minneapolis, MN (May 10-13, 1998)

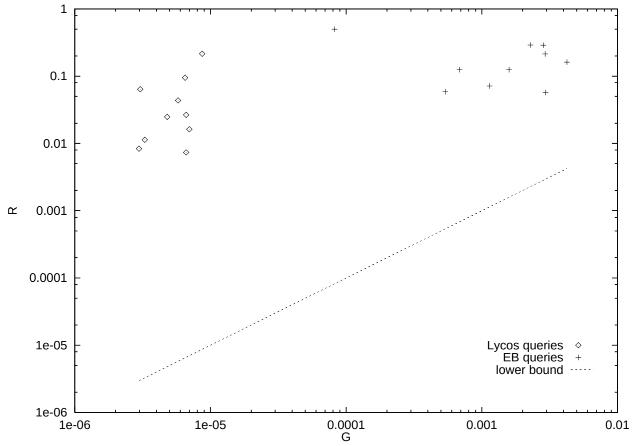

Figure 1: Link topology measures from two distributed text environments, the Web and *Britannica Online.* The R and G measures are based on retrieved sets by the *Lycos* and Britannica search engines, respectively [6, 14]. The lower bound is $R = G$. (Reprinted from [16].)

environments.

2 Text as environment

The agent paradigm can be reduced to a very simple algorithm in which the agent repeatedly receives input encoding state features and produces output encoding actions. What features are chosen to encode state and how the function mapping input to output is computed are the representation "details" by which any two agents will differ.

Distributed environments lend themselves to the agent paradigm. Examples of state are a machine, file system, or piece of data. Actions can entail reading a file, migrating a process, or issuing a request. In a distributed hypertext environment, a state is a document and its features are the words that make up the document. The action performed by an agent is the traversal of a link to access a new document from the same or a different server. Let us now consider both resources — word features and link structure — more closely.

Words are the principal asset in text collections, and virtually all information retrieval systems take advantage of words to describe and characterize documents, query, and concepts such as "relevance" or "aboutness." Distributed information environments — from small hypertext collections such as a user manual, to larger corpora such as a CD-ROM encyclopedia, and to immense networked environments such as the Web — are normally made of documents containing text, and therefore statistical features such as word frequencies remain central. Two documents are "similar" if they are about similar topics, and therefore would appear close to each other in a space whose topology is based on word features. This metric can be called *word topology* and is the reason why documents are usually represented as word vectors in information retrieval.

We want to argue that the structure imposed by information providers upon the organization of documents can give agents another useful resource. Links are manually inserted, positioned, and described by authors within documents. This superimposes a structure upon the collection of documents, for the very purpose of guiding the browsing user — or agent. We have conjectured that even in unstructured information spaces, authors effectively cluster documents about related topics by letting them point to each other [16]. To see this, call R the conditional probability that a document is relevant with respect to a query, given that it is pointed by a link from another document that is relevant with respect to the same query. And call G the generality of the query, i.e., the fraction of documents that are relevant. Then if $R > G$, it is more likely to hit a relevant document from another relevant document than from any random document. Hence the clustering property, that we call *link topology.* Figure 1 illustrates our finding that link topology is a real, measurable asset.

Note that the information encoded by the word and link topologies are quite distinct, and arguably complimentary. Links, constructed manually to point from one page to another, reflect an author's attempts to relate her writings to others'. Word topology is an epiphenomenal consequence of word vocabulary choices made by many authors, across many pages. The entire field of free text information retrieval is based on the statistical patterns reliably present in such vocabulary usage. By making our agents *perceptually* sensitive to word topology features and capable of *acting* by traversing link topology, we expect to find interesting relationships between the purposeful, manual linkage of Web authors and the words they use.

This paper is an attempt to begin to address some of the hard questions that arise when one considers statistical and structural features of distributed text environments. How can agents best take advantage of both word and link topologies? What representations would best enable agents to internalize features correlated with their task performance? How to generalize good strategies across users, information providers, networks, and time? How do word and link topologies constrain each other's cues from an agent's adaptation perspective? For example, if the user wants information about sports, agents browsing through pages about "rock climbing" and "rock music" should attribute different weights to the word "rock." Where an agent is situated in the environment provides it with *context* within which to analyze word meanings — a structured, situated approach to polisemy. Conversely, the words that surround links in a document provide an agent with valuable information to evaluate links and thus guide its path decisions — a statistical approach to action selection.

3 ARACHNID

ARACHNID[1] is a system for online, distributed information search. It is based on the simple algorithm sketched in Figure 2. The heuristic behavior by which an agent chooses what links to follow, while navigating from document to document, depends on the particular agent representation that one may pick. Representation is central to the specification of any ARACHNID implementation, and affects all parts of the algorithm.

A crucial part of ARACHNID is the use of relevance feedback. The user may assess the relevance of (some of) the documents visited up to a certain point by the algorithm. Such relevance assessments take place off-line, but they can alter the behaviors of agents on-line.

[1] The acronym, due to Dave Demers, stands for "Adaptive Retrieval Agents Choosing Heuristic Neighborhoods for Information Discovery." For up-to-date information on the status of the project, visit the ARACHNID website at `http://www.cs.ucsd.edu/~fil/agents`

```
0.  initialize population of agents, each with energy ε/2
while  there are alive agents:
  1.  pick random agent a
  2.  pick link to follow from document where a is situated
  3.  fetch new document D_a
  4.  E_a ← E_a − c(D_a) + E(D_a)
  5.  mark D_a as visited
  6.  optionally 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)
  8.  process optional relevance feedback from user
```

Figure 2: Pseudo-code for the ARACHNID algorithm.

3.1 Algorithmic details

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

In step (2), each 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, 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 move, i.e., continue to visit documents on behalf of the user. There is only one energy "source" in the system: agents are rewarded with energy if the visited documents appear to be relevant. The $E(D)$ function is used by an agent to evaluate the relevance of documents. If a document had previously been visited and assessed by the user, the user's assessment is used; if the document had not been visited before, its relevance must be estimated. Hence the need to mark visited documents (step (5)).

In practice, the marking mechanism is implemented via a cache, which also speeds up the process by avoiding duplicate transfers of documents. Caching documents is a form of communications, and one might observe that communication is a bottleneck for the performance of distributed algorithms. However, since network communication is by far the most expensive resource for the algorithm, the performance improvement warranted by the use of a cache should outweigh any degradation due to its constraints. The effects of cache size are not considered in the experiments described in this paper; the cache is large enough to contain all the visited documents.

There is also only one energy "sink" in the system: agents are charged energy costs for the use of network resources incurred by transferring documents. The cost function $c(D)$ should depend on used resources, for example transfer latency or document size. For simplicity we assume a constant cost for accessing the network, and a smaller constant cost for accessing the cache — so that chronic latecomers are discouraged.

Instantaneous changes of energy can be used, in step (6), as reward/penalty signals. This way, agents can adapt during their lifetime by reinforcement learning. While this adaptive process is subject to the noise of short-term fluctuations in energy, it allows an agent to modify its behavior based on prior experience, by learning to predict the best links to follow. This form of adaptation is appropriate to detect environmental cues and regularities at small space and time scales.

In step (7), an agent may be killed or be selected for reproduction. In the latter case offspring are mutated, providing the variation necessary for adapting agents by way of evolution. Since selection is based on the energy that has been accumulated over a long time (with respect to a single life cycle), evolution averages out short-term and short-range noise and can thus produce agent behaviors better adapted over larger space and time scales. Notice that, since the threshold ϵ is a constant, the decision whether an agent should reproduce is independent of other agents, and therefore ARACHNID minimizes communication among agent processes — a necessity for distributed algorithms.

Finally, in step (8), the user provides the system with relevance feedback. This process is optional and can take place off-line, without any direct interaction with the agents. The user may assess any visited document D as relevant or non-relevant, with feedback $\phi(D) = \pm 1$. All the words in the document are also assessed by updating a "feedback list" of encountered words. Each word in this list, k, is associated with an integer count ω_k that is initialized with 0 and updated each time any document is assessed by the user: $\forall k \in D$

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

The word feedback list is maintained to keep a global profile of which words are relevant to the user.

The output of the algorithm is a flux of links to document, ranked according to some relevance estimate —modulo relevance assessments by the user.

3.2 Architecture

Another view of the ARACHNID system is offered in Figure 3, illustrating the architecture of the system. The Web interface is based on the `libwww-perl` library [12]. The prototype is written in C and Perl and runs on UNIX and Macintosh platforms. Agents obey Internet etiquette by complying with the proposed standard for robot exclusion [10]. Agents also employ standard information retrieval tools such as a filter for noise words and a stemmer based on Porter's algorithm [8]. Finally, agents store an efficient representation of visited documents in a shared cache on the client machine. Each document is represented by a list of links and stemmed keywords. If the cache reaches its size limit, the LRU (least recently used) replacement strategy is used.

Figure 3 highlights the central dependence of the ARACHNID system on agent representation. In the remainder of the paper we will describe, analyze, and validate one representation of ARACHNID agents. A representation consists of the genotype, which 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 behaviors.

3.3 Agent representation

The first component of 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

[2]This list could be obtained, for example, by consulting a search engine.

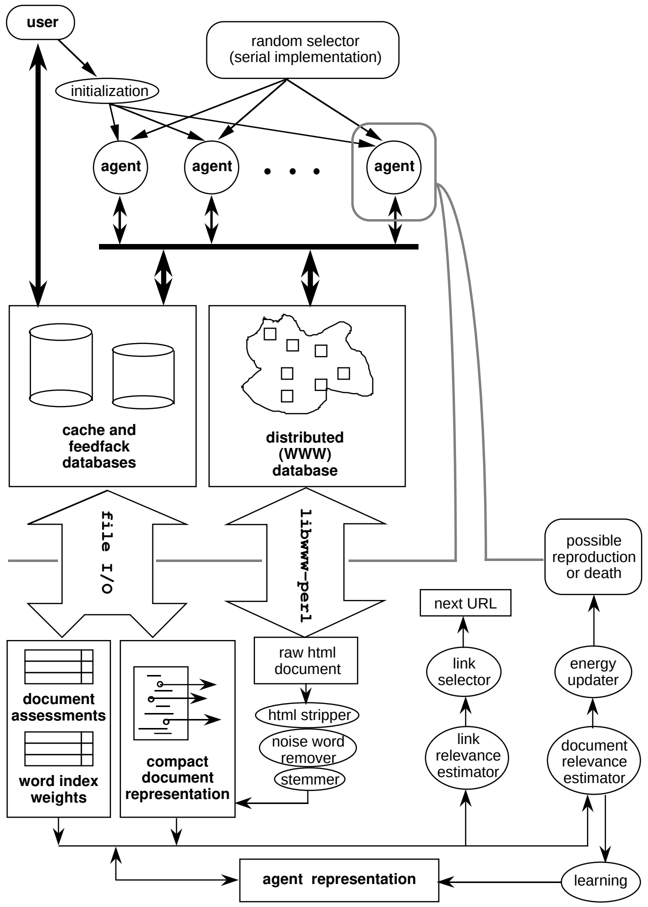

Figure 3: Architecture of the ARACHNID agent population.

contains about its outgoing links, and the latter represents the degree to which an agent trusts the user's relevance assessments. β and γ are initialized with uniform random distributions in intervals delimited by 0 and constants β_{max}, γ_{max}, respectively.

For the reasons outlined in Section 2, each agent's genotype also contains a list of keywords, initialized with the query terms. And since feed-forward neural nets are a general, versatile model of adaptive functions, we use them as a standard computation device. Therefore genotypes also comprise a vector or real-valued weights, initialized randomly with uniform distribution in a small interval centered around 0. The neural net has an input for each keyword in its genotype; it uses the hyperbolic tangent as its squashing function, with input and activation values in $[-1, +1]$, and a single real-valued output also in $[-1, +1]$. 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.

Let us now describe how the different parts of the system are implemented, based on the above representation. In step (2) of the algorithm, an agent performs action selection by first computing the relevance estimates for each 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.

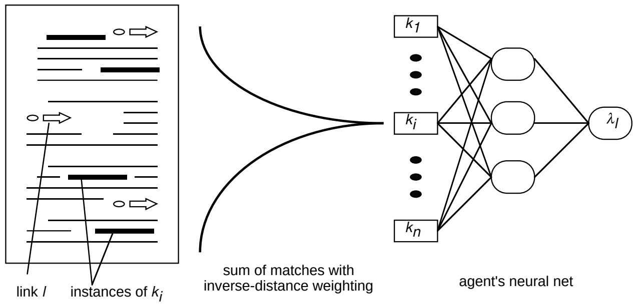

Figure 4: How an agent estimates each link from the current document. For each link in the document, each input of the neural net is computed by counting the document words matching the keyword corresponding to that input, with weights that decay with distance from the link.

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. In the experiments reported here, we use a distance weighting function which is biased towards keyword occurrences most close to the link in question. More specifically, for link l, the neural net receives, for each keyword k input:

$$in_{k,l} = \sum_i \frac{1}{dist(k_i, l)}$$

where k_i is the i-th occurrence of k in D_a and $dist(k_i, l)$ is a simple count of other, intervening links. The neural network then sums activity across all of its inputs, and produces output λ_l. The process is illustrated in Figure 4. Then, the agent uses a stochastic selector to pick a link with probability distribution:

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

After a link has been chosen and the corresponding new document has been visited, in step (4) of the algorithm the agent has to determine the corresponding energy gain and loss; both depend on whether or not the document had been visited previously. If the document is in the cache, and the user has assessed the relevance of the current document as $\phi(D_a)$, then agent a receives energy

$$E(D_a \in \text{cache}) = \gamma_a \phi(D_a)$$

If the user provided the system with relevance assessments, the word feedback list represents a profile of his interests that is both more current and more accurate than the original query; otherwise, the list simply reduces to the original query. This list is used to estimate the relevance of previously unvisited document, so that the corresponding energy intake can be computed:

$$E(D_a \notin \text{cache}) = \tanh\left(\sum_{k \in D_a} freq(k, D_a) \cdot I_k\right)$$

where $freq(k, D_a)$ is the frequency of term k in document D_a and I_k is the weight of term k based on relevance feedback. The latter is an extension of the $TF \cdot IDF$ (term frequency–inverse document frequency) index weighting scheme:

$$I_k = \omega_k \cdot \left[1 + \log\left(\frac{1}{C_k}\right)\right]$$

where C_k is the fraction of cache documents containing k. Such a weighting formula differs from more traditional $TF \cdot IDF$ schemes [22] in at least two respects. First, it is not aimed at weighting terms based on how well they describe documents, but rather on how well they correlate with relevance. Therefore it employs algebraic term frequencies to account for negative contributions from documents that contain the term but are anti-correlated with relevance. Second, it is computed online and therefore uses document frequencies based on the contents of the cache rather than the entire collection. The hyperbolic tangent is used to normalize energy intakes into the appropriate range $[-1, +1]$ — the same range as the corresponding neural nets prediction.

In step (6) of the algorithm, the agent can compare the relevance (assessed or estimated) of the current document with the estimate of the link that led to it. By using the connectionist version of Q-learning [13], the neural net can be adjusted to improve the accuracy of the link estimator. After agent a visits document D_a, $E(D_a)$ is used as reinforcement signal. The neural net's weights are updated by back-propagation of error, using teaching input:

$$E(D_a) + \mu \cdot \max_{l \in D_a} \lambda_l$$

where μ is a discount factor. In the current ARACHNID implementation, learned changes to the weights are "Lamarkian" in that they are inherited by offspring at reproduction [3].

At reproduction (step (7) of the algorithm), the offspring clone is mutated to provide the evolutionary process with the necessary power of exploration. 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. Weights are mutated by adding random noise:

$$w^i_{a'} \leftarrow U[w^i_a(1-\kappa_w), w^i_a(1+\kappa_w)]$$

The word vector is mutated by replacing the least useful (discriminating) term $\arg\min_{k \in a'}(|I_k|)$ with one expected to better describe the current document to the user, i.e., a term making the current document more similar to those assessed by the user. In order to keep any single keyword to take over the whole genotype, mutation is a stochastic process; a new term is selected with probability distribution

$$\Pr[k] = freq(k, D) \cdot |I_k|$$

where D is the document of birth. Learning will take care of adjusting the neural net weights to the new keyword.

4 Evaluation

Evaluation of a system like ARACHNID is complex and must be carried out at different levels. At a high level, we must be confident that the system performs significantly better than simpler strategies against which it can be compared. For the online information discovery task, the Web generally lacks the relevant sets that are necessary to measure *recall* (the fraction of relevant documents that have been visited) and *precision* (the fraction of visited documents that are relevant).

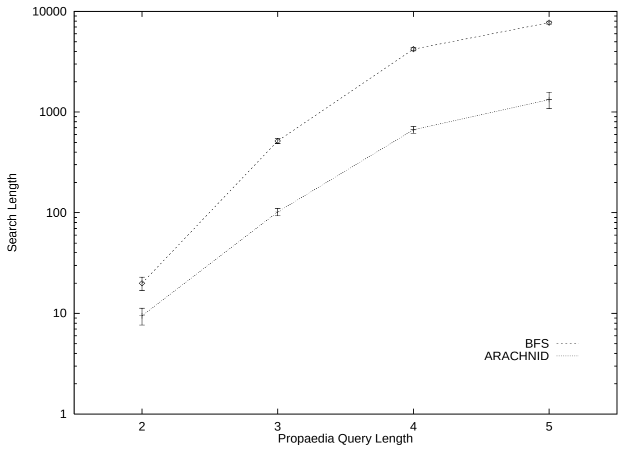

Figure 5: Search length of ARACHNID and BFS for queries at different depths in the "Human Society" Propaedia category of Encyclopaedia Britannica. Error bars are standard errors of means over same-depth queries.

We have therefore selected a subset of the Web as an information environment, the corpus of the *Encyclopaedia Britannica* [6]. The advantage is that we can make use of readily available relevant sets of articles associated with a large number of queries. Articles are in fact organized as the leaves of a hierarchical topical tree, called *Propaedia,* which is manually built and updated by skilled human editors. Therefore we can use the category title of any Propaedia node as a query, and the articles associated with the subtree rooted at that node as the relevant sets. In additions, each article also has hyperlinks to one or more Propaedia categories and possibly to other, semantically related articles, forming a graph. In the following evaluation experiments we limit the search to a subtree of the Propaedia with approximately 700 nodes, and the associated graph of approximately 11,000 articles.

4.1 Macro analysis

In the first experiment we measure one version of *search length,* a criterion combining recall and precision, defined as the number of non-relevant documents visited until the first relevant document is found [4]. This measure provides us with a "macro" analysis because the collective behavior of the agent population is observed as a whole, and averaged across a large number of queries. The search length of ARACHNID is compared with that of a simple non-adaptive algorithm, namely BFS (breadth-first-search). ARACHNID uses reinforcement learning, but for a fair comparison no user feedback is provided. The results are shown in Figure 5: the improvement of ARACHNID over BFS is as large as one order of magnitude.

At a lower level, we considered the behavior of the ARACHNID ecology in response to a single query. This way we analyzed the contributions of different parts of the system, namely reinforcement learning and relevance feedback. It was shown elsewhere [16] that Q-learning alone can significantly accelerate the discovery of relevant documents, shortening search length by a factor of 10 and improving precision

rank	*k*	*I*
1	COURT	0.034
2	SYSTEM	0.023
2	PARTI	0.023
2	GOVERN ⋆	0.023
5	POLIT ⋆	0.020
5	POWER	0.020
7	ADMINISTR	0.017
8	TH	0.016
9	OFFIC ⋆	0.015
9	CABINET	0.015
	...	
21	CONSTITUT	0.011
	...	
33	OFFICI	0.009
	...	
71	STRUCTUR ⋆	0.007
71	INSTITUT ⋆	0.007
	...	
83	BRANCH ⋆	0.006
	...	

Table 1: The top portion of the word feedback list after 350 document visits. The original query terms appear with stars.

accordingly. Likewise, the use of positive relevance assessments alone was shown to improve performance after the system locates the first relevant documents. Finally, the synergy between learning and user feedback results in the greatest improvement (an observed 4-fold increase in recall) because relevance feedback can take advantage of the earlier discoveries elicited by learning.

The population-wide word feedback list illustrates the information that is made available at a global, collective level by relevance feedback. Consider for example the following query: "Political institutions: The structure, branches, and offices of government." After the noise words have been removed and the remaining words have been stemmed, the query is reduced to `POLIT INSTITUT STRUCTUR BRANCH OFFIC GOVERN`. This is all the initial population knows about what the user is interested in. But after some of the visited documents are assessed by the user, her preferences become better defined. Consider the situation after 350 document have been visited by a number of different agents, assuming that all of the relevant documents have received positive feedback every 50 pages visited. For the above query, the word feedback list would look like Table 1. This list captures an image of what word features are best correlated with relevance. The term `COURT`, for example, appears to have the highest correlation with relevance even though it was not part of the query, while `GOVERN`, sharing second place with `SYSTEM` and `PARTI`, is the best among the original query terms. Other query words appear less useful in determining whether a document is relevant.

4.2 Micro analysis

A different type of evaluation can be obtained by a "micro" analysis of individual agents trying to answer the same user query, but at different times and situated in different areas of the information environment. Let us consider the same query as in the example above. A typical agent in the initial population (shown in Figure 6) has a word vector which coincides with the user query; its initial weights are random. After learning, the weights reflect the relative importance of the query terms; based on environmental cues perceived by this agent during its life, `POLIT` and `BRANCH` are perceived as the most and the least relevant terms, respectively.

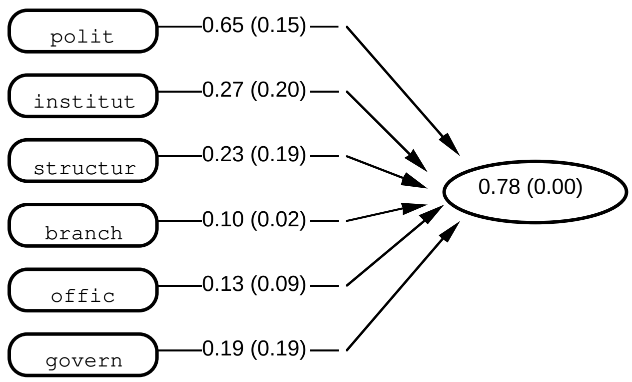

Figure 6: An agent from the initial population. The neural nets in this case are simple perceptrons, without hidden units. The numbers shown are the connection weights and output bias after learning (numbers in parentheses are before learning).

Two agents born after 350 document have been visited and assessed, shown in Figures 7 and 8 respectively, have internalized some of the global environmental cues (cf. Table 1) into their internal representations. Query words that are not very useful (e.g., `INSTITUT` and `BRANCH`) have disappeared from the keyword vectors through evolution, their places being taken by words that better correlate with user preferences (e.g., `SYSTEM` and `PARTI`).

Not all agents living at one time, however, are alike. They can differ substantially in their perceptions of what features are relevant to the user, based on where they are situated and on their evolutionary and life histories. This is a crucial consequence of our representation model, reflecting the fact that context refines the meaning of words and phrases [23]. For example, agents A and B have internalized some common features (`SYSTEM, GOVERN`) and some individual ones (e.g., `POLIT` vs. `OFFICI`). Their differences point to the role played by the documents visited by their ancestors, as well as those in which they are born, in shaping their representations. Agent A (Figure 7), situated on a page about "colonialism," will be looking for terms such as `POLIT` and `TH` that best discriminate between that document and others. Agent B (Figure 8) will focus more on the word `OFFICI`, better describing the "chancellor" document which gave rise to its birth.

We conclude this analysis by noting that even features shared by agents may be quantitatively different. For example, both agents have `GOVERN` in their genotypes; but while agent A attributes weight 0.19 to this term, the feature is duplicated in agent B and the combined weight is slightly larger (0.21). [3]

5 Conclusion

In this paper we have discussed how to represent agents that are to perform autonomous information tasks in distributed text environments. We have shown that a population of agents who both learn and evolve can adapt to a large, dis-

[3] The negative weight is due to a mutation; learning will eventually change its sign and further increase the combined weight.

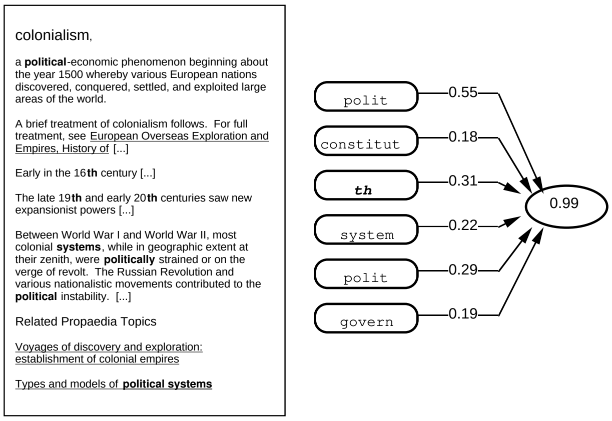

Figure 7: Agents A, born after 368 documents have been visited. In this and the following figure, the weights shown are those inherited at birth (plus mutations) and the keyword with emphasis is the result of a new mutation — in this case, `TH` is a term usually associated with historical events. On the left, part of the document where agent A was born, with keyword hits in bold. ©1997 Encyclopaedia Britannica, Inc.

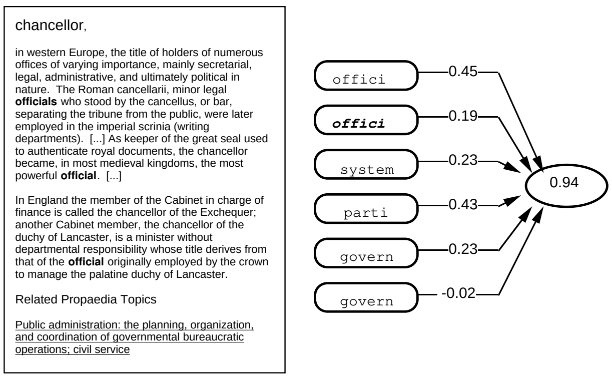

Figure 8: Agents B, born after 372 documents have been visited, and its birth document. Note that the new keyword, `OFFICI,` is distinct from the query word `OFFIC` (cf. Table 1). ©1997 Encyclopaedia Britannica, Inc.

tributed, heterogeneous, and dynamic environment at different time and space scales. We have also argued that such agents need to be able to internalize the salient features of their local environments, by detecting the clues that best correlate with their fitness.

We have described how agents in the ARACHNID system are able to achieve such goals by exploiting both statistical word regularities of text environments and the link structure of networked environments. For applications where both topologies are available, such as online information discovery on the Web, the ARACHNID approach seems promising.

Macro analysis results provide us with encouraging support for the approach: the ARACHNID population can locate relevant documents in a large distributed corpora 10 times faster than a non-adaptive BFS algorithm, using the same resources. Also, individual learning can interact with user relevance feedback, when this is available, to create a synergy that further boosts system performance.

A micro analysis enabled us to determine that single agents can in fact internalize important local features of the environment into their internal representation, while the collective ecology captures a more global snapshot of what features best discriminate between what is relevant to the user and what is not. Agent behaviors evolve with time, change over an agent's lifetime, and are different from agent to agent depending on what parts of the environment each has experienced.

We have argued that distributed agents are well suited to perform tasks in distributed environments. The current ARACHNID implementation is client-based, but the agents that we envision are mobile agents that execute on the CPUs of servers. It is not just information that must travel, as in current client-server protocols, but agent code as well. This way agents can use their intelligence to select the information to be sent back to the user's machine, improving network bandwidth. Secure languages and protocols are needed before information providers will welcome trusted autonomous agents to their servers; agent research is providing systems technology with the thrust that may enable such mobile agents developments feasible very soon [19].

Many different models of interaction among agents are also worth exploration. For example, research is under way into the issues of agents learning from other agents, agent collaboration, and agent communication languages. The first form of direct agent interaction that we are going to address is *recombination*. This is a natural part of the genetic algorithm that is used to model the evolution of ARACHNID populations. An agent at reproduction can crossover its internal representation with that of a nearby agent, perhaps one situated on the same server. The two can exchange their internal representations, internalizing experiences that are now relevant to each other because of their proximity.

Because we have been, in our early experiments, most interested in the behavior of agents on a carefully controlled and structured corpus (Encyclopedia Britannica), the full diversity we can reasonably expect from our agents as they interact with the real Web cannot be demonstrated. We have shown at least some divergence in the features that allow one agent to be successful within one topical area of the Encyclopedia and another, but the real purpose of open-ended evolutionary methods like those we propose is to adapt to the much wider variation found in Web media. We expect there to be roles for many different types of agents, sensitive to widely varying user demands, and effective at searching disparate corpora.

In fact, we anticipate enough varieties of "expertise" within agents that an entire *marketplace* will form for their various skills. The collaborative, social sharing of search expertise adapted on behalf of one user, then exploited by another, has been considered by others [2, 20, 7]. The extremely simple form of our agents (viz., small neural networks) allows their expertise to be exchanged in a particularly straight-forward fashion: If I like what your agent is doing for you, and you allow it, I can simply clone a new copy of this agent for my own use. As soon as I have interacted with my copy of this agent, it will quickly come to take on features of my idiosyncratic preferences.

But progress towards a true economic marketplace for the exchange of information-seeking expertise almost certainly will require the integration of a *middleman* role. Rather than seeking, digesting and evaluating "raw" information resources directly, such middleman information brokers must be made especially sensitive to the *value added* by the agent's processing. How much more will a user pay for the results of their agent's searches than they would for a simple list of all documents visited? Assessing the actual economic value of this service can be compared to moderated news groups and various "clipping services," where a person manually performs a very similar function. This type of editorial focusing is the ultimate goal of our information-seeking agents, but comparison against the currently available alternatives (e.g., the *Computists' Communique* newsletter [11] shows just how far we have to go.

6 Acknowledgments

We are grateful to Encyclopaedia Britannica, Inc. for allowing us to make use of their data in our experiments. Thanks to Charles Elkan, Russell Impagliazzo, Gary Cottrell, an anonymous reviewer, and the members of the Cognitive Computer Science Research Group at UCSD for helpful discussions and suggestions.

References

[1] M. Balabanović. An adaptive web page recommendation service. In *1st Intl. Conf. Autonomous Agents*, 1997.

[2] R. Belew. Adaptive information retrieval: Using a connectionist representation to retrieve and learn about documents. In *ACM Special Interest Group on Information Retrieval*, pages 11–20, Cambridge, MA, 1989.

[3] R. Belew and M. Mitchell, editors. *Adaptive Individuals in Evolving Populations: Models and Algorithms.* Santa Fe Institute Studies in the Sciences of Complexity. Addison Wesley, Reading, MA, 1996.

[4] W. Cooper. 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, 1968.

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

[6] Encyclopaedia britannica. http://www.eb.com/.

[7] Filmfinder. http://www.filmfinder.com/.

[8] W. Frakes and R. Baeza-Yates. *Information Retrieval: Data Structures and Algorithms.* Prentice-Hall, 1992.

[9] T. Joachims, D. Freitag, and T. Mitchell. Webwatcher: A tour guide for the world wide web. In *Proc. IJCAI*, 1997.

[10] M. Koster. The web robots pages. http://info.webcrawler.com/mak/projects/robots/robots.html, 1997.

[11] K. Laws. The computists' communique. http://www.computists.com/.

[12] http://www.ics.uci.edu/pub/websoft/libwww-perl.

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

[14] Lycos. http://www.lycos.com/.

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

[16] F. Menczer. Arachnid: Adaptive retrieval agents choosing heuristic neighborhoods for information discovery. In *Proc. 14th Intl. Conf. on Machine Learning*, 1997.

[17] F. Menczer and R. Belew. Arachnid software demo. 1st Intl. Conf. Autonomous Agents, 1997.

[18] A. Moukas and G. Zacharia. Evolving a multi-agent information filtering solution in amalthaea. In *1st Intl. Conf. Autonomous Agents*, 1997.

[19] D. Rus, R. Gray, and D. Kotz. Transportable information agents. In *1st Intl. Conf. Autonomous Agents*, 1997.

[20] U. Shardanand and P. Maes. Social information filtering: algorithms far automating "word of mouth". In *Proc. ACM Conference on Human Factors in Computing Systems*, pages 210–217, New York, NY, 1995. ACM.

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

[22] K. Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:111–121, 1972.

[23] A. Steier and R. Belew. Exporting phrases: A statistical analysis of topical language. In R. Casey and B. Croft, editors, *2nd Symposium on Document Analysis and Information Retrieval*, 1994.