Efficient Web Spidering with Reinforcement Learning

Jason Rennie[†]
jr6b@andrew.cmu.edu

[†]School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Andrew McCallum[‡†]
mccallum@justresearch.com

[‡]Just Research
4616 Henry Street
Pittsburgh, PA 15213

Abstract

Consider the task of exploring the Web in order to find pages of a particular kind or on a particular topic. This task arises in the construction of domain-specific search engines. A selective, directed web spider can be much more efficient than a spider that gathers new pages indiscriminantly. This paper argues that the creation of efficient web spiders is best framed and solved by reinforcement learning, a branch of machine learning that concerns itself with optimal sequential decision making. One strength of reinforcement learning is that it provides a formalism for measuring the utility of actions that give no immediate benefit, but do give benefit in the future.

Topic-specific spidering fits into the reinforcement learning framework because valuing hyperlinks with future reward is important. Experimental results on large collections of real web data show that a reinforcement learning spider finds relevant pages three times faster than a competing spider based on breadth first search. The results also show that our spider is not yet taking full advantage of future utility because of inaccuracies in our approximation for mapping hyperlinks to their expected future utility. Thus we believe that improving the accuracy of this mapping will increase performance even further, and we present ideas for doing so.

1 Introduction

Spiders are agents that explore the hyperlink graph of the Web, often for the purpose of finding documents with which to populate a search engine. Extensive spidering is the key to obtaining high coverage by the major Web search engines, such as AltaVista and HotBot. Since the goal of these general-purpose search engines is to provide search capabilities over the Web as a whole, they aim to find as many distinct web pages as possible. Such a goal lends itself to strategies like breadth-first search. If, on the other hand, the task is to find pages of a particular kind or on a particular topic, (as is the case for a domain-specific search engine), then an intelligent spider should try to avoid hyperlinks that lead to off-topic areas, and concentrate on links that lead to documents of interest.

This paper describes our research in directed web spidering. We argue that the creation of efficient web spiders is best framed and solved by reinforcement learning, a branch of machine learning that concerns itself with optimal sequential decision making. One strength of reinforcement learning is that it provides a formalism for measuring the utility of actions that give no immediate benefit, but do give benefit in the future. Reinforcement learning agents represent this delayed benefit by learning a mapping from each available action to a scalar value indicating the sum of future discounted rewards expected for executing that action. The "discount" makes later rewards less valuable than sooner rewards, thus encouraging expediency.

In our current instantiation of a reinforcement learning spider, we learn a mapping from text in the neighborhood of a hyperlink to the future discounted count of relevant pages available as a result of retrieving the page on the other side of the hyperlink. The mapping from text to a scalar is performed casting regression as text classification. Specifically, we discretize the scalar values into a finite number of bins, and use naive Bayes to classify the text into a finite number of classes; the value assigned to a particular hyperlink is then a weighted average of the values of the top-ranked bins.

Our research in efficient spidering is part of a larger project that has created Cora, a domain-specific search engine containing computer science research papers (McCallum *et al.* 1999). We have spidered computer science departments and labs, so far finding over 50,000 research papers in postscript format. Each of these papers is transformed into ASCII, then run through a specially-trained hidden Markov model to automatically find the title, authors, abstract, references, etc. Forward and backward references between papers are resolved. Finally, all this information is made available in a searchable, public web interface at *www.cora.justresearch.com*.

In Cora, efficient spidering is a major concern. The majority of the pages in many computer science department web sites do not contain links to research papers, but instead are about courses, homework, schedules and admissions information. Avoiding whole branches and neighborhoods of departmental web graphs can significantly improve efficiency and increase the number of research papers found given a finite amount of crawling time.

We report on a series of experiments using the complete set of web pages and hyperlinks from four university computer science departments—a total of more than 53,000 pages and 592,000 hyperlinks. The specific spidering task, based on the Cora domain-specific search engine, is to find research papers as efficiently as possible. The results show that a reinforcement learning spider finds half of the available research papers with only one-third the number of page retrievals as a competing spider based on breadth-first search.

However, the experiments indicate that our spider is not yet taking full advantage of representing delayed reward. A reinforcement learning spider that uses delayed reward performs slightly worse overall than a spider that uses immediate reward only—although it performs dramatically better in the beginning. Our analysis indicates that this is caused by decreased text classification accuracy in the delayed-reward case, where the number of discretized bins is larger and the training data is more noisy. Improving classification accuracy should allow the delayed-reward spider to dominate the immediate-reward spider everywhere, (not just at the beginning), and thus out-perform the breadth-first-search spider by even more than the current factor of three. The last sections present ideas for increasing this accuracy.

2 Reinforcement Learning

Here, we give a short description of reinforcement learning, explain why it is an appropriate framework for spidering and describe the approach we use to apply reinforcement learning to spidering.

In machine learning, the term "reinforcement learning" refers to a framework for learning optimal decision making from rewards or punishment (Kaelbling, Littman, & Moore 1996). Upon taking an action, the learner is never told the correct action for a particular state, but is simply told how good or bad the selected action was, expressed in the form of a scalar "reward."

A task is defined by a set of states, $s \in \mathcal{S}$, a set of actions, $a \in \mathcal{A}$, a state-action transition function, $T : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$, and a reward function, $R : \mathcal{S} \times \mathcal{A} \rightarrow \Re$. At each time step, the learner (also called the *agent*) selects an action, and then as a result is given a reward and its new state. The goal of reinforcement learning is to learn a *policy*, a mapping from states to actions, $\pi : \mathcal{S} \rightarrow \mathcal{A}$, that maximizes the sum of its reward over time. The most common formulation of "reward over time" is a discounted sum of rewards into an infinite future. A *discount factor*, γ, $0 \leq \gamma < 1$, expresses "inflation," making sooner rewards more valuable than later rewards. In addition to preventing the sum of future rewards from blowing up to infinity, this also encourages the agent to solve the task in as few steps as possible. Accordingly, when following policy π, we can define the *value* of each state to be:

$$V^{\pi}(s) = \sum_{t=0}^{\infty} \gamma^t r_t, \tag{1}$$

where r_t is the reward received t time steps after starting in state s. The optimal policy, written $\pi^{\star}$, is the one that maximizes the value, $V^{\pi}(s)$, for all states s.

In order to learn the optimal policy, we learn its value function, $V^{\star}$, and its more specific correlate, called Q. Let $Q^{\star}(s,a)$ be the value of selecting action a from state s, and thereafter following the optimal policy. This is expressed as:

$$Q^{\star}(s,a) = R(s,a) + p\gamma V^{\star}(T(s,a)). \tag{2}$$

We can now define the optimal policy in terms of Q by selecting from each state the action with the highest expected future reward: $\pi^{\star}(s) = \arg\max_a Q^{\star}(s,a)$. The seminal work by Bellman (1957) shows that the optimal policy can be straightforwardly found by dynamic programming.

2.1 Spidering as Reinforcement Learning

As an aid to understanding how reinforcement learning relates to spidering, consider the task of a mouse exploring a maze to find several pieces of cheese. The agent's actions are moving among the grid squares of the maze. The agent receives a reward for finding each piece of cheese. The state is the position of the mouse and the locations of the cheese pieces remaining to be consumed (since the cheese can only be consumed and provide reward once). Note that the agent only receives immediate reward for finding a maze square containing cheese, but that in order to act optimally it must choose actions considering future rewards as well.

In the spidering task, the on-topic documents are immediate rewards, like the pieces of cheese. The actions are following a particular hyperlink. The state is the locations of the on-topic documents remaining to be consumed. The state does not include the current "position" of the agent since a crawler can retrieve any page for which it knows the URL. The number of actions is large and dynamic, in that it depends on which documents the spider has visited so far.

The key features of topic-specific spidering that make reinforcement learning the proper framework for defining the optimal solution are: (1) performance is measured in terms of reward over time, and (2) the environment presents situations with delayed reward.

2.2 Practical Approximations

The problem now is how to apply reinforcement learning to spidering in such a way that it can be practically solved. Unfortunately, the state space is huge: two to the power of the number of on-topic documents on the Web. The action space is also large: the number of unique URLs with incoming links on the Web. Thus we need to make some simplifying assumptions in order to make the problem tractable and to aid generalization. Note, however, that by defining the exact solution in terms of the optimal policy, and making our assumptions explicit, we will better understand what inaccuracies we have introduced, and how to select areas of future work that will improve performance further. The assumptions we choose initially are the following two: (1) we assume that the state is independent of which on-topic documents have already been consumed; that is, we collapse all states into one, and (2) we assume that the relevant distinctions between the actions can be captured by the words in the neighborhood of the hyperlink corresponding to each action.

With these assumptions, our Q function becomes a mapping from a 'bag-of-words" to a scalar (sum of future reward). Learning to perform efficient spidering then involves only two remaining sub-problems: (1) assigning appropriate Q values to each hyperlink in the training set, and (2) learning a mapping from text to Q values using the training data. Assigning Q values to our training set entails striking a balance between future and immediate reward and setting a valuation on various types of future reward. We perform the mapping from text to Q value by casting this regression problem as classification (Torgo & Gama 1997); we discretize hyperlink Q values, train a naive Bayes classifier on the corresponding neighborhood text and compute the Q value of an unknown hyperlink by computing a weighted average of classification scores.

2.3 Value Criterion

In order to train an efficient spider using reinforcement learning, we must first create a mapping from hyperlinks to Q values. As we have assumed a single state to simplify our system, we will calculate Q values with all on-topic documents in place. In our mouse and cheese scenario, this is the same as calculating Q values before any of the cheese has been eaten.

The simplest mapping gives a value of 1 to every hyperlink that points directly to a research paper and a value of 0 to all other hyperlinks. This is equivalent to using the reinforcement learning framework with $\gamma = 0$. In many domains, this is not a useful optimization criteria because the attributes associated with an immediate reward action are distinct from those that identify future reward actions. In the domain of text, this is not necessarily the case. Text that is used to describe pointers to research papers may also be used to point

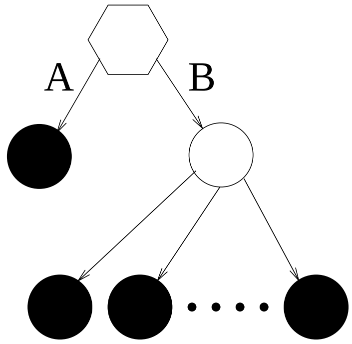

Figure 1: A representation of spidering space where arrows are hyperlinks and circles are web documents. The spider is represented as a hexagon. A and B are hyperlinks that the spider may elect to follow; filled-in circles denote reward. When a spider is given the choice between an action that provides immediate reward and one that provides future reward, the spider always achieves the maximum discounted reward by choosing the immediate reward first.

toward repositories of such documents.

A slightly more involved criteria is one that values an action, A, according to the rewards achievable from the web page that is retrieved by performing A. This is identical to using reinforcement learning with $\gamma > 0$. For each hyperlink, we use dynamic programming to calculate a discounted sum over rewards that can be achieved using the hyperlinks stemming from the corresponding web page.

A subtle issue comes to bear because of the nature of spidering. In general reinforcement learning problems, a future reward action may be more beneficial than one that provides immediate reward. This is often the case when reward varies in magnitude or when reward is highly dependent on the state of the system. However, for the problem of spidering, this is not the case. In fact, given the choice between an action that elicits immediate reward and one that yields future reward, it is *always* better for the spider to choose the immediate reward action.

An example, depicted in Figure 1, gives good intuition for why future rewards never provide greater benefit. Imagine the case where a spider has two actions that it can take, A and B. Action A retrieves a research paper and yields a reward of 1. Action B retrieves a web page that includes hyperlinks to 1000 research papers, each of which provides a reward of 1 when retrieved. When the spider performs A before B, the sequence of rewards is as follows:

101111111...1

When the spider performs B before A, the first two rewards are reversed:

011111111...1

The only difference between the two is the time at which the first reward is received. It is beneficial to achieve rewards as quickly as possible, hence it is preferable to perform action A before action B. As the reinforcement learning framework allows for the possibility of future rewards being valued more highly than immediate rewards, some of our experiments will calculate Q values so that no future reward hyperlink receives a value higher than an immediate reward hyperlink.

2.4 Neighborhood Text

With a valuation for the set of known hyperlinks, it is necessary to specify features that can be used to compare these to unknown hyperlinks. For this, we choose the text found in the neighborhood of a hyperlink. Such text is commonly used to describe the content of the page connected to the hyperlink. An example of this, which is commonly found in computer science faculty home pages, is a research paper bibliography that contains a hyperlink to a research paper:

> "Using the Future to Sort Out the Present: Rankprop and Multitask Learning for Medical Risk Analysis," (postscript) R. Caruana, S. Baluja, and T. Mitchell, Neural Information Processing 7, December 1995.

Here, the anchor text is the hyperlink and the text of the bibliography provides a good description of the document one would find by following that hyperlink. In the experiments presented in this paper, we associate with each hyperlink two sets of words, 1) the anchor text and tokens from the URL and 2) the full text of the web page where the hyperlink is located. This associates a significant portion of text with each hyperlink and allows each hyperlink to be identified uniquely. Experiments not presented in this paper show that using a more restricted set of neighborhood text does not enhance spidering performance.

2.5 Naive Bayes

We use the text classification algorithm naive Bayes to perform the mapping from text to Q values. Here, we describe that algorithm in detail.

Naive Bayes approaches the task of text classification from a Bayesian learning framework. It assumes that text data is generated by a parametric model, and uses training data to calculate estimates of the model parameters. Equipped with these estimates, it classifies new test documents using Bayes' rule to turn the generative model around and calculate the posterior probability that each class would have generated the test document in question.

The classifier parameterizes each class separately with a document frequency, and also word frequencies. Each class, c_j, has a document frequency relative to all other classes, written $\mathrm{P}(c_j)$. Each class is modeled by a multinomial over words. That is, for every word, w_t, in the vocabulary, V, $\mathrm{P}(w_t|c_j)$ indicates the frequency that the classifier expects word w_t to occur in documents in class c_j.

We represent a document, d_i, as an unordered collection of its words. To classify a new document with this model, we make the naive Bayes assumption: the words in the document occur independently of each other given the class of the document, (and furthermore independently of position). Using this assumption, classification becomes straightforward. We calculate the probability of each class, given the evidence of the document, $\mathrm{P}(c_j|d_i)$, and select the class for which this expression is the maximum. We denote $w_{d_{ik}}$ to be the kth word in document d_i. We expand $\mathrm{P}(c_j|d_i)$ with an application of Bayes' rule, and then make use of the word independence assumption:

$$\begin{aligned} \mathrm{P}(c_j|d_i) &\propto \mathrm{P}(c_j)\mathrm{P}(d_i|c_j) \\ &\propto \mathrm{P}(c_j)\prod_{k=1}^{|d_i|}\mathrm{P}(w_{d_{ik}}|c_j). \end{aligned} \tag{3}$$

Learning these parameters ($\mathrm{P}(c_j)$ and $\mathrm{P}(w_t|c_j)$) for classification is accomplished using a set of labeled training documents, $\mathcal{D}$. To estimate the word probability parameters, $\mathrm{P}(w_t|c_j)$, we count over all word occurrences for class c_j the frequency that w_t occurs in documents from that class. We supplement this with Laplace 'smoothing' that primes each estimate with a count of one to avoid probabilities of zero. Define $N(w_t, d_i)$ to be the count of the number of times word w_t occurs in document d_i, and define $\mathrm{P}(c_j|d_i) \in \{0, 1\}$, as given by the document's class label. Then, the estimate of the probability of word w_t in class c_j is:

$$\mathrm{P}(w_t|c_j) = \frac{1 + \sum_{d_i\in\mathcal{D}} N(w_t, d_i)\mathrm{P}(c_j|d_i)}{|V| + \sum_{s=1}^{|V|}\sum_{d_i\in\mathcal{D}} N(w_s, d_i)\mathrm{P}(c_j|d_i)}. \tag{4}$$

The class frequency parameters are set in the same way, where $|\mathcal{C}|$ indicates the number of classes:

$$\mathrm{P}(c_j) = \frac{1 + \sum_{d_i\in\mathcal{D}} \mathrm{P}(c_j|d_i)}{|\mathcal{C}| + |\mathcal{D}|}. \tag{5}$$

Empirically, when given a large number of training documents, naive Bayes does a good job of classifying text documents (Lewis 1998). More complete presentations of naive Bayes for text classification are provided by Mitchell (1997) and Nigam *et al.* (1999).

2.6 Regression as Classification

Now that we have specified methods for assigning known hyperlinks Q values, choosing features from each hyperlink for comparison and classifying such features into a discrete set of classes, we can now describe the full process of regression we use to map unknown hyperlinks to Q values.

We perform the mapping by casting this regression problem as classification (Torgo & Gama 1997). We discretize the discounted sum of future reward values of our training data, place the hyperlinks into the bin corresponding to their Q values as calculated above, and use the hyperlinks' text as training data for a naive Bayes text classifier.

In order to determine the value of an unknown hyperlink, we calculate the probabilistic class membership for each bin and compute a weighted average of the bins' average Q values. The probabilistic class membership values are generated by naive Bayes and are used as the weights in averaging Q values. Thus, the problem of learning a mapping is reduced to the problem of partitioning the set of known hyperlinks and assigning each of those classes an average Q value.

3 Experimental Setup

This section describes the experimental methodology used to evaluate our spiders' ability to locate research papers within computer science department web sites.

3.1 Data

In August 1998 we completely mapped the documents and hyperlinks of the web sites of computer science departments at Brown University, Cornell University, University of Pittsburgh and University of Texas at Austin. They include 53,012 web pages and postscript files and 592,216 hyperlinks. The target pages (for which a reward of 1 is given) are computer science research papers. We identify research papers by converting postscript files to text and searching for an abstract or introduction section in addition to a references or bibliography section. An experiment using a random sampling of 200 postscript files shows that this algorithm has approximately 95% precision. A total of 2,263 postscript research papers were identified within the webs of the four computer science departments.

3.2 Experiments

For each spider, we perform a series of four test/train splits in which the data from three universities is used to train a spider that is then tested on the fourth. We create three spiders based on various forms of reinforcement learning and compare their performance. In order to create a baseline for these experiments, we also perform experiments with a spider that uses a FIFO action queue; it follows hyperlinks in the order it finds them. We call this the breadth-first spider.

The first spider is described in section 2.3 as the degenerate form of a reinforcement learning spider, that is, the case where γ is set to zero. It partitions documents into two classes for the regression task. One class is those hyperlinks that point to research papers; the other class includes all remaining hyperlinks. We call this the immediate spider.

The second spider is our first attempt to make use of future reward. It is also described in section 2.3 and is an application of the reinforcement learning framework with a non-zero γ value. We call this the future spider.

We introduce a third spider that attempts to remedy some of the problems inherent in the immediate and future spiders. Such a spider must 1) rank research paper hyperlinks above all others and 2) appropriately prioritize other hyperlinks according to their future reward. The immediate spider only singles out research paper hyperlinks and does not explicitly rank other hyperlinks according to their future reward. The future spider does not satisfy #1 because it assigns a Q value greater than 1 to some non-immediate reward hyperlinks. However, it does rank future reward hyperlinks and thus encapsulates #2.

Max Reward	Distance to nearest reward				
	0	1–2	3–4	5+	∞
0					0.000
1–100	1.000	0.250	0.063	0.016	
101+		0.500	0.125	0.031	

Table 1: Q value assignment for the distance spider. Note that some boxes are not assigned values because documents for those cases do not exist. $Q = 1.0$ is the class of immediate reward links. $Q = 0.0$ is the class of hyperlinks that lead to no reward.

To describe a mapping from hyperlinks to Q values that exhibits the properties of the optimal spider, we construct by hand a set of hyperlink classes (displayed in Table 1) based on distance to the nearest reward and the maximum attainable reward. In order to avoid assigning Q values larger than 1 to future reward hyperlinks, we do not calculate Q to be a discounted sum of all future reward, but rather the discounted value of the nearest reward, modified slightly according to the total amount of future reward. We call this the distance spider.

3.3 Evaluation Metrics

We use two methods for evaluating the performance of our spider. The first is the number of web pages retrieved before half of the research papers are retrieved. This provides a simple, intuitive evaluation of performance, but is not robust. A spider may be able to find half of the research papers almost immediately and then have incredible difficulty in finding any more. A preferred method of evaluation is the calculation of a sum of rewards where each reward is discounted by one minus the percent of web pages yet to be retrieved. This has the effect of calculating the area under the curve when research papers retrieved is plotted against the number of web pages retrieved. When presenting these "integral" scores, we normalize scores so that a spider that retrieves all research papers before its first action (an impossible scenario) achieves a score of 100.

4 Results

In our experiments, we provide evidence that reinforcement learning-based spiders perform significantly better than breadth-first, reducing the amount of required effort by as much as two-thirds. We also show that a spider that does not distinguish between future and zero reward hyperlinks can perform well over the lifetime of the spider. However, we note that such a spider does not perform well at the beginning stages and is outperformed by a spider that distinguishes between different amounts of future reward. Additionally, we show that a spider that inappropriately weights future reward hyperlinks above immediate reward hyperlinks is outperformed by one that values immediate reward hyperlinks most highly.

We present empirical findings from our four university data set; each experiment is an average across results from the four universities. We run our spider on each university web, using the web from the other three universities as training data.

Our first graph, shown here in Figure 2, displays the performance of the distance, future and breadth-first spiders. As can be seen, both distance and future perform significantly better than breadth-first, showing that there is significant value in using a reinforcement learning framework such as we have constructed. Here, distance must follow only 9.0% of hyperlinks before locating half of the research papers while breadth-first must traverse 27.3%. This is a 67% reduction in the amount of work needed to locate a significant number of research papers.

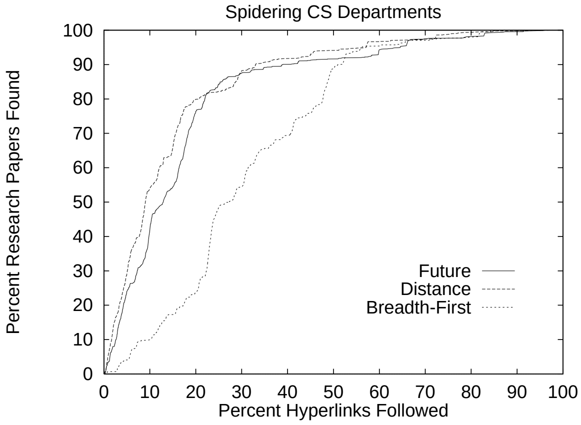

Figure 2: The performance of the distance, future and breadth-first spiders, averaged over four test/train splits with data from four departments. The vertical axis shows the percentage of on-topic documents found, while the horizontal axis shows the percentage of hyperlinks followed thus far into the spider's exploration. Note that distance outperforms future and both reinforcement learning spiders find target documents significantly faster than breadth-first.

This graph is also useful for comparing the performance of the distance and future spiders. As can be seen, the distance spider outperforms the future spider at nearly every step of the way. Our integral metric confirms this improved performance, rating distance as 85.2 and future as 82.6. The distance spider comes 15% closer to the upper bound on integral scores. Both spiders out-perform the breadth-first score of 69.4.

The fact that distance performs better than future is significant because the primary difference between the two spiders their handling of immediate reward hyperlinks. As was mentioned earlier, immediate reward hyperlinks are always more beneficial than future reward hyperlinks. Distance explicitly includes this fact in its model by assigning the highest Q values to immediate reward hyperlinks.

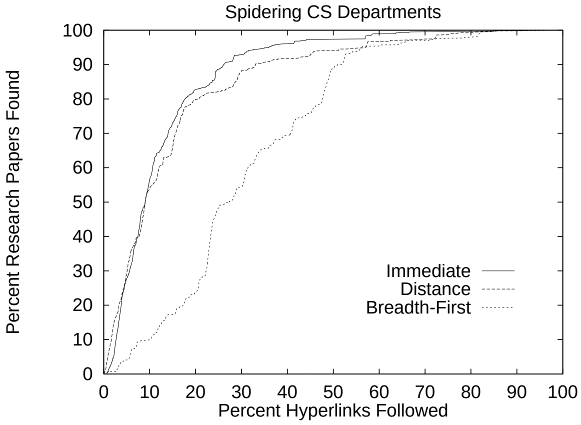

Figure 3: The performance of the immediate and distance spiders. Immediate has a higher integral score, but distance has better performance through some of the early stages of spidering.

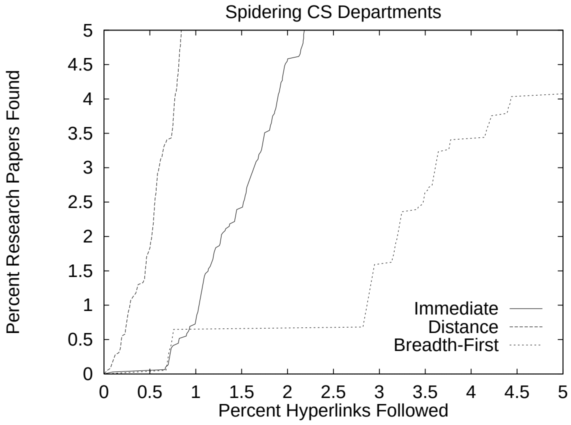

Figure 4: A close-up view of early performance of the immediate and distance spiders. Note the scaled axes. Distance immediately begins retrieving research papers while immediate must wade through nearly 100 hyperlinks before finding its second research paper.

Our second graph, shown in Figure 3, again displays the performance of breadth-first and distance, replacing future with the immediate spider. Compared with breadth-first, immediate clearly performs much better, reducing by two-thirds the amount of work necessary to retrieve half of the available research papers. Immediate also performs better than the distance spider, achieving an integral score of 87.4, 15% closer to the upper bound than the distance score of 85.2.

Note that immediate is able to achieve this excellent performance without an explicit stratification of future reward hyperlinks. This is somewhat surprising because it does not explicitly account for one of the necessary properties of the optimal spider. However, this may be due to the importance of immediate reward in the later stages of spidering. In classification experiments, we find that the classifier used for the immediate spider achieves 94% recall in classifying immediate reward hyperlinks. That is to say that of the hyperlinks that are actually immediate reward hyperlinks, the classifier for the immediate spider labels 94% of those as being immediate reward hyperlinks. After 10% of the hyperlinks have been followed, most hyperlinks have been found and the task is reduced to one of identifying immediate reward hyperlinks.

While immediate performs better than distance in aggregate, it is not true that immediate has collected more research papers than distance through the entire spidering run. In fact, in the early stages of spidering, the performance of the distance spider is significantly better. In Figure 4, we show the performance of immediate and distance through their retrieval of the first 28, or 5% of research papers. Compared to immediate, distance follows 61%, or 177 fewer hyperlinks before retrieving the first 5% of research papers. Even more significant is the fact that immediate retrieves only a single paper before having followed 88, or 0.67% of the available hyperlinks. This exhibits an inadequacy of the immediate spider and shows that immediate is not fully able to judge the value of future reward hyperlinks, even when faced with cases where it is possible to judge those values using only textual features.

5 Discussion and Future Work

Our results section provides strong evidence that for the task of locating research papers within computer science department webs, reinforcement learning using text-based features can yield us a substantial win over a more rudimentary method such as breadth-first search. We also give evidence that there is value in learning the features of future reward hyperlinks. Even though the immediate spider performs well without taking regard for the features found in future reward hyperlinks, it is not able to efficiently find research papers from the beginning. Here we give two reasons why the immediate spider performs well and propose ideas for improving spidering performance.

In order to gain some understanding of why the immediate spider performs so well, we examine the words which are most predictive of the immediate reward class for the immediate spider.

As one can see in Table 2, many of the words are used to uniquely describe Computer Science research papers. It is common for a web page containing a link to a research paper to contain a citation or other descriptive text for that research paper. Hence the reason for such features. It is interesting to note that some of these words, such as "technical," "report" and "papers" are commonly used by computer science departments and professors to indicate a hyperlink that points to a collection of research papers. Thus immediate should rate highly some future reward hyperlinks.

Immediate performs best toward the end of a run, most likely because of its ability to identify research papers. Near the end of a run, most of the immediate reward hyperlinks are known and the spidering task is reduced to one of identifying such hyperlinks. As was mentioned earlier, immediate performs very well at this task, achieving a 94% recall rate.

While immediate performs well at the end of a spidering run, distance performs best toward the beginning—when identification of high value future reward links is most important. As each of these two spiders perform best at two different tasks, an area for future work is experimenting with combinations of these spiders. If it is possible to select future reward links as well as distance, while retaining immediate's ability to identify immediate rewards, then it may be possible to construct a spider that outperforms both of them.

0.0238	proceedings	0.0072	report
0.0166	pp	0.0069	science
0.0128	postscript	0.0069	technical
0.0120	computer	0.0068	tr
0.0120	international	0.0065	intelligence
0.0117	conference	0.0063	boyer
0.0115	acm	0.0061	workshop
0.0105	proc	0.0061	springer
0.0104	symposium	0.0061	aaai
0.0098	systems	0.0061	artificial
0.0083	paper	0.0061	algorithms
0.0078	computing	0.0060	reasoning
0.0076	logic	0.0060	lifschitz
0.0072	learning	0.0060	papers
0.0072	ieee	0.0058	vol

Table 2: Most predictive words in class model used for immediate for class of documents pertaining to hyperlinks that point to research papers. Scores shown are weighted log likelihood ratios (Cover & Thomas 1991). Note that these words are, intuitively, words that are likely to be used to describe hyperlinks that point directly to Computer Science research papers.

One of the assumptions that we made about our application of reinforcement learning to spidering is that the state of a spider is not important for identifying the value of an action. While our assumption is not true, it is difficult to compare two different states,

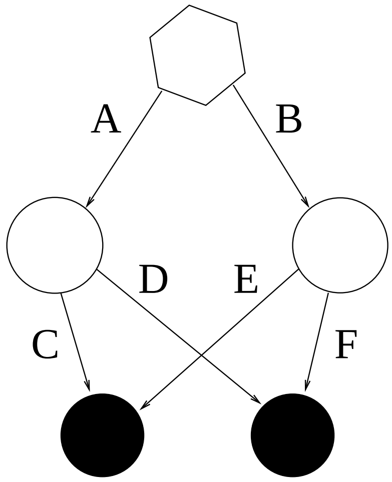

Figure 5: A representation of spidering space where arrows are hyperlinks and circles are web documents. Hyperlinks are labeled with letters of the alphabet; the spider is denoted as a hexagon; circles denote web documents; filled-in circles denote reward. A and B are hyperlinks that the spider may elect to follow. The actions selected by the spider may change the value of some hyperlinks.

especially when much of the search space is unknown. Figure 5 shows an example of such a situation.

Here, the spider may elect to follow hyperlinks A or B. Initially, both A and B have future reward as their children point to research papers. However, if the spider were to select actions A, C and D, in that order, then B would no longer be of value. Of course, one cannot use text features to account for these changing states. However, it is true that there exist certain hyperlinks near reward that are worthless because discovering those hyperlinks necessitates traversing the page to which the hyperlink points. We may be able increase spidering performance by rating these links according to the state of the spider once it finds such a hyperlink.

In selecting a feature set to represent a hyperlink, we ignored much of the information available to us and restricted ourselves to using unstructured, plain text. We believe that performance increases may be found using features of a hyperlink that we have neglected, such as HTML markup and the graph structure of the web around the hyperlink. Such additions are likely to improve classification accuracy and thus increase performance of the spider.

6 Related Work

Several other studies have explored efficient information gathering from the Web.

ARACHNID (Menczer 1997) is a system that uses a collection of agents for finding information on the Web. Each agent competes for limited computation resources, procreating and mutating proportionally to its success in finding relevant documents. Information gathering experiments are demonstrated on the Encyclopedia Britannica Propaidia tree and synthetic data. By contrast, our spider has roots in optimal decision theory, and searches unstructured web pages from the real Web.

WebWatcher (Joachims, Freitag, & Mitchell 1997) is a browsing assistant that helps a user find information by recommending which hyperlinks to choose. It thus restricts its action space to only hyperlinks from the current page. WebWatcher uses reinforcement learning to learn the value of each word on a hyperlink. Our work is not user-centric and strives to find a method for learning an optimal decision policy for locating relevant documents when hyperlink selection is unlimited.

Laser (Boyan, Freitag, & Joachims 1996) is a search engine that automatically optimizes a number of parameters to achieve improved retrieval performance. The CMU CS Web is used as the test bed and evaluation is based on the user's selection of links presented by Laser. The work finds that incorporating HTML markup into the TFIDF weighting scheme improves retrieval performance. Utilizing such markup may also be effective for improving spidering performance.

Cho, Garcia-Molina, & Page (1998) introduce a metric, PageRank, for valuing a web page based on its linkage properties. They show that PageRank is an effective spidering metric for locating pages with high PageRank counts or back link counts. However, these metrics perform poorly when the task is to locate pages that are relevant to a particular topic or query. Our research focuses on exactly that aspect: creating a framework to locate web documents that are related to a particular topic.

Balabanovic & Shoham (1995) describes a system that gathers interesting web pages for a user. They use a TFIDF metric to compare documents based on their textual features and use a best-first search technique to find such documents. Weights are modified daily based on user evaluation of gathered web pages. Results are positive, outperforming random selection and performing as well as human selected "cool" pages.

7 Conclusions

Reinforcement learning is an effective framework for spidering. We have shown that, for the task of finding research papers, a reinforcement learning spider is able to reduce effort by one-third over breadth-first search.

The most important next step after these preliminary results is to relax some of the restrictive assumptions made thus far. Towards this goal, experiments aim to show that representing delayed reward can further improve the efficiency of a directed spider. We believe that there are many features that are not indicative of immediate reward but are predictive of future reward; following them should improve performance further. We are also studying enlarged feature sets to represent a hyperlink for the purpose of the Q function: namely, we are working on using word neighborhoods, headers, titles, and words from hyperlink-neighboring pages.

Acknowledgements

Much of the work for this paper has been done in conjunction with the Cora project, a Computer Science research paper search engine, available from *www.cora.jprc.com.* We wish to thank Kamal Nigam and Kristie Seymore for their involvement in Cora and for their helpful comments and observations relating to this research. We also wish to thank Tom Mitchell and Mark Craven for their insights.

References

Balabanovic, M., and Shoham, Y. 1995. Learning information retrieval agents: Experiments with automated web browsing. In *AAAI-95 Spring Symposium on Information Gathering from Heterogenous, Distributed Environments.*

Bellman, R. E. 1957. *Dynamic Programming.* Princeton, NJ: Princeton University Press.

Boyan, J.; Freitag, D.; and Joachims, T. 1996. A machine learning architecture for optimizing web search engines. In *AAAI workshop on Internet-Based Information Systems.*

Cho, J.; Garcia-Molina, H.; and Page, L. 1998. Efficient crawling through URL ordering. In *Computer Networks and ISDN Systems (WWW7)*, volume 30.

Cover, T. M., and Thomas, J. A. 1991. *Elements of Information Theory.* New York: John Wiley and Sons.

Joachims, T.; Freitag, D.; and Mitchell, T. 1997. Webwatcher: A tour guide for the World Wide Web. In *Proceedings of IJCAI-97.*

Kaelbling, L. P.; Littman, M. L.; and Moore, A. W. 1996. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research* 237–285.

Lewis, D. D. 1998. Naive (Bayes) at forty: The independence assumption in information retrieval. In *ECML-98*.

McCallum, A.; Nigam, K.; Rennie, J.; and Seymore, K. 1999. Building domain-specific search engines with machine learning techniques. In *AAAI-99 Spring Symposium on Intelligent Agents in Cyberspace*.

Menczer, F. 1997. ARACHNID: Adaptive retrieval agents choosing heuristic neighborhoods for information discovery. In *ICML '97*.

Mitchell, T. M. 1997. *Machine Learning*. New York: McGraw-Hill.

Nigam, K.; McCallum, A.; Thrun, S.; and Mitchell, T. 1999. Text classification from labeled and unlabeled documents using EM. *Machine Learning*. To appear.

Torgo, L., and Gama, J. 1997. Regression using classification algorithms. *Intelligent Data Analysis* 1(4).