Abstract of "Exploiting the Structure of the Web for Spidering" by Joel D. Young, Ph.D., Brown University, May 2005.

Published experiments on searching the Web suggest that, given training data in the form of a (relatively) small subgraph of the Web containing a subset of a selected class of target pages, it is possible to conduct a directed search and find additional target pages significantly faster (i.e., with fewer page retrievals) than by performing a blind or uninformed random or systematic search such as depth- or breadth-first. An agent performing such a task is termed a *spider*. These experiments were carried out in specialized domains or under conditions that are difficult to replicate.

We present an experimental framework in which to reexamine and resolve the basic questions in such a way that results can be replicated and built upon. We provide high-performance tools for building experimental spiders and make use of the ground truth and static nature of the WT10g TREC Web Corpus.

We leverage powerful machine learning techniques within this experimental framework to learn regressions on discounted reward and navigational distance and apply these regressions to the problem of spidering. Experimental results on TREC 2001 Web ad hoc tasks show significant performance gains over blind and systematic search techniques.

Abstract of "Exploiting the Structure of the Web for Spidering" by Joel D. Young, Ph.D., Brown University, May 2005.

Published experiments on searching the Web suggest that, given training data in the form of a (relatively) small subgraph of the Web containing a subset of a selected class of target pages, it is possible to conduct a directed search and find additional target pages significantly faster (i.e., with fewer page retrievals) than by performing a blind or uninformed random or systematic search such as depth- or breadth-first. An agent performing such a task is termed a *spider*. These experiments were carried out in specialized domains or under conditions that are difficult to replicate.

We present an experimental framework in which to reexamine and resolve the basic questions in such a way that results can be replicated and built upon. We provide high-performance tools for building experimental spiders and make use of the ground truth and static nature of the WT10g TREC Web Corpus.

We leverage powerful machine learning techniques within this experimental framework to learn regressions on discounted reward and navigational distance and apply these regressions to the problem of spidering. Experimental results on TREC 2001 Web ad hoc tasks show significant performance gains over blind and systematic search techniques.

Exploiting the Structure of the Web for Spidering

by

Joel D. Young

B.S., Computer Science, Worcester Polytechnic Institute, 1990

M.S., Computer Science, Air Force Institute of Technology, 1996

Sc.M., Computer Science, Brown University, 2001

A dissertation submitted in partial fulfillment of the
requirements for the Degree of Doctor of Philosophy
in the Department of Computer Science at Brown University

Providence, Rhode Island

May 2005

This dissertation by Joel D. Young is accepted in its present form by the Department of Computer Science as satisfying the dissertation requirement for the degree of Doctor of Philosophy.

Date ____________ ____________________________________

Thomas L. Dean, Director

Recommended to the Graduate Council

Date ____________ ____________________________________

Thomas Hofmann, Reader

Date ____________ ____________________________________

Eugene Santos, Jr., Reader
University of Connecticut

Approved by the Graduate Council

Date ____________ ____________________________________

Karen Newman
Dean of the Graduate School

Vita

Major Joel Young was born on September 4, 1968 in Northampton, Massachusetts of Patricia Young and Peter Young. He graduated from Worcester Polytechnic Institute with High Distinction on 19 May 1990. While he was at WPI, Joel was inducted into the Upsilon Pi Epsilon computer science honor society (for which he served as President) as well as the Tau Beta Pi engineering honor society. Joel was also an active member of the Zeta Psi fraternity. Major Young received his commission through ROTC upon graduation and entered active duty ten months later.

His first assignment was to Keesler Mississippi for the Basic Communications-Computer Course from which he graduated first in his class. On 21 August 1991, Major Young came to Offutt AFB, Nebraska to work at Headquarters Strategic Air Command's Comm-Computer Center where he worked in the Cruise Missile Flight Simulation Section (Cruisers) which he became chief of in 1993. While at Offutt, HQ SAC stood down and the United States Strategic Command stood up and took its place.

In May 1995 the Cruisers were decommissioned and soon after Major Young transfered to the Air Force Institute of Technology (AFIT) where he completed his Master of Science in computer science and was inducted into the Eta Kappa Nu honor society. His research efforts focused on temporal reasoning with uncertainty. Major Young graduated from AFIT on 17 December 1996 as a Distinguished Graduate. At graduation, Joel received AFIT's highest award for academic achievement, the Gross Award.

From AFIT, Joel transferred to Air Combat Command (ACC), Communications Group (ACC CG)(formerly Computer Systems Squadron) at Langley AFB, Virginia. While at the Comm Group, he served in several positions including Program Manager, Plans and Resource Flight, Element Lead, Future Plans, Chief of Rear Operations

Support Center Communications, Lead Expeditionary Forces Experiment '98 Langley Site Engineer, and Chief of ACC's Systems Analysis Element, and acting Flight Commander of the Project Management and Systems Analysis Flight. In 1997 Joel deployed to Riyadh, Kingdom of Saudi Arabia supporting Operation Southern Watch as the United States Central Command Communications Operations Officer, Forward providing personnel, materiel, and engineering support to AF organizations at 10 Air Force installations throughout Southwest Asia. After returning from Saudi Arabia, Major Young lead the team planning, engineering, procuring, and installing all communications for what was then the Rear Operations Support Center to support Joint Vision 2010 and the Expeditionary Forces Experiments.

Major Young left Air Combat Command in July 1999 to attend Brown University as part of the Air Force Institute Of Technology's Faculty Track program. Joel was promoted to Major on 1 December 2001. After completing his PhD in Computer Science at Brown, Joel will take an Assistant Professorship at AFIT.

Acknowledgements

First and foremost I thank my committee members starting with my adviser, Tom Dean. Tom guided and mentored me enormously during my years at Brown. He was also exceptionally patient while I built momentum to finally get my thesis document written. Next I thank Gene Santos who encouraged me to consider Brown University for my doctoral work. I have never, not even for a moment, regretted this decision. I also thank Gene for years of long distance friendship and collaboration. Finally, I thank Thomas Hofmann for transmitting his passion for machine learning. I believe my conversations with him will yield many rich fruits in the years to come.

Next I would like to thank all of the members of the Brown Ballroom Dance Team. Your companionship, support, and love turned some of the most difficult times of my life into times of purest bliss. I miss you all! Christina, I will never forget you. Russell, thank you for picking up and carrying on where she left off. Michael and Thao and Susan, my adoption into your salsa circle opened a new world to me.

I would like to thank those who have been closest to me these years: Amanda and Nora and Mary who pushed me hard when I most needed it yet still loved me even when I sometimes pushed back. Lijuan who has been like my little sister. Noemi, my dearest friend, who has been so caring and understanding. My office mates, Dave and Heidi and Ying, with whom I spent so very very many hours, sometimes sharing more than we share with our closest lovers.

In the end and with my greatest love I thank my beautiful daughter Alix Elizabeth Young. She is the light and joy of my life! Thank you Wendy for raising her so lovingly!

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The world wide web has grown over the past decade into the largest human generated technological artifact—billions of linked pages traversed by millions of people every day. While the web contains an enormous quantity and depth of information, searching on the web can be quite difficult. Commercial search engines, sometimes called trawlers, go quite a way towards allowing us to find what we are looking for, but even so they don't go quite far enough. In short, it is easy to find lots and lots of related pages, but often few or none of them are what you are looking for.

While most search engines are extremely adept at finding documents easily categorized by a small set of key words, for some search problems it can be difficult to find a set of keywords which return a meaningful set of candidates. Other aspects of the web that make life difficult for trawlers are that parts of the web are inaccessible to the trawler (intranets, robot exclusions) and that the web is constantly changing thus trawlers inevitably fall behind.

The web can be viewed as a large directed graph in which web pages are the nodes in the graph and the hyperlinks in each page are the edges. One can readily imagine a robot walking through this graph selecting at each node which out-going edge to take next. Unlike a traditional robot, a web robot (appropriately called a *spider*) can teleport across the graph to any known location it knows the URL for at zero cost, that is, it doesn't have to walk the graph. So a spider taking advantage of this property has at each step the choice of exploring any node in the graph it knows how to get to. Those nodes which are known, but unvisited, are called *fringe nodes*. The set of all such nodes is called the *fringe*.

In the remainder of this thesis when we refer to a “page” think “node in a web graph” and when we refer to a “link” or “URL” think “directed edge in a web graph.”

A *target* page is simply a member of a possible arbitrary subset of the pages in the web-graph. It seems reasonable to focus on the cases where the subset is defined by some set of common properties. These subsets of pages are called *thematically unified clusters* [18]. An example might be pages that contain the term “Nietzsche” and are located in university philosophy department web sites.

The fundamental question for web spidering is:

Question 1. *Given an arbitrary starting page, is it possible to conduct a directed search to find target pages satisfying some target identification criteria faster than by performing a blind or uninformed random or systematic search?*

For certain target sets, such as pages with high in-degree, it is possible for a spider to efficiently find additional pages in the target set [11, 45]. However in general we consider this too hard a problem. Instead we relax the problem slightly:

Question 2. *Given training data in the form of a (relatively small) sample/subgraph of the web containing a subset of the target pages, is it possible to conduct a directed search and find additional target pages faster (with fewer page retrievals) than by performing a blind or uninformed random or systematic search, such as depth-first or breadth-first search?*

This question addresses a fundamental property of the web and, as such, is interesting even if we may not be able to efficiently exploit it.

We believe the answer to this question is positive, at least for a significant class of non-trivial search problems. Working on the hypothesis that the answer to Question 2 is “yes”, we immediately have the problem of how to take advantage of this property of the web for efficient search. If we view this as a learning problem, two approaches are immediately suggested: on-line and off-line learning. We only consider off-line learning at this time. Also, for the work presented in this thesis, we ignore the problem of target identification. In other words, we assume a target identification oracle is provided which decides if the page is or is not a target. In practice the oracle could use some sort of similarity metric, or just a list of known targets.

Question 3. *If the answer to Question 2 is in the affirmative, can we efficiently process the training data to generate an efficient search heuristic achieving the desired performance?*

This is really a two-part question. First, if the information is present on the web, can we efficiently process the training data to generate a search strategy performing better on average than random or systematic search strategies? Second, can we generate an *efficient* strategy? In other words, can we find a strategy with complexity small enough to still be advantageous in terms of real time and resources vs. random or systematic searches. One can imagine finding a strategy which requires so much computation to estimate the value of exploring a link, that while it requires exploring fewer links, actually takes longer to run.

Our working hypothesis is that the answer to both of these questions is "yes" for an interesting set of search problems. We can reasonably expect this from prior successful web-spidering efforts and as subgraphs of the web tend to be similar with respect to certain properties too larger subgraphs and to the web as a whole. The fractal phenomenon is called *self-similarity* and is observed in many characteristics including in-degree, out-degree, and PageRank power laws.

In the discussion so far we talked about using information local to a link to decide whether or not to explore that link. What exactly we mean by *local* may be the $25 million question. Ideally there would be sufficient information in the page containing the link itself, however, it is likely that there isn't enough information local to a single page and we must (or it would be more efficient to) also use information from other pages in the neighborhood of the link. For example the explored parents/children of the page containing the link. By local we mean any amount of information which can be reasonably considered to be small relative to the size of the web-graph.

In order to have a hope of answering these questions positively we need a better understanding of the structural properties of the web. In the remainder of this chapter we briefly explore important aspects of the web structure, models of how we might look at the web given what we know of the structure, and an outline of how these models are leveraged for spidering.

1.1 Properties of the Web

In this section we outline the high level properties of the web essential for spidering. We start with a discussion of common graph properties useful for describing web graphs. From this we talk about a richer model of the web-graph where we look at the web as a collection of hyperedges laid on top of the traditional web graph. We then talk about other projections of the web.

1.1.1 Graph Properties

As mentioned previously, we can view the web as a directed graph in which the nodes in the graph correspond to web pages and the edges in the graph correspond to the hyperlinks between those pages. Although the web is continually growing, we can take a snapshot at any particular moment to define our graph. We call such a graph a *web-graph*.

Note that this definition is still a little bit fuzzy since we haven't defined what a page is. Let's define a *page* as the dataset returned from a `http` request on a particular URL. Since a significant (and growing) portion of the web is generated dynamically based on the uniform resource locater (URL) provided, there could be as many web pages as there are valid URLs that can be written. A common approach is to only consider the *static* web, i.e., that portion of the web which is not generated by web template systems, e.g. `.php`, `.cgi`, and `.asp`. Of course it isn't necessarily obvious that a page isn't dynamic, e.g. `citeseer`. Also many pages with unique URLs have identical content. This phenomenon is called *aliasing* [32]. One might want to merge all identical pages into a single node. It is also possible that a page may have more than one link to another page (especially if ignoring anchors). Should these additional links be part of the graph giving rise to a multi-graph or should they be ignored? For now we won't specify exactly how we project from the web to our web graph.

Now that we have our directed graph, we can talk about the properties of this graph ignoring semantic content.

1. The *in-degree* of a node is the number of edges pointing into the node, i.e., the number of pages linking to that page, aka *backlinks* or *inlinks*.

2. The *out-degree* of a node is the number of edges pointing out of the node, i.e., the number of pages linked to from that page, aka *outlinks*.

3. The *PageRank* [46] of a node is a measure of how reachable a page is. Loosely, it is defined recursively in terms of its in-degree, its parents in-degree, and so on.

4. Closely related to PageRank are hub and authority scores [33] measuring, respectively, how strongly a page links to important pages and how important a page is.

5. The sizes of *strongly connected components* (SCC). A strongly connected component is a largest set of nodes such that all of the nodes can reach all of the others following only outlinks. A variation is the *in-SCC* size distribution which is defined on reachability through backlinks.

6. The sizes of *weakly connected components* (WCC). Reachability using the backlinks and outlinks. Corresponds to a connected component on the undirected graph.

7. The counts and sizes of *bipartite cores* [35]. A bipartite core consists of two sets of nodes such that every page in one set is pointed to by every node in the other set.

8. The *diameter* of the web graph. The diameter is the largest shortest distance (aside from infinity) between any two nodes in the graph. Diameter can be calculated based on backlinks, outlinks, or both.

9. The *average diameter* is the average shortest distance over all pairs of nodes with non-infinite shortest distance.

10. The graph's *clustering coefficient*. The clustering coefficient is an average of how tightly coupled the neighbors of each node are, that is how close the neighbors of a node are to forming a clique [59]. It is calculated by dividing the number of existing edges in a subgraph by the maximum number $(2n(n-1))$.

Several empirical studies have shown that many of the above web properties follow power-law distributions [8, 4, 2, 47]. Also, while the web does have different connected components (especially weak ones), by their very nature of being completely closed they aren't directly relevant to web spidering except that any page in a different component will never be reachable. The distribution of connected components is useful though in deciding if a particular web snapshot is web-like.

Milgram, in 1967 [44], described the small world phenomenon in social networks. For example, with a small number of steps, a short path can be found between any two people using only first name acquaintanceships, e.g., if you know me you can get to the President of the United States in three or fewer steps and if you know someone who knows me, you can get to the President in four. A small world network is defined by having low degree, small diameter and high clustering coefficient [34, 59]. This phenomenon is also observed on the web [1, 34].

1.1.2 Hypergraphs and Thematically Unified Clusters

The above web-graph properties suggest several ways to partition the pages of the web. One can look at all of the nodes in particular SCCs or all nodes having PageRanks in a particular range. Each of these partitions defines subsets of the nodes of the graph. These subsets can be thought of as hyperedges in a hypergraph. A hypergraph is a graph in which edges connect two *or more* nodes or equivalently a set of node subsets.

Dill et al˙ [18] studied properties of various subgraphs of the web. They found that if the subgraphs are selected with some semantic theme, they tend to show many of the same properties found in the web-graph as a whole as well as of larger subgraphs. This phenomenon is called *self-similarity* and is similar to that found in fractals.

They introduce the term *thematically unified clusters* (TUC) to describe subgraphs of the web selected with such a semantic theme. For example we can think of the target pages for a spider (and the links between them) as a particular TUC. The high degree of connectivity within a TUC allows spiders such as ARACHNID to exploit the structure inside a TUC, however, to actually find the TUC from outside it, we need to know how to navigate from page to page until we enter a TUC. Of course, these pages along the way are members of their own TUCs. One approach to solving the spidering problem then is to map the connections between TUCs forming a navigational backbone and then to navigate across that backbone.

In addition to semantically meaningful clusterings, there are many possible ways of projecting the web-graph onto a hypergraph. One possibility is to define a hierarchy of communities where a community is simply a collection of pages with more links into the community than out [19, 20]. Trivially the entire web-graph defines a community. One could then find a partition of the set of nodes of the web-graph such that each of the partitions has more links in than out. This process could then be repeated.

Probabilistic HITS

Cohn et al. introduce a probabilistic model providing a similar semantics to Kleinberg's hub and authority scores with the benefit of having a probabilistic foundation. The model categorizes potentially high-dimensional web pages (a dimension for each link) into a lower dimensional space defined over topics. For each of the model's topics, the model can provide the web pages most likely to link to other pages in that topic (hubs) as well as the web pages most likely to be linked to by on-topic pages (authorities).

1.1.3 Semantic Properties

The world wide web isn't simply a raw graph. Each of the pages of the web contains a potentially large amount of useful information. This information can be represented as a high-dimensional vector of features describing each web page.

Since web pages can largely be treated as text documents, a natural set of features are the term frequency counts for each of the words found in the text of the web page. This feature space might be augmented with additional features such as information characterizing the HTML markup in the pages, the time since last modification, or even summaries of graphical properties such as PageRank.

Once a feature space has been selected, any two documents can be easily compared by finding their *distance* or *similarity* in the vector space. Common techniques include cosine-angle distance and inner product similarity. One can also apply techniques for clustering documents based on their position in the vector-space and also for classifying/regressing the documents into particular categories.

A difficulty with using a vector-space model for the web is the massive dimensionality. A small trawl of a few tens of thousands of pages from the web can easily result in hundreds of thousands of different terms. The classic first-cut techniques to deal with the dimensionality is to use stopping and stemming. In stopping we drop all the stop words such as and and or which typically have very high term counts in all English language documents and hence don't make good discriminators. Typical stop-word lists have about 500 terms. The second common technique, stemming, involves finding the root of the word and collapsing all terms with a common root to a single dimension in the feature space. Perfect root identification is difficult but efficient approximation techniques such as Porter stemming do well in practice.

After applying these classic techniques for reducing dimensionality, we can have too many dimensions. Powerful techniques with various degrees of rigor for reducing the dimensionality include randomized projection matrices, latent semantic indexing, probabilistic latent semantic indexing, and information gain.

1.2 Searching the Web

Since a web link uniquely identifies a page, if we know the entire web graph then a web link gives us access to complete information on the benefit (in terms of finding target pages) of exploring the page linked to. But then, if we already knew the web graph there would be little need for spidering. What we are interested in, then, is what and how much information about the web-graph is necessary and sufficient to make a good estimate.

At a high level, we can reduce the spidering problem to that of selecting which of many pages to explore next. What we are looking for is a procedure that estimates some quantity correlated with the potential reward to be gained by exploring a particular link. By *reward* we mean both the target pages at or reachable from that link as well as the information gained by exploring the link. The procedure takes as input the current observations of the world, optionally makes some further observations, and then estimates the quantity, e.g. estimate the number of targets reachable in 3 steps from the link. The estimate is then used by the spider to decide which link from the fringe should be explored next.

Since the web graph is finite, a spider using any exhaustive technique will eventually find all of the target pages simply by repeatedly exploring pages from its fringe. Each time a new page is opened any links whose target pages haven't been explored are added to the fringe. Different strategies for selecting pages from the fringe can give us a breadth-first search, a depth-first search, or anything in between.

There is a spectrum of possibilities in "anything in between." At the most basic level is a simple random search in which pages are selected uniformly at random from the fringe. The random search can be shifted closer to breadth-first or depth-first by biasing the sampling distribution to favor or disfavor pages discovered more recently. Breadth-first, depth-first, random search are what we call *naive* search strategies. Naive in that they behave agnostically with respect to the semantic content of the pages.

The technique of ranking the pages by some estimator of the available reward gives us a best-first search. Of course we would like an estimator which accurately estimates the reward achievable from selecting that page, but what other properties do we need? There are three primary goals: First the estimator needs to be readily learned from the training data; Second the estimator needs to be able to efficiently compute the score for a given feature vector (web page); Finally the estimator needs to perform well guiding the best-first search on test data. For this thesis, by perform well we mean "outperforms the naive search strategies."

1.3 Experimental Environment

We state above that a good estimator for guiding the best-first search needs to perform well in tests. What is the environment in which these tests are performed?

In our early, proof-of-concept experiments, we performed experiments using the live web. We used pages from the Google directory entry for the philosopher Nietzsche as targets and used Google's `link:` feature to get the backlinks for building our training set. This had several problems. First was we became dependent on the internet "weather." The success of our experiments depended on the time of day and the status of all the routers in our path. The same experiment conducted from one day to the next would have different results. Also we became vulnerable to the whims of Google who changed their interface to the search engine regularly. Finally, for about half a year, our access to Google was cut off due to terms of service issues. While, these issues were eventually resolved successfully, we had already moved on.

To escape from our problems with the weather and permission to use the Google backlink feature, we decided we needed our own snapshot of the web. Rather than trying to grow our own, we decided to use a pre-packaged snapshot. As we already had access to the WT10g [16] corpus it was a leading candidate. What really cinched it though was that the NIST TREC competition had used the WT10g corpus and provided queries with labeled data giving all of the targets in the corpus. This makes for an ideal training and testing environment. Finally the WT10g corpus is structured quite like the web as a whole [55] which is very important for using success on the corpus as indicator of applicability to the "real" web.

We developed a set of HTTP proxy servers able to serve the WT10g corpus as if one were accessing the real web through a normal firewall/proxy. This allowed us to leverage all of the same tools we used for our proof-of-concept experiments with the live web. Our environment is discussed more fully in Chapter 4.

1.4 Experiments on Spidering

In our initial exploration of this problem we selected the problem of searching for pages containing the family name "Nietzsche." Clearly this is a simplified problem, easily addressed by the trawler based engines. The simple target criteria allows us to trivialize the problem of target detection and focus on next action selection.

We trawled for pages out to an arbitrary depth (4), through (a sampling of) the parent links (using Google) from a set of starting nodes (high PageRank Nietzsche pages). The resulting pages are then post processed through a target identifier to mark all discovered targets as root nodes in the web-graph. A shortest path (Dijkstra) is executed to label each page with its distance from the closest known target. The target identifier used is a simple check for the existence of the word "Nietzsche." The trawl is limited to pages in the {`.com`, `.net`, `.org`, and `.edu`} domains.

The results from a trawl are then converted into a format compatible with Andy McCallum's Bag-Of-Words (BOW) library. We then train a linear support vector machine (SVM) to discriminate between pages at each distance from all the other trawled pages. For example, if the training set has pages labeled with five different distances from targets, five support vector machines are trained, one for each distance. To estimate the navigational distance of a candidate page we convert the page into a bag-of-words and run that bag-of-words through each of the five machines. We then select that machine which scored the largest displacement onto the positive side of the decision boundary. The distance for that machine is our estimate. The SVM regression is then used to generate a priority value for each link on the fringe hopefully favoring discovered links closer to target pages in a spider run. The results from these experiments our discussed in Chapter 3.

For experimental repeatability (See Section 1.3) and other reasons for our full scale testing, we switched to using the WT10g corpus and the TREC tasks. In our testing on the WT10g corpus, we extended our work on learning a regression on navigational distance to learning an estimate of discounted reward earnable by selecting a particular page to explore next. We also evaluated an alternative technique in which we focus on learning an ordering of pages rather then some particular score. These results are presented in Chapter 5

1.5 Summary

In this chapter, we introduced the problem of web spidering and argued that efficient web spidering should be considered from the vantage point of a solid understanding of the structure of the World Wide Web. We outlined our experimental environment and outlined the experiments we conduct on the web to demonstrate our spidering hypothesis.

In the remainder of this thesis we develop the theme of demonstrating the truth of our hypothesis. In Chapter 2, we discuss some of the important previous work by the seminal researchers in the field. We focus on understanding the structure of the web and on efforts for searching the web, particularly spidering. In Chapter 3, we introduce in detail the spidering problem and our approach for solving it. We also outline our initial results from working on the live web. In Chapter 4, we lay out the experimental framework we use for our experimentation focusing on the dataset used, the tasks selected, as well as the hardware and software developed as our test harness. In Chapter 5, we detail our experimental method and dissect our results. We conclude, in Chapter 6, with a summary of our results in proving the spidering hypothesis and with an outline of future research enabled by this thesis.

Chapter 2

Foundations and Previous Work

Effective behavior in any domain requires an understanding of one's environment. For spidering, the environment is the World Wide Web. The more we learn about the Web, the more effectively one can search within it. For this reason, we start our discussion with advances in our knowledge of the structure of the web. From this foundation we explore techniques for exploiting this structure for web spidering and search. In addition we discuss relevant aspects of reinforcement learning and machine learning.

2.1 Structure of the Web

In this section we survey key research into the structure of the world wide web. The results are presented with the constant bias towards those items enabling web spidering.

2.1.1 Graphical Properties

Power Law Properties

Many features of the World Wide Web are exhibit power law behaviors [4, 2, 8, 47] in their distribution. That is, the frequency or size of the features can be described using Equation 2.1:

$$P(k) \approx k^{-\gamma} \tag{2.1}$$

Where k is the magnitude of the occurrence, γ is a phenomenon specific constant, and $P(k)$ is the probability that the event takes value k. Power law distributed phenomenon can be recognized by a straight line when plotted on a log-log graph. The slope of the graph gives γ.

In [4, 2] Albert, Jeong, and Barabasi show, based on a crawl of the `nd.edu` domain and other confirmation crawls, that the in-degree and out-degree distributions of the web follow the power law with coefficients $\gamma_{out} = 2.45$ and $\gamma_{in} = 2.1$.

Of crucial importance to web spidering is the idea of *diameter*. Albert et al. define the diameter of the web , $<d>$, as the average over all pairs of vertices of the smallest number of links to follow between the vertices. Based on a simple power-law generative model they predict $<d> = 0.35 + 2.06 \log(N)$ where N is the size of the graph. The prediction from this corpus is confirmed by their observation of the `nd.edu` domain. At the time of writing, they estimate the diameter of the web-graph as 19. Note that I measure the diameter of the WT10g corpus at just over 9 while their model would predict a diameter of a touch over 7.

Since the length of the expected shortest path between two pages is small, a spider which can make a perfect choice at each step could reach the target quickly however a spider performing a naive search would have to crawl a significant constant fraction of the web.

In [8], Broder et al. extend these power law results with significantly larger crawls. They confirm the in-degree power law constant observed by Albert et al., and observe out-degree power law with a different constant however. Interestingly out-degree doesn't follow power law for small out-degrees. They also observe power law distributions for strongly and weakly connected components.

From their observation of the sizes of the strongly connected components, they challenge the generality of the Albert, Jeong, and Barabasi diameter results. If less than half the nodes are in the strongly connected component, then there is a significant probability that the distance between two randomly selected pages is infinite. By performing repeated breadth-first searches from random starting points in their corpus they developed the now classic butterfly diagram. They observe the directed diameter of the strongly connected component to be at least 28 and the average connected distance ($< d >$) for in- and out-links as about 16 and only 7 for undirected distance.

75% of pairs of pages are not directly connected. This implies that the web really doesn't have small world properties: it isn't connected.

PageRank

The in-degree of a page gives us a measure of how important that page is on the web: the more pages that link to a page, the more important the page must be. There are, however, some problems with using in-degree to measure importance. First, a very important page might be pointed to by only one other page. That other page could have high in-degree but that wouldn't be reflected in the ranking of the first page. Second, as the web is a human-manipulatable creation, the owner of a page can artificially boost the in-degree simply by creating other pages linking to it.

To work around these problems, Page and Brin et al. defined the notion of PageRank [46]. PageRank is a global measure of the importance of a page. Imagine a web surfer who simply chooses a page at random from the links available on the current page. In other words, the surfer is taking a random walk on the web graph. If every page were reachable from every page, eventually the probability of the surfer being in a particular page would be independent of the surfer's starting location. However the web-graph isn't fully connected so the surfer reaches dead ends at times. To get around this, we also allow the surfer to jump at random to any other page in the web. In other words, we link all pages with no links to all of the pages of the web-graph. The random walker could still get stuck in cycles (rank sinks), so we also allow the random walker to occasionally jump to some other page at random. If we now let the surfer take a random walk, the probabilities of being in a particular page would converge. This distribution is the PageRank.

The PageRank calculations [7, 46, 25, 26, 52] can also be written as follows:

$$PR'(v) = \sum_{u} PR(u)T(v,u) \tag{2.2}$$

where v and u are nodes in the web-graph, $PR(u)$ is the PageRank at page u, and $T(v,u)$ is the probability of transitioning to node v from node u. For a typical PageRank calculation, $T(v,u)$ is

$$T(v,u) = \frac{1-\beta}{N} + \frac{\beta}{O(u)} \mathcal{I}_{(u,v)\in G} \tag{2.3}$$

where N is the number of nodes in web-graph G, $\mathcal{I}$ is the identity function, $O(u)$ is the out-degree of page u, and β is the probability of whether to walk on the graph or to jump randomly. If we write our PageRank equation in matrix form we get:

$$\vec{pr} = T\vec{pr} \tag{2.4}$$

where T is the transition matrix formed from $T(v,u)$. Since T is a stochastic matrix $\vec{pr}$ is an eigenvector of Z with eigenvalue 1. $\vec{pr}$ is the vector of all our PageRanks.

As initially proposed in [46] and further developed in [26, 52] the transition matrix can be weighted to favor pages of certain types. Page and Brin [46] examined the effects of changing the jump distribution from uniform to putting all of the weight on a single page. They found that the rank of pages in the neighborhood of that page were strongly influenced. For example they compared using John McCarthy's homepage and Netscape's page as the jump destination and found (not surprisingly) that computer science pages were typically ranked higher in percentile by the McCarthy PageRank vs. the Netscape PageRank. Richardson and Domingos [52] tie both the jump and walk distributions to distributions generated from query relevance measures such as TF-IDF (see 2.1.2 below). In limited usability studies they conclude that their modified PageRank performs better than a combination of query relevance and traditional PageRank. Haveliwala [26] presents an approach in which customized PageRanks are calculated for each of the top-level ODP categories. The jump distribution is uniform over the leaf pages in the category. At query time, the customized PageRanks are combined by the class probabilities conditioned on the query terms. The probabilities are calculated through a maximum likelihood unigram language model.

Hyperlink-Induced Topic Search (HITS)

Similar in spirit to PageRank, Kleinberg [24, 33] defines *hub* and *authority* scores in terms of the web-graph structure. A hub is a page that points to authoritative pages while an authority is a page pointed to by many hubs. Like PageRank this is a recursive definition. The authority score $x(v)$ for page v is

$$x(v) = \sum_{u|(u,v)\in G} y(u) \tag{2.5}$$

where $y(u)$ is the hub score of page u and is defined as:

$$y(u) = \sum_{v|(u,v)\in G} x(v) \tag{2.6}$$

Both $\vec{x}$ and $\vec{y}$ are normalized such that their L-2 norm is 1. Let A be the adjacency matrix for the web-graph. Starting from an assignment of $\vec{1}$ to both $\vec{x}$ and $\vec{y}$ and iteratively applying the equations above, $\vec{x}$ converges to the largest eigenvector of $A'A$ and $\vec{y}$ converges to the primary eigenvector (that with the largest eigenvalue) of AA'. Notice that $A'A$ is the *co-citation* matrix, i.e. element (u, v) gives the number of pages that link to both pages u and v, also AA' is the *bibliographic coupling* matrix, i.e. element (u, v) is the number of pages that are pointed to by both u and v.

In addition to the primary eigenvector, the other eigenvectors of $A'A$ and AA' are also interesting. It turns out their eigenvector sets are identical and the eigenvectors associated with the common eigenvalues are coupled and show additional sub-community structure. These eigenvectors can have both negative and positive values and if one examines the set of pages with positive value vs. those with negative values, the sets tend to be antagonistic, e.g., pro-life v.s. pro-choice. As the size of the associated eigenvalue drops, the eigenvalue tends towards noise.

While the algorithm above only uses the link structure, the full HITS algorithm is actually performed on topic relevant subgraphs. A subgraph for HITS is built from a set of root pages gathered from a search query and expanded with the parents and children of those pages. Because the hubs and authorities portion of the algorithm relies strictly on link structure, some problems can arise. In [10] Chakrabarti et al. point out some of these problems. On very specific queries, sometimes the hubs and authorities HITS find are more apropos to a more general category. This isn't that surprising because a very tight set of core pages would presumably be embedded in a more general community. Some hubs and authorities are strong on a couple of disparate topics. HITS won't be able to see this and may give authority to pages outside of the desired topic. Finally, pages on a particularly topic relevant site may all point to a common page, e.g. the webmaster's page, which will lead HITS to conclude that page is an authority on the topic even though it isn't. They propose mitigating these tendencies by using weighted sums when iteratively computing the hubs and authority scores. The weights are calculated based on the semantic content of the pages.

Bipartite Cores

HITS helps us identify the authoritative pages on a topic and those pages which serve as hubs linking to the authorities. Together the hubs and authorities define a community of pages. The authoritative pages gain their authority from co-citation by other pages including the hubs. Kumar et al [35] refine this idea to emphasize the tight communities defined by patterns of co-citation. Consider the graph generated as follows: take each page in a web-graph and copy it into a set of authority pages and into a set of hub pages preserving only those links which tie from the hub pages to the authority pages. This new graph is *bipartite*, that is it partitioned into two sets. Suppose that there exist a few, say j, pages in the authority set which are linked to by every single one of i pages in the hub set. This collection of pages would form a very tight community. Such a community is called a (i, j)-sized bipartite core.

Kumar et al. propose that these bipartite cores can server as harbingers of nascent web communities and by finding these cores on the web one can identify new communities, perhaps even before the creators of the community are aware. Because finding cores is computationally expensive, the authors perform an extensive exclusion (including filtering for aliased pages) and pruning pre-processing step to get a small enough dataset to allow enumeration of cores. Starting from a 200 million page dataset they pruned it down to about 20 million candidate authority pages. From this data set they identified about 200 thousand unique $j = 3$ cores. Spot checking of the cores showed that almost all of them identified actual communities. Also they found that pages in cores tended to have a higher life span vs. the typical web page's six month lifespan.

Probabilistic HITS

As discussed previously, HITS is based on finding the principle eigenvectors of the co-citation (AA') and bibliographic coupling ($A'A$) matrices in effect minimizing the mean squared error of the factored approximation of these matrices. In other words, HITS makes an implicit assumption of Gaussian noise. However since the Gaussian is symmetric and has infinite tails, it is not appropriate for modeling hyperlinks. Cohn and Chang [13] propose a latent variable model based on a multinomial, rather than Gaussian, noise distribution.

While the authors applied their method to bibliographic data, for consistency we will use web terminology below. The PHITS model is based on the independence assumption that if one knows the community then the destination page of a link is independent of source page of that link. In other words, the probability of observing of a link (d_i, c_j) from page i to page j given the community z is simply $P(d_i|z)P(c_j|z)$ and to find the probability of an arbitrary edge we sum over the communities:

$$P(d_i, c_j) = \sum_{z \in Z} P(z)P(d_i|z)P(c_j|z) \tag{2.7}$$

where $Z = \{z_1, z_2, ...z_k\}$ are the different communities.

The parameters for this model are the $k-1$ probabilities for the communities $P(z)$ and the k $P(d|z)$ and $P(c|z)$ vectors. The parameters are learned through EM and due to the number of parameters being linear in number of training samples, overfitting can occur. This is avoided through the use of *tempering* [28, 13, 5]. $P(c|z)$ and $P(d|z)$ play the rolls of HITS authority and hub weights, respectively.

Small World

An interesting property of the web crucial for the success of spidering (also observed in social networks [44]) is the small world phenomenon [34]. The phenomenon consists of two components: first, each page is connected to most other pages by a relatively short series of links and second, the graph is highly clustered. Kleinberg gives a generative graph model which provably exhibits the small world phenomenon and an algorithm which with high probability using only local information finds a short path from an arbitrary source to an arbitrary target. Adamic [1] gives empirical evidence that the web itself has small world properties.

Self Similarity in the Web

Dill et al. [18] propose that the web has a fractal nature. In other words, portions of the web share similar properties to the web as a whole. They introduce the terminology *Thematically Unified Clusters* (TUCs) to refer to sets of pages selected to be cohesive clusters. They observe that these clusters obey power law distributions for in-degree, out-degree, connected component sizes, and bipartite core counts. Also the TUCs show the same butterfly graph structure as the web as a whole. They also note that TUCs are connected together by what they call a *navigational backbone.*

2.1.2 Contextual Properties

The above description focused on graphical properties of the web. While looking only at the graph structure tells us a lot, a strong understanding of the web requires more than that. There is too much valuable information stored in the pages themselves. In this section we briefly discuss some standard information retrieval techniques which consider only the content of the documents (web pages). From this base we will discuss methods which take advantage of both the link structure and the content of the web.

The traditional method for representing documents is the vector-space or bag-of-words model [23, 53]. In the vector-space model, documents are represented as points in a high dimensional space. Each dimension represents a term in the dictionary and the value for a document in a particular dimension is typically some representation of the number of occurrences of that term. The model is often called the bag-of-words or BOW model because it discards all structure in the documents and reduces a document to simply a collection (or bag) of terms.

Once the document collection has been reduced to a set of vectors, the cosine of the angle, aka cosine-angle distance, provides a powerful measure of the difference between two documents or between a query and the documents. The cosine-angle is computed:

$$\begin{aligned} cosineangle(d_i, d_j) &= \frac{d_i \cdot d_j}{|d_i| \times |d_j|} \qquad (2.8) \\ &= \frac{\sum_{k=1}^{t} d_{i_k} \times d_{j_k}}{\sqrt{\sum_{k=1}^{t} d_{i_k}^2} \times \sqrt{\sum_{k=1}^{t} d_{j_k}^2}} \qquad (2.9) \end{aligned}$$

where d_{i_k} refers to the frequency of term k in document i. The resulting value ranges between zero and one and gives a measure of the similarity between two documents.

The raw term-frequency vectors are often normalized to reduce the impact of terms that occur frequently across many documents in the collection. We define the normalized term-frequency tf for document i and term k as

$$tf_{ik} = \frac{d_{ik}}{\max_j d_{ij}} \qquad (2.10)$$

This normalizes the raw term frequencies by the size of the largest dimension in the document. To reduce the impact of common terms, traditionally one then normalizes with a measure of the frequency a term occurs across documents. A common measure is the *inverse document frequency* or *idf*. The *idf* for term k is

$$\begin{aligned} idf_k &= \log \frac{N}{n_k} \qquad (2.11) \\ &= \log \frac{N}{\sum_{i=1}^{N} d_{ik}} \qquad (2.12) \end{aligned}$$

One then multiplies the normalizes term-frequency score by the inverse document frequency to get the classic TF-IDF weight for term k in document i:

$$tfidf_{ik} = tf_{ik} \times idf_k \qquad (2.13)$$

Cosine-angle similarity measures with TF-IDF weights have been the bread and butter technique of information retrieval for over 30 years now.

In addition to easy computation of document-document and document-query similarity, the vector-space model opens the door to many dimension reduction, classification, and clustering techniques. Here we present a few common techniques.

Latent Semantic Indexing

Latent Semantic Indexing (LSI) is a linear algebra based technique for dimension reduction. The term frequency vectors are projected into a *latent semantic space* capturing most of the covariance in the data. The latent space is often an order of magnitude smaller than the original space.

The latent space is selected by performing a singular value decomposition (SVD) on the matrix of documents. All of the singular values (eigenvalues of the eigenvectors of the covariance matrix) except the k largest. k can be selected to give a desired dimension reduction, say $k = 500$ or $k = 1000$, or to capture some fixed amount of the variance. The documents are then projected into this reduced space capturing some of the hidden structure of the documents.

For comparison against queries or for other analysis, the documents in the latent space can be projected back into the term frequency vector space. Often this re-projection reveals patterns of synonymy: Terms that had zero weight in the original vectors will have weight in the new vector do to the coincidence of terms.

There is an implicit assumption of normality arising from the calculation of the singular value decomposition. For some applications such as image compression or analysis, the Gaussian assumption is appropriate, but term frequencies do not tend to be normal. Despite this, strong results are achievable with LSI.

Naive Bayes

Naive Bayes defines a conditional probability distribution of the class of a document given the terms in the document. The naive Bayes model assumes that terms in each document are independent given the class of the document. Just as the assumption of normality is false for text documents in LSI, this assumption is clearly wrong also. Never the less, naive Bayes is a very effective technique for classifying text documents.

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)} \alpha P(d|c)P(c)$$

Since the classifications are assumed to be mutually exclusive, that is, each document is generated by only one class, the normalization term $P(d)$ can be dropped. Typically the terms in the document vector are assumed to be generated by either a Bernoulli (word occurs or not) or multinomially (term counts). The model is learned by estimating the parameters of the chosen distributions from the data. Various statistical techniques can be used for the learning.

Probabilistic Latent Semantic Indexing

As discussed previously, LSI assumes a Gaussian noise model, however the Gaussian is not an appropriate distribution for text and hyperlink models. Hofmann [29, 28] proposes a latent model with strong probabilistic foundations.

The model is defined over a fixed collection of documents and a fixed dictionary of words and it assumes an unobserved variable upon which documents and words are dependent. The unobserved variable can take on one of k values where k is an externally specified parameter. Let D be the set of document indices $1..n$, let W be the set of word indices $1..m$, and let Z be the set of values $1..k$ which the latent variable can assume. Then

$$P(d \in D, w \in W) = \sum_{z \in Z} P(d|z)P(z)P(w|z) \tag{2.14}$$

Gives us the probability of observing word w in document d. Given knowledge of z, d and w are independent. If we write the above model in matrix notation

$$P_{i,j} = P(d_i|z_k)_{i,k}\mathrm{diag}(P(z_k))_k P(w_j|z_k)'_{j,k} \tag{2.15}$$

then we see that $\mathrm{diag}(P(z_k))_k$ plays the roll of $\hat{\Sigma}$, $P(d_i|z_k)_{i,k}$ plays the roll of $\hat{U}$, and $P(w_j|z_k)_{j,k}$ plays the roll of $\hat{V}'$.

Because this is a probabilistic model, it can be meaningfully refactored to answer many interesting questions.

PHITS/PLSI

As we saw in our discussion of HITS, taking advantage of the content of web pages (in that case through seeding the process with pages drawn from a search query) can be profitable. Cohn and Hoffman [14] present a combined technique giving a probabilistic model accounting for both the link structure between documents and contents there of.

2.2 Searching the Web

In addition to direct understanding of the structure of the web, we get valuable insight into the structure of the web by observing the behavior of spiders operating on the web. Below we give a more rigorous definition of what we mean when we talk about spiders, broad categories of spiders discussed in the literature, and interesting observations of the behavior of these spiders.

2.2.1 Search Engines

Many research groups and companies have focused on developing the big-iron search engines [6, 7]. These systems, while dramatically successful (Google is currently trading at over $170 per share), are not what we are interested in. They work by trawling and digesting huge portions of the web. Research in this area tends to focus on storage techniques and fast and relevant information retrieval. Algorithms need to be exceedingly fast since the databases are many terabytes in size. Some of the techniques from the big search engines such as PageRank [7] are relevant to understanding the nature of the web and to spidering.

The search engines answer different kinds of questions then we are interested in. They are very good at identifying important web pages answering some particular textual query. They can break down, however if the information we are interested in is in a rapidly changing portion of the web or in a robot-exclusion zone or private intranet that can't be trawled. It is also (currently at least) difficult to specify queries involving relationships between pages such as pages about a particular philosopher linked to from rock musicians.

2.2.2 Spider Types

At the most general, a *spider* is an agent that explores the web. We parameterize spiders on three primary dimensions: the goal, the resource constraints, and the strategy for achieving the goal. Variance on these dimensions yields a vast set of different spiders.

The goal of a spider might be to find pages in a particular TUC, e.g. the entire web graph, pages containing the name "Nietzsche", or pages in the Brown website.

Another possibility might be to find pages scoring highest on some evaluation. Spiders with these goals are called *topic seeking spiders*.

Types of resource constraints for spiders include the amount of time and memory available. Possible strategies include random or systematic approaches such as depth-first or breadth-first search and best-first strategies using various heuristics.

Cho, García-Molina, and Lawrence Page (of PageRank fame) provide a solid problem definition for web spidering [11]. They give the problem as "What URLs to scan and in what order?" in the context of three different classes of spiders:

- Crawl and Stop: The spider crawls for K steps. The perfect spider would have visited the $R_1, ..., R_K$ most important pages defined by some importance metric where R_K has the lowest importance. This is the gold standard. A real spider would, when crawling K steps, cover $M \leq K$ pages with importance as good or better than page R_K. They define the performance of such a spider as $P_{CS}(C) = (M\dot{1}00)/K$. The ideal spider would have performance 100.

    1. Goal: Find the most important pages.
    2. Constraints: Only K retrievals allowed.

- Crawl and Stop with Threshold: The spider takes K steps and finds $T \leq K$ targets. Target pages are defined by some threshold on the importance measure. Let H be the number of such pages. $P_{ST}(C) = T/K$. For $K \leq H$, the ideal spider would have performance K/H.

    1. Goal: Find pages in a TUC (above a threshold).
    2. Constraints: Only K retrievals allowed.

- Limited Buffer Crawl: Spider can only keep B pages in its buffer and must drop pages when the buffer is full. The spider is allowed to exhaust the web graph if possible. Target pages are defined as those with importance above some threshold or as all the pages whose importance is higher or equal to the Bth highest page. Note that the strategy for this type of spider must also select which pages to forget.

    1. Goal: Find pages in a TUC (above a threshold) or find the most important pages.

2. Constraints: Only B pages can be remembered.

2.2.3 Spider Strategies

In the previous section we discussed different dimensions on which spiders might be defined and gave some canonical types based on their goals and constraints. As mentioned, a spider also needs a strategy to decide what pages to explore at each step.

There is a wide range of spidering strategies. The simplest are the familiar random and systematic traversal patterns on graphs:

- Breadth First: Selects pages based on a first seen first explored basis.
- Depth First: Selects pages on last seen next explored basis.
- Random Search: Selects a page at random from the fringe.
- Random Walk: Selects a page at random from the links on the most recently explored page with unexplored links.

With unconstrained memory, all of the above will eventually exhaust a fixed web graph.

More advanced spidering strategies attempt to perform a best-first search, selecting that page from the fringe which is hoped to help best achieve the goal. In the remainder of this section we detail several of the key innovations in strategies for web spidering.

Filippo Menczer introduced one of the earlier intelligent web spiders [42] with his ARACHNID (Adaptive Retrieval Agents Choosing Heuristic Neighborhoods for Information Discovery) system. Menczer proposes a spider architecture combining aspects of genetic programming with reinforcement learning.

The spiders are started from a set of starting documents and provided a set of query keywords. Each spider is given a random assignment to the parameters governing its behavior and started at one of the starting documents with an initial energy store.

Spiders are selected at random from a pool of spiders. The chosen spider selects which link to follow from the current page, processes feedback from the user if the page is rated by the user otherwise a relevance score is computed, learns from the experience, and if enough energy is gained, replicates itself and mutates the copy.

Energy is gained in the relevance feedback step and is lost each time a page is fetched. If the energy goes to zero, the spider dies and if it crosses a threshold then the spider reproduces and gives half its energy to the child.

Two variations are proposed. A "naive" spider which does link selection in the target document based on counts of query keyword matches modified by distance the keyword was observed in the page from the link under consideration. The links are selected probabilistically based on these counts. Relevance computation is based on the cosine matching function. No learning is done in the Naive spider.

In the learning spider, link selection is done on the basis of "counts" generated by a feedforward neural network with inputs for each of the spider's keywords. In addition to gaining energy from labeled pages, the spider tracks all the terms observed with a counter, one for each term in the document, which is incremented if the document is a target page and decremented otherwise. These term counters are then used to calculate modified TF-IDF weights in the relevance computation on new pages. Learning is done by a connectionist Q-learning algorithm where the reinforcement computation is used as a reinforcement signal for training the neural net by back-propagation of error.

Menczer tested the non-naive spider on a portion of the Encyclopedia Britannica called the *Propaedia* organized in a hierarchical topic tree. He ran the spider four times varying on whether neural net weight learning and relevance feedback were provided. From very limited runs (at least as presented in the paper), he concluded that the spider using both learning and relevance feedback performed significantly better.

ARACHNID operates on the assumption that pages in a given topic tend to point to other pages in that topic at a rate higher than expected based on the topic density of the web as a whole. They call this property *semantic topology*, that is, semantic topology exists when the probability of an arbitrary on-topic page linking to other on-topic pages is greater than that of an arbitrary page drawn from the web as a while linking to other on-topic pages. Semantic topology, if found, is exploited by the learning algorithms to help the spider stay inside the topic pages. If a spider is not started in a relevant document, then it will make what is effectively a random walk until it finds relevant documents.

This notion of semantic topology corresponds to the fractal phenomena observed in the web. Pages in a TUC tend to connect to other pages in a TUC. We would like spiders to be able to exploit target-rich portions of the web-graph when found.

Web Communities

Exploiting a community property of the web closely related to semantic topology, Flake et al [19, 20] present an alternative and complementary strategy for efficiently identifying web communities solely from link structure. They define a *community* as a collection of pages with more undirected links into the community than out of it. They point out that this definition is unworkable in practice as it requires graph partitioning.

They go on to define an approximate iterative algorithm based on max-flow/min-cut. Loosely, they begin with a set of seed pages. From this set of seed pages they crawl to a low fixed depth and compute a community on the crawled graph using min-cut from a virtual source tying all of the seeds together and tying each non-seed node to a virtual sink. They then select the pages in the computed community with the highest degree, add those pages to the seed list and repeat.

Starting from the home pages of Francis Crick, Stephen Hawking, and Ronald Rivest they created three communities using the above algorithm. Each community had about 200 pages. Using the Kullback-Leibler distance they identified those terms most differentiating the pages in the community from $10,000$ random pages. The terms appeared highly relevant to the target communities.

Importance Metrics

As hinted at in the description of basic spider types, Cho et al. define an importance metric defining the target set. They propose several candidate metrics:

- Page/Query Similarity Given a page P and a query Q, $IS(P, Q)$ gives the importance of the page relative to the query. Possibilities are inner-product, cosine-angle, etc. If the pages are normalized, e.g. TF-IDF, then only an approximate importance $IS'(P, Q)$ can be computed without spidering the entire web.

- Backlink Count $IB(P)$ is the number of inlinks into page P. This is a global metric.

- PageRank $IR(P)$ is the *backlink* PageRank.

$$IR(P) = (1-d) + d\left(\sum_{i=1}^{n} IR(T_i)/c_i\right) \tag{2.16}$$

  where $T_{1\ldots n}$ are the parents of P and c_i is the outlink count of T_i (assumed to be the size of the graph if T_i has no outlinks. The PageRank measured is approximate as knowledge of the entire graph isn't available to the spider.

- Forward Link Count $IF(P)$. Can't be readily estimated without actually reading P.

- Location Metric $IL(P)$ based on the URL of page P.

- Combinations of the above.

Finally they define an *ordering metric* which the spider uses to select the next page to explore. Typically they tie the ordering metric to some importance metric. Using the Stanford WebBase [27], they compare several ordering metrics with the different goals. Interestingly, PageRank often performs better at finding high backcount pages than backcount itself.

Cho et al. use a uniformly sampled spider and a breadth first spider as baselines. It would be nice if they also compared a random sampled spider that only sampled from the fringe as well.

Perhaps the most interesting experiment is when they combine backlinks with similarity for the importance metric with a threshold with PageRank for the ordering metric. The PageRank does significantly better than backlinks and loosely on a par with breadth-first search. This is interesting since it suggest that graphical properties can be useful features for finding at least certain kinds of pages on the web. Note however that the target type is partially defined on a in-link structure for which PageRank appears to be an upstream predictor [46].

Spidering as a Reinforcement Learning

In a series of papers culminating in [41], Andrew McCallum and company detail their development of a topic focused spidering algorithm motivated by reinforcement learning. The spider is part of the larger CORA system [41] which indexes and classifies computer science papers into a 70 leaf hand-crafted classification tree. The task of the spider is to find as many papers as possible while exploring as few pages as possible.

Their formal model is based on the classic Q-learning model [31] for reinforcement learning with delayed reward. In their idealized model, state is defined over the set of target pages and the set of all the web pages. At any particular step, the state of the spider is the set of all target pages not yet discovered and the set of all URLs previously encountered. An action is simply visiting a URL from the set of URLs referenced in the visited pages but not yet explored. This set of possible actions is called the *fringe*. Equivalently, we could define the state as simply a partitioning of the web graph, where the encountered URLs are on one side of the partition. The available actions would then be those unexplored URLS on the edge of the partition.

State transition function:

$$T : S \times A \rightarrow S \tag{2.17}$$

Reward function:

$$R : S \times A \rightarrow \Re \tag{2.18}$$

The policy:

$$\pi : S \rightarrow A \tag{2.19}$$

The value function:

$$V^{\pi}(s) = \sum_{t=0}^{\infty} \gamma^t r_t \tag{2.20}$$

The optimal Q function where γ is the geometric decay parameter:

$$Q^*(s,a) = R(s,a) + \gamma V^* \left(T(s,a)\right) \tag{2.21}$$

The optimal policy:

$$\pi^*(s) = \operatorname*{argmax}_a Q^*(s,a) \tag{2.22}$$

Given this model and assuming full observability, the task of developing an optimal spidering solution is, conceptually at least, trivial from this definition, one simply uses dynamic programming to propagate the rewards back and choose the action with the highest $Q(s, a)$.

However, when spidering on the real web, the state space is too large for dynamic programming hence an approximate policy is needed. McCallum et al. elect to solve for a near-optimal policy where they select that page from the fringe which is closest to a reward page. In earlier papers [50, 51, 40, 39] they consider using the optimal policy but appear to have rejected that as of little or no benefit.

The next decision (not explicitly stated this way in their papers) is to use an explicit exploration vs. exploitation step in which a portion of the web is gathered as a training corpus. They then use this portion to compute an approximate mapping from state-action pairs to Q values. Because the state space is so large, they train on only a subset of the space of state-action pairs: that portion where the visited partition consists only of a single unexplored URL and hence the fringe contains only a single page and thus only a single action is available.

For each state-action pair in the training set they calculate the Q values using the web graph of the training corpus. They calculate the near-optimal policy starting at each pair in the training set, run that for a finite number of steps, and then from that run, compute an estimate of the Q value for the pair. They also associate with each pair a vector of features. The features are term counts drawn from the *neighborhood* text in the pages linking to the URL in the state-action pair. They define neighborhood differently for different tasks. For example when spidering for postscript papers they use the entire text of the page, plus the anchor and nearby (whatever that means) text considered differently.

The next choice is how to train the regression. They elect to discretize the continuous space of Q values into a set of bins which become classes for a naive Bayes classifier trained on the feature vectors (apparently with smoothing from shrinkage [21] using the classification tree). The estimated Q value for a new feature vector is computed by performing an average of the Q values for each bin, weighted by that feature vector's class probabilities.

Their results, depending on the binning selected, are quite strong. They contrast their algorithm using a ($\gamma = 0.5$) with a breadth first search algorithm as well as with an immediate reward algorithm ($\gamma = 0$). In their test domain, they notice that in the second half of the search process, the immediate reward algorithm does better. They hypothesize that this is because by this stage in the crawl there are many links yielding immediate reward which the immediate reward crawler is optimized to find. In other words, when the target density is low, the future reward algorithm is more effective.

In my opinion, there is some obfuscation going on with their description. Since they are not using “real” Q values, but the near-optimal approximation based on choosing the action closest to a target and a relatively small number of bins (4 or 5), the Q values naturally bins (from large value to small) into a discrete set of values (assuming constant $R(s, a) = 1$ and $\gamma <= 0.5$):

$$\left(\sum_{0}^{\infty} \gamma^t, \gamma^0\right], \left(\gamma^0, \sum_{1}^{\infty} \gamma^t\right], \left(\gamma^1, \sum_{2}^{\infty} \gamma^t\right], \ldots \tag{2.23}$$

The question is why continue to use Q values instead of simply performing a regression directly on navigational distance to the nearest target? This is compatible with the near-optimal solution. Why not simply train for the near optimal solution? This isn't tested.

Despite this concern, the study is extremely promising as it shows, within at least certain domains, a machine can be trained using only local features to spider significantly more efficiently than breadth first search, also with a small stretch their regression can be viewed as simply a naive Bayes regression on navigational distance. The catch is their domains are all closed and extremely small when compared to the size of the live web.

Focused spidering through query modification

Srinivasan, Pant, and Menczer continue the research on the edge of the ARACHNID world with [48, 56]. They consider what characteristics of topics may influence how different search strategies perform. They build a topics list from leaves of the Open Directory Project (ODP) [15]. The URLs pointed to by the topic leaves are used as target URLS From these target URLs they spider backwards using Google's top 20 rated backlinks to a depth of two giving a maximum set of 4000 pages (10 target pages per topic). From this they randomly select 10 pages as seed URLs for the topic.

Topics are represented by the keywords associated with the ODP leaf node and the ODP human created anchor text surrounding the links to the external targets and text descriptions. This text is called the topic's *description*.

Topics are describe with respect to three measurements on either lexical features or link features. The characteristics are:

- Generality: The size of the "discourse" set. Lexically measured by the number of hits for keywords query across Google, AltaVista, and FAST. Graphically measured by the average number of inlinks of targets in the target set across Google and AltaVista.

- Cohesiveness: Lexically measured by average pairwise lexical TF-IDF topic normalized cosine similarity. Graphically measured by the number of target-set child links linking back into the target set, normalized by the number of child links.

- Authoritativeness: Graphically measured by the Kleinberg authority on target-set union child-of-target-set union top-20-parent-links from Google. Lexically measured by submitting the ODP description of each page to AltaVista to get top 10 links and performing HITS as above. The average authority score is the lexical authority.

The performance of the crawlers is measured by *recall* (target pages found vs. pages explored) (no discount, finite horizon) and with the average position of relevant pages in the crawl (linear discount, finite horizon).

They evaluate against several different search strategies, similar to those used in Cho et al.:

- Breadth First:

- Best-N-First: Fringe is ordered by cosine similarity and N pages are selected at each step for exploration before sorting. $N = 1$ and $N = 256$.

- PageRank: PageRank amplified with keyword counts in parent pages.

- Shark Search-N: Scores inherited from parents to depth $d = 3$ with relative importance vs. neighborhood scores of $r = 0.1$. $N = 1$ and $N = 256$.

They then analyze the results of the 100 topics against the seven algorithms and compute Spearman's rank correlation coefficient ρ.

They use this to draw conclusions including that exploratory searches ($N = 256$) perform better which isn't surprising since the horizon is so short. Specific topics are easier, etc.

In [48] Pant et al. focus on issues of exploration vs. exploitation. Their spider uses cosine similarity between the page and the query to rank order actions. Instead of only selecting one action at each time step, they vary the number of pages explored at each step. They call increasing the number of pages explored at each step *exploration* and decreasing the number *exploitation*.

Target Context Graphs

Diligenti et al. [17] present a best-first spider using an offline learning stage. Starting from a target set, they use the backlink capability of search engines to develop what they call a *context graph* around the target set. The context graph is built by spidering backwards through the parents to a fixed depth. If the context graph is too large, they use a sampling from the true graph.

They then separate the pages into classes based on distance to the target set and train naive Bayes classifiers using a standard bag-of-words model to calculate the class probabilities of arbitrary pages based on the text of the page. However, unlike in [51], they do not perform a weighted average of the resulting classifiers. They simply choose the class whose classifier yields the highest class probability. For each of the classifiers they assign a threshold probability, below which a page is marked as unknown. Then when spidering, they select the next page to explore by choosing that page in the fringe to spider next having the lowest estimated class (closest to target) and highest class probability within that class. They call a spider using this strategy *context focused*.

The authors compare their context-focused spider described above, a breadth first spider using the naive Bayes classifier only to recognize target pages, and a focused spider which also uses the naive Bayes classifier to recognize target pages and then attempts to exploit the target pages. They compare the spiders based on the ratio of retrieved target documents vs. number of downloads. Based on the limited results reported in the paper, they get dramatically better results from the context-focused spider ($50\% - 60\%$ better) than the focused spider and many times better than the breadth-first spider over a 12000 page run. They do not discuss the starting point for the evaluation runs. It may be the same dataset they train on which would explain the dramatic results.

2.3 Summary

In this chapter we discussed some of the core research exploring the structure of the web. Understanding the structure of the web is critical to both motivating the possibility of learning spidering strategies and for developing such strategies. In particular the ideas of the fractal nature of the web motivate us that if we can find patterns local to one portion of the web, or one portion of thematically unified cluster, these patterns will map onto the larger web or to the thematically unified cluster as a whole. In other words, if we can learn to spider from a small subset of a target class, we should be able to leverage that knowledge to search on the web as a whole for other exemplars of that target class.

We also presented a summary of some of the landmark and interesting developments in the field of web search and spidering. These various results demonstrated the feasibility of various spidering and focused crawling tasks as well as exploring algorithms for such tasks. The experimental results often either tested on the same portions of the web-graph they trained, searched starting from known targets, started extremely close to the targets being searched for, or combinations of all the above. The success of these techniques was motivational, the limitations of their results emphasizes the importance of our focus on the general spidering hypothesis: Can we learn, given training data in the form of a small sample of the web containing target pages, to conduct a directed search and find additional target pages faster than by performing a blind or uninformed search?

Chapter 3

Exploiting the Structure of the Web for Spidering

In prior chapters we discussed the necessity of understanding the structure of the web for efficient and effective spidering. In this chapter we discuss our contributions for understanding this structure. We also discuss our view of the web spidering task and present some of our early, motivational results on estimating minimum navigational distance and using that estimate for spidering.

3.1 Spidering and Navigational Distance

Let us consider the problem of web spidering from a very high level. Perhaps even higher than we looked at in the introduction. We will begin with a consideration of the ideal and constraints upon the ideal.

Before we begin, let us be clear on what the purpose of a spider is. For our purposes, a spider's job is to find target pages. Not only that, if we have a measure of the importance or quality of a target page, we want the spider to find the highest quality pages first. Finally we want the spider to use as few actions (retrieval of pages) as possible.

Under this scenario, the ideal spider will simply return a list of target pages while performing no actions. If there is a quality ordering on the target pages, then the ideal spider reports them ordered thusly. It is difficult to imagine how such a spider could be implemented without an oracle, that is, without global knowledge of the web graph *combined with* massive computational power.

Our next refinement is to limit the spider to only reporting pages which it has actually retrieved. Now the ideal spider requires an action for each target page but is still difficult to implement as it would have to guess the URLs for the target pages.

Consider a set of pages called a *seed* set. These pages will serve as a starting point for an ideal spider. The spider is restricted to only returning targets actually retrieved as above and it is restricted to only retrieving pages for which it has previously discovered the URLs and it can only discover URLs by finding them in retrieved pages or in the seed set. Now we can imagine an ideal spider which performs a best-first search always picking that page (oracle) which will help it optimally find target pages.

The problem we now face, is deciding what we mean by "optimally find target pages." The possibilities include finding the first target page with the fewest actions and finding the most target pages in a fixed number of actions. One can look at these problems in terms of *reward* functions. A reward function gives a value (payoff) to each page retrieved. For example each target page might give a reward of 1 and each non-target page a reward of 0. If our goal is to find target pages quickly we may want to penalize actions which do not bring us to target pages or equivalently we can discount the rewards from finding pages after multiple steps. Looking at the problem stochastically, we can consider the reward for an action in terms of *expected value* (cumulative expected discounted reward, etc.).

Regardless of which of these reward types we select, our ideal spider is simply choosing at each step the best page to expand next. It is still difficult to envision how this ideal spider might be implemented, however we might hope to approximate it. That is, we might hope to create a spider whose performance approaches that of the ideal spider, without having access to an oracle. So what does the spider do? Somehow or other it needs to select a sequence of actions to try and maximize its reward. If it simply tries at each step to pick that action the ideal spider makes, the spider may work very well, but it may also do very poorly because when it deviates from the ideal path, it no longer has a role model to imitate. Think of a robot playing a strategy game. It is trying to imitate an ideal strong player, however if the robot ever gets in a weak position, the ideal strong player may no longer be a good role model as it never plays in those situations.

This problem is ameliorated in our web scenario as actions not taken at a step are available at all future steps (note this changes under memory constraints). If the spider makes a bad choice, the previous good choices are still available. On the other hand one can imagine a spider that might select what it estimates to be a suboptimal action (perhaps with quality weighted random action selection) in the hope that thereby the quality of the entire sequence will be improved.

This situation is like that of an explorer navigating through the wilderness trying to reach a destination. Like the spider He has a good understanding of what directions to travel and how to choose, but he also has a friend floating in a balloon at 5-thousand feet who can see the best possible path to take, but can only tell the explorer the next step. If the explorer decides to follow his friend's advice, he may find himself stuck in a swamp (beyond his innate ability to navigate through) should the balloon be blown away.

Like the explorer, we have a fork in our possible spidering strategies: We can actively try to select a sequence of good actions such that the resulting reward approaches that of the ideal spider or we can simply try to pick the best action at each step we can. While these two paths are not mutually exclusive, they do shape how we consider the problem. In this work, we focus more on selecting the best possible action at each step.

If we think of this as a machine learning problem, then our actual spider wants to learn to behave as close to the ideal spider as possible. We want to find target pages, not just starting at target pages and exploiting the target rich neighborhood thereof, but starting at arbitrary starting pages on the web. In order to function effectively we must be able to select pages from the fringe such that we rapidly move towards target pages.

Because topics drift on the web very quickly, it is unlikely that we will have any contextual clues on the starting pages directly related to targets. We will need instead to look for patterns that help us move towards the target. Like the explorer, all we have is knowledge of our immediate surroundings and a memory of our past experiences on the web.

There are two broad strategies we can take. We could start from a clean slate and attempt to learn on-line but it might be a long time before enough target pages are found in order to learn a useful gradient. Instead we take an offline learning approach where we leverage a small sample of example targets and the community of pages linking to them.

So far we have decided to approximate the ideal spider with an offline machine learned spider which tries to choose the best possible action available at each step. Now we need to decide what exactly we are going to try and learn and from what. Let us consider the training dataset to be a portion of the web containing a set of known target pages and all of the undirected children thereof out to some depth. Let us also assume we have an ideal spider which can operate on this training dataset. (note that this ideal spider might be very expensive to implement even with full knowledge of the training set).

A direct machine learning approach would be to put the ideal spider in many different circumstances, observe the action the spider performs, and then try to learn from this. If our dataset contains n web pages then we have as many as n-factorial different circumstances (fringe + visited pages) to learn from, one for each partitioning of the graph. This is too large a number and hence will have to be severely pruned in practice. Just as Rennie and McCallum did in [51], one might choose to train on only those circumstances in which exactly one web page is known and its children in the fringe. We might want to train on multiple series of fringes in which the ideal spider starts at a single page and expands the fringe making ideal choices for some number of steps. We may want to simply select a random set of known nodes and the associated best selection from the fringe.

However we select the set of training examples, we next must decide how we will learn from them. What features of the training set will be used? Somehow we will need to select a small enough set of features to be tractable for training. Some of the features we select may be more global in nature and thus only estimable during the on-line portion when the spider doesn't have complete visibility. We could have the spider try to learn directly to select the best page to explore. If our ideal spider could provide a quality-ordered list of pages to explore at each step, the spider might learn how to perform a rank-ordering.

Or instead of using a direct approach, we might try to leverage an inductive bias and try to learn to estimate some feature (set) which, if exactly known, is a good predictor of the ideal spider's behavior. For example, a spider whose target is high in-degree pages, might want to use estimated PageRank as a predictive feature. One might even try to estimate the discounted reward function itself. It may turn out that a feature which is an extremely good cue to selecting the best action is extremely difficult to estimate well. Some features which are easier to estimate might not perform well even if known perfectly.

While a direct approach is attractive as it doesn't have such an explicit inductive bias, it is also potentially quite difficult. The primary focus in this thesis is on learning to estimate some value which serves as a good predictor.

In summary, we are focusing on developing a spider which uses offline machine learning to learn how to select the best page to explore from the fringe at each step. The prediction is made by picking that page which scores highest on an estimate of some function of a set of features whose exact values we expect to be correlated with our reward.

3.1.1 Evaluating the Spiders

Once we have trained a spider how do we evaluate it? The reward functions mentioned above are good gold-standards if the ideal solution can be tractably computed on the test corpora. Calculating the distance to target measures is trivially easy, however calculating the estimated discounted reward is very difficult and can only be done for sub-graphs with a max height of about four hops from the target set. In practice the distance to target and discounted reward gold standards perform very similarly so we don't typically need to calculate the discounted reward gold standard for the testing data.

Once a spidering strategy is learned we evaluate its performance across a range of short range and long range tests. The short range tests are performed by selecting a set of start pages a fixed height from known targets and exhaustively searching to a fixed depth. This fixed depth search is called a *search-cone*. The short range tests can also be started from the targets themselves. The short-range tests evaluate the exploration (finding targets rich areas) and exploitation (finding targets in target rich areas). The performance is evaluated by sandwiching the learned spider's performance between that of the gold standard spiders and that of the naive (breadth-first, depth-first, and random spiders). We also evaluate the spiders with long-range tests where the search cones are not limited to a fixed depth, but instead to a fixed number of page retrievals.

3.1.2 Experimental Environment

While our ultimate goal is for our spiders to operate on the "wild web", vagaries in our service from Google and our web connectivity quickly lead us to realize that an offline corpus is required. We are looking at using the 1.65 million page WT10g [16] corpus as well as a home grown corpus of approximately 2 million pages centered around the top ranked pages on Frederich Nietzsche from the Google Directory. Offline corpora also provide the benefit of repeatable experiments.

3.1.3 Regression on Navigational Distance

If we want to find target pages quickly then intuitively we should pick pages from our fringe which we expect to bring us closer to target pages. This is what Diligenti et al. [17] considered in their work on spidering with context graphs. They used a naive Bayes model trained to classify pages into a set of fixed depths or into an "other" category.

Rather than considering a naive Bayes classifier, we look at using a support vector machine with a linear kernel. A machine is trained to identify pages at teach depth vs. all other pages. When classifying a new page, the machine with greatest confidence is selected.

Our training set is built by trawling for pages out to a height four hops, through a sampling of the in-links, using the Google engine, from a set of starting nodes (high rank Nietzsche pages from the Google directory). The resulting pages are then post processed through a target identifier to remark all discovered targets to be root nodes in the web-graph. A shortest path algorithm is executed to relabel each page with its distance from the closest target. The target identifier used is a simple check for the existence of the word "Nietzsche". The trawl is limited to pages in the `.com`, `.net`,

`.org`, and `.edu` domains. The results from a trawl are then converted into a format compatible with McCallum's Bag-Of-Words (BOW) library [38].

The SVM regression is then used to generate a priority value for each link on the fringe hopefully favoring discovered links closer to target pages in a spider run. We explore several priority schemes. In the outline below, the following notation is used:

$$\begin{aligned}
\vec{d} &= \text{source document bag-of-words} && (3.1)\\
\vec{l} &= \vec{d} + 10(\text{anchor text bag-of-words}) && (3.2)\\
\texttt{svm}(\vec{v}) &= \text{results of each machine applied to bag-of-words } v && (3.3)\\
& \text{sorted from best to worst} && (3.4)\\
\texttt{svm}(\vec{v})[i].\text{weight} &= \text{confidence of the } i\text{th machine} && (3.5)\\
\texttt{svm}(\vec{v})[i].\text{distance} &= \text{class of the } i\text{th machine} && (3.6)\\
\texttt{svm}(\vec{v})[i].\text{atan} &= (\arctan(\texttt{svm}(\vec{v})[i].\text{weight} + 1.5)/3.0) && (3.7)
\end{aligned}$$

1. Using only the regression calculated distance computed on the term vector augmented with the link's anchor text, a link is scored:

   $$S_1 = \frac{1}{2^{\texttt{svm}(\vec{l})[1].\text{distance}}} \tag{3.8}$$

   This score is loosely based on the idea of a q-learning approach with a discounted reward of $\frac{1}{2}$.

2. Leverage the lower confidence machines by folding their scores into the calculation weighted by $1/2^i$ where i is the the order statistic on the machines ordered by margin:

   $$S_2 = \sum_i \left(\begin{array}{c} \frac{1}{2^{\texttt{svm}(\vec{l})[i].\text{distance}}} \cdot \texttt{svm}(\vec{l})[i].\text{atan}\cdot \\ \frac{1}{2^i} \cdot \mathcal{I}(1,-1)\left(\texttt{svm}(\vec{l})[i].\text{distance} < \texttt{svm}(\vec{l})[1].\text{distance}\right) \end{array} \right) \tag{3.9}$$

   where $\mathcal{I}(1,-1)(\text{cond})$ yields 1 if cond = `true`.

3. Use an estimate of the gradient, i.e., how much might following the link improve our situation using the calculations for S_2:

$$S_3 = S_2(\vec{l} - S_2(\vec{d}) \tag{3.10}$$

4. Amplified gradient plus distance (combination of S_3 and S_1) This estimate will be used extensively in our experiments and is referred to as the *augmented delta score*:

$$S_4 = (3S_3 + S_1) \tag{3.11}$$

5. Breadth-first search: Simply expand the fringe outwards in a breadth first manner.

6. Random walk on fringe:

$$S_6 = \texttt{random}(0, 1) \tag{3.12}$$

7. The above score functions plus noise.

$$S_7 = S_? + \texttt{noise}(0) \tag{3.13}$$

Issues

Because the trawl needs to be sampled, it is possible that the distance to target labels produced by the above strategy could be over-estimates. I have talked with Thomas Hofmann briefly about techniques where you know your training labels may only be bounds but that hasn't been considered in the experiment's yet. There are further discussions of this issue in Chapter 5.

Computing the SVM regression can take a very long time for large datasets. I believe that a voting algorithm between multiple sets of regression machines may be effective. I also believe that bagging/boosting will be helpful.

Approach 1 does not discriminate sufficiently between links. In practice every link on a page gets the same score. In fact almost always, the svm($\vec{l}$)[$*$].distance vector is identical for all links in a page. Note that the svm($\vec{l}$)[$*$].weight vector is not identical. Also even if score S_1 did differentiate between links in a page, there is only say 5 different possible scores so again we don't have a strong enough discriminator to find a gradient. S_2 tries to take advantage of the other classifiers, however it is extremely ad hoc.

The approach used for scoring function S_3 doesn't work well based on initial trials. The training data is waited heavily toward pages at depth 3 and the trained SVM machine gives better confidence on links going from distance 4 to distance 3 versus other transitions. Since this scoring method emphasizes taking those actions for which the machine is most confident, the spider tends to get stuck exploring pages at distance 3. The spider seems to make good progress early on, but it never zeros in.

Support Vector Machine Training Results

We gathered several datasets. In this section, we present the datasets and the training results for the support vector machines. Each dataset is characterized by the the seed set, the target depth searched to, the number of pages loaded, and the maximum number of parents sampled from each page.

sample:	max 15 | depth: 4 | pages: 5000
	Seed Pages:
`http://`	`www.swan.ac.uk/german/fns/fns.htm`
`http://`	`plato.stanford.edu/entries/nietzsche/`
`http://`	`members.aol.com/KatharenaE/private/Philo/Nietz/nietz.html`
`http://`	`infonectar.com/nietzsche/nietzsch.htm`
`http://`	`www.geocities.com/thenietzschechannel/`
`http://`	`users.aol.com/lrdetrigan/index4.html`

Table 3.1: Seed set and build criteria for Dataset One.

sample: max 10 | depth: 4 | pages: 4000

	Seed Pages:
http://	www.cwu.edu/~millerj/nietzsche/
http://	www.swan.ac.uk/german/fns/fns.htm
http://	www.pitt.edu/~wbcurry/nietzsche.html
http://	habitantes.elsitio.com/hpotel/
http://	plato.stanford.edu/entries/nietzsche/
http://	members.aol.com/KatharenaE/private/ Philo/Nietz/nietz.html
http://	members.aol.com/lrdetrigan/index4.html
http://	infonectar.com/nietzsche/nietzsch.htm
http://	www.geocities.com/thenietzschechannel/
http://	users.aol.com/lrdetrigan/index4.html
http://	www.friedrichnietzsche.de/
http://	www.rpi.edu/~macphm/Nietzsche/

Table 3.2: Seed set and build criteria for Dataset Two.

sample: all | depth: 4 | pages: 40000

	Seed Pages:
http://	www.cwu.edu/~millerj/nietzsche/

Table 3.3: Seed set and build criteria for Dataset Three.

This dataset was too large to train on and a support vector machine couldn't be trained for it. However it was used as a test set for the first two machines. Also note that while the target depth was 4, after identifying the additional targets discovered and calculating the new depths to the true target set, only depth 3 was actually reached in $40,000$ pages.

To give a feel for the quality of the machines trained, we present here the confusion matrices for machines one and two applied to the three data sets. In the confusion matrices, the rows give the actual classes (depths) and the columns give the predicted. A perfect classifier should give a diagonal matrix.

These preliminary results seem quite promising but are still early. The machines show a definite trend towards correctly classifying pages, or of classifying pages to within one step of the correct distance.

One significant problem is the particular regression machine used. For time reasons, code from McCallum's BOW [38] library was used to perform the regression.

true distance	0	1	2	3	4	total	percent correct	mse	me	inverse mse	inverse me
0	387	5	5	1	.	398	97.2	0.0854	0.0452	0.0121	0.0179
1	1	1268	35	.	.	1304	97.24	0.0276	0.0276	0.0019	0.0071
2	.	11	2271	23	.	2305	98.52	0.0148	0.0148	0.0005	0.0024
3	.	1	13	1278	.	1292	98.92	0.0132	0.0116	0.0003	0.0015
4	.	.	1	1	79	81	97.53	0.0617	0.0370	0.0005	0.0031

Table 3.4: Confusion matrix for machine one applied to dataset one.

true distance	0	1	2	3	4	total	percent correct	mse	me	inverse mse	inverse me
0	327	26	28	17	.	398	82.16	0.7312	0.3342	0.0886	0.1228
1	82	637	373	202	10	1304	48.85	1.0376	0.6817	0.0569	0.1644
2	61	279	1393	540	32	2305	60.43	0.5167	0.4360	0.0266	0.0820
3	29	120	553	565	25	1292	43.73	1.0209	0.7005	0.0370	0.1092
4	1	1	52	27	.	81	00.00	3.2099	1.7037	0.0371	0.1582

Table 3.5: Confusion matrix for machine one applied to dataset two.

true distance	0	1	2	3	4	total	percent correct	mse	me	inverse mse	inverse me
0	997	144	240	86	9	1476	67.55	1.3699	0.6220	0.1658	0.2274
1	285	2996	3478	1219	52	8030	37.31	1.1341	0.7917	0.0585	0.1858
2	421	4344	10945	4464	390	20564	53.22	0.5861	0.5072	0.0288	0.0989
3	70	1251	4744	3072	26	9163	33.53	1.1354	0.8165	0.0331	0.1228

Table 3.6: Confusion matrix for machine one applied to dataset three.

true distance	0	1	2	3	4	total	percent correct	mse	me	inverse mse	inverse me
0	295	18	39	11	2	365	80.82	0.8356	0.3753	0.1003	0.1363
1	17	452	188	104	10	771	58.63	0.9222	0.5746	0.0422	0.1282
2	31	186	850	252	11	1330	63.91	0.4556	0.3925	0.0251	0.0777
3	19	151	460	451	16	1097	41.11	1.1404	0.7612	0.0392	0.1201
4	5	41	87	72	1	206	00.49	4.2184	1.8883	0.0756	0.2109

Table 3.7: Confusion matrix for machine two applied to dataset one.

true distance	0	1	2	3	4	total	percent correct	mse	me	inverse mse	inverse me
0	355	5	4	1	.	365	97.26	0.0822	0.0438	0.0117	0.0175
1	.	745	26	.	.	771	96.63	0.0337	0.0337	0.0021	0.0084
2	.	5	1313	12	.	1330	98.72	0.0128	0.0128	0.0004	0.0021
3	.	1	9	1087	.	1097	99.09	0.0119	0.0100	0.0003	0.0014
4	.	.	.	2	204	206	99.03	0.0097	0.0097	0.0000	0.0006

Table 3.8: Confusion matrix for machine two applied to dataset two.

true distance	0	1	2	3	4	total	percent correct	mse	me	inverse mse	inverse me
0	1019	87	239	123	8	1476	69.04	1.5434	0.6545	0.1744	0.2289
1	542	2042	3332	1942	172	8030	25.43	1.6426	1.0304	0.0809	0.2375
2	781	2882	9517	6765	619	20564	46.28	0.7414	0.6053	0.0363	0.1103
3	175	725	4001	3528	734	9163	38.50	1.0051	0.7323	0.0329	0.1060

Table 3.9: Confusion matrix for machine two applied to dataset three.

The technique used (described above) is not well founded and there are many other possible techniques for training and combining the results of the classifiers for each depth. Also, because we are using the technique to rank-order pages for spidering, we find that binning the pages into depths is not sufficient. What we would like, and attempted to do with the various scoring hacks, is place the pages on a continuum of closeness to the target, so while in truth a page is exactly one distance, it might be recognized by the machine as being at some fractional distance such as 2.35 from a target. We need to study more principled methods of regression with support vector machines.

Implementation Issues

Because of the sampling strategy (crawl with re-depth sorting), the sample populations are strongly biased towards depth 3. This may have biased the quality of results. Future experiments will train on pages sets selected from paths so that each class will have an identical number of constituents (some may be duplicates). See Chapter 5, Section 5.2 for experiments with a spiders trained in a manner similar to this.

Spidering With the Machines

The SVM regression machines trained from dataset one and two above were used for spidering from a few different starting points. An anecdotal summary of results shows that the machines are indeed able to exploit a gradient. They can get stuck though for long periods (for example one run got stuck in a www.nagps.org set of surveys about philosophy departments (even though it did find some survey comments about Nietzsche).

Also, once a solid Nietzsche page was found, the spider was able to lock in and exploit it. Some Nietzsche hits are just “casual” cites in other contexts so those don’t trigger the exploitation phenomenon.

3.2 Summary

While these results are interesting, clearly much further work was needed. Also, we need to examine different kernels for the SVM as well as compare against the naive Bayes. To reduce the training time for the machines, we need to reduce the dictionary sizes further. Other than more dynamic stop-word pruning, we will select those words providing the highest information gain.

In addition to regression on navigation distance, it would also be interesting to look at regression on other values such as q-values such as in Rennie and McCallum's spidering through reinforcement learning.

Chapter 4

Experimental Framework

Studying the structure of the World Wide Web and studying the performance of spiders operating on the web is a daunting task. The web is changing continuously, access to the web is limited by myriad choke points, and experiments require repetition of tasks on the same or similar subsets of the web which can negatively impact service. For these reasons and others we need to take a snapshot of a portion of the web local to our experimental platform. Once we have isolated our portion of the web, we need to provide robust access to it.

For performing our spider research, we need to be able to run a large number of very similar experiments against our snapshot. For this we need a scaffolding for repeating the experiments. Finally we must select a set of tasks with which we will evaluate our spiders' strategies.

In this chapter, we discuss the selection and properties of the web snapshot we selected: the WT10G corpus; we discuss, at a high level, the software and hardware framework needed for the experiments and a description of the analysis tools used; and finally we discuss the selection of the tasks which we will use to evaluate our spiders.

4.1 The Corpus

Spiders operate on the world wide web. At first thought, we might think that for evaluating spider strategies, testing directly on the Web is sufficient. One might even think it necessary. There are several problems however with using the live web.

The first problem with evaluating spiders on the web is the problem of speed. Each page retrieval can take several seconds. Trawlers and large scale web indexing systems get around this problem by making many page retrievals in parallel. This isn't an option however for evaluating our spiders as we are deciding on the next action after every page retrieval. Since evaluating a single spidering strategy against a single corpus requires several thousand spidering runs retrieving millions of pages, evaluating on the live web simply isn't feasible.

A further problem with working directly on the web is that experiments are not repeatable. There are two aspects of this. First, the web is evolving continuously and each time a spider runs it may see a different set of pages. A tiny change such as the addition of a single URL to a page could dramatically change the results of a spider. One example would be if a URL was added linking from one target cluster to another. This new link would open up a whole new set of targets to subsequent spider runs. We strongly expect this type of phenomenon to happen, especially in emerging topics. Second, there are many bottlenecks in our connection to the web which we do not have control over. A high period of congestion could (and did in our early work) make days of experiments invalid. It felt almost like being Ben Franklin waiting for a thunder storm—we could only do our experiments when the internet weather was right.

An administrative problem in working with the live web is that in training our spiders we assume the availability of backlinks into pages. This isn't directly available on the live web without the use of search engines such as Google or AltaVista. Permission (which is rather difficult to get) is required from these services for automated querying.

A final, and perhaps most significant, problem with using the live web is related to the changing nature of the web as discussed above. Not only can we not repeat our experiments, but other researchers can not repeat our experiments. Repeatability of experiments is a requirement of experimental science.

Next, we might think of creating our own small snapshot of the web. The difficulty with this, aside from the small matters of coding and of running the trawl, is deciding what to include and what to exclude and of knowing whether or not the portion of the web selected is representative of the web as a whole. At a high level, some properties desired in a web snapshot are:

- graph properties like that of the web
- large and relatively diverse, yet
- not too large for wide spread use

The graph properties being like that of the real web is extremely important. For example, in TREC-8 graph structure techniques such as PageRank and HITS did not show promise. Bailey et al theorized [3] that this is because the WT2g corpus used in TREC-8 suffers from a dearth of inter-site links, significantly damaging the performance of connectivity based algorithms Also, due to issues of copyright to the actual pages retrieved, we would not have clear rights to distribute our corpus to other researchers allowing them to reproduce our work.

A third option, and the one we decided to use, is to use a prebuilt snapshot of the web. We needed to find a high quality web snapshot that was proven to be web-like. Fortunately, there was the WT10g [16] corpus. This corpus was used by National Institute for Science and Technology (NIST) for their 2001 Text REtrieval Conference (TREC) competition and is available for a moderate price on CD from the Cooperative Research Centre for Advanced Computational Systems (CSIRO). The corpus contains 1.7 million web pages taking a total of 10 gigabytes. The creators of the WT10g corpus focused on avoiding the following deficiencies found in earlier corpora, particularly WT2g:

- Binary pages
- Duplicate pages
- Non-English pages
- Missing home pages
- Generated pages

Missing home pages in a web corpus used for spidering experimentation could be very damaging. Home pages provide key structure and high quality links and should be very important for spidering.

In addition the WT10g creators aimed for the following properties:

- rich in inter-server links
- containing all pages, especially homepages, from a set of servers

For various reasons, including time constraints, they drew their initial pages from the 100 gigabyte VLC2 corpus which was in turn drawn from the 320 gigabytes 1997 Internet Archive. After filtering for binary, duplicate, non-English, and generated pages, the 18 million page VLC2 corpus was reduced to about 11 million pages. They then filtered out all servers in VLC2 containing less than 5 pages and then binned the servers by the number of pages they served. The servers in each bin were rank ordered taking into account in-link, out-link, relevance (proportion of pages scoring in the top 500 in the 10,000 queries in the TREC-8 Web Track Large Task), and the ratio of acceptable (not rejected due to binary, duplicate,...) pages to total pages. Then the servers were selected in rank order from each bin in proportion to the number of servers contained in the bin. This process ensured power-law distribution of server size, as well as good inter-server connectivity. Furthermore, Ian Soboroff [55] showed that many of the properties desired in a snapshot of the web such as the number of connected components or in-degree distribution are very close to the web as a whole.

In Figure 4.1 we can see the expected power-law distribution of strongly connected components. A strongly connected component is a collection of web pages from which each of the pages can reach any of the other pages in the component following URLs on the page. In the limit at least, any page in a particular strongly connected component will have paths connecting to any page reachable from other pages in the component. Also, due to the exponential growth experienced when expanding a search outwards from a starting set of seed pages, the set of pages found at a particular depth from different pages in the component will rapidly converge to being identical as the depth increases (modulo one or two hops as the depth gets large, i.e., greater than say 9).

Another aspect of the true web we would like to carry over into our experimental snapshot is the diameter. Remember that Albert et al. define the diameter as the

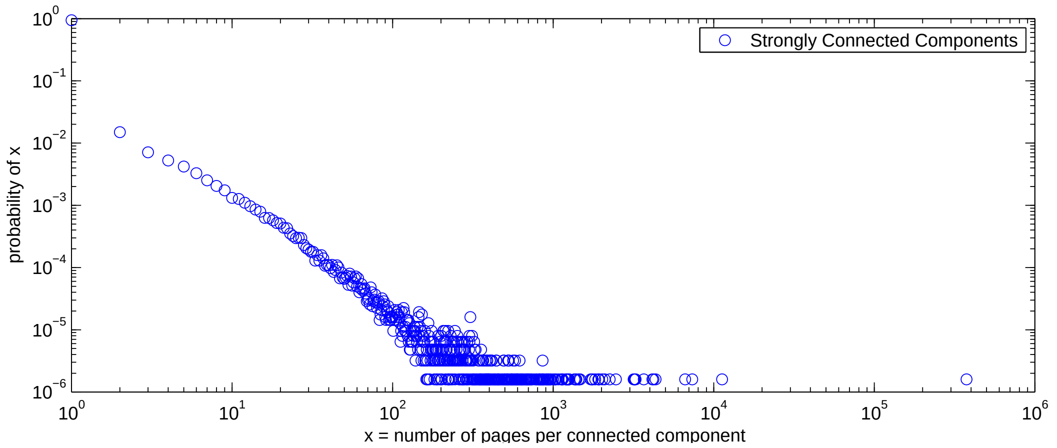


Figure 4.1: Distribution of strongly connected components WT10g

average over all pairs of vertices of the smallest number of links to follow between the vertices. For the real web they estimated the diameter as 19 and gave the equation $< d >= 0.35 + 2.06 \log(N)$ (where N is the number of pages) as an estimator for the diameter [2]. Note that I measure the diameter of the WT10g corpus at just over 9 while their model would predict a diameter of a touch over 13. As suggested in [55] to explain other structural anomalies, this may be an artifact of the bias of selecting by server by size. The measured diameter of 9 would suggest a web size of only $20,000$ nodes using Albert et al's model.

We can see in figures 4.4 and 4.5 that the WT10g corpus shares a similar butterfly structure [8] with the web as a whole. The center circle, or thorax, is the largest strongly connected component (SCC) in the web graph. The left wing is the portion of the web from which the center is reachable but which isn't reachable from the center and, conversely, the right side represents that portion of the web which is reachable from the center and from which the center isn't reachable. Unfortunately my artistic skills are not up to the level needed to draw a butterfly like in Figure 4.4. In figure 4.5, the peak point at $x = 0$ represents the thorax (375 thousand pages) with the negative x values showing the left wing (224 thousand pages) and the positive x values showing the right wing (499 thousand pages). This correspondence between WT10g structure and true-web structure is important as it shows that, at least at a

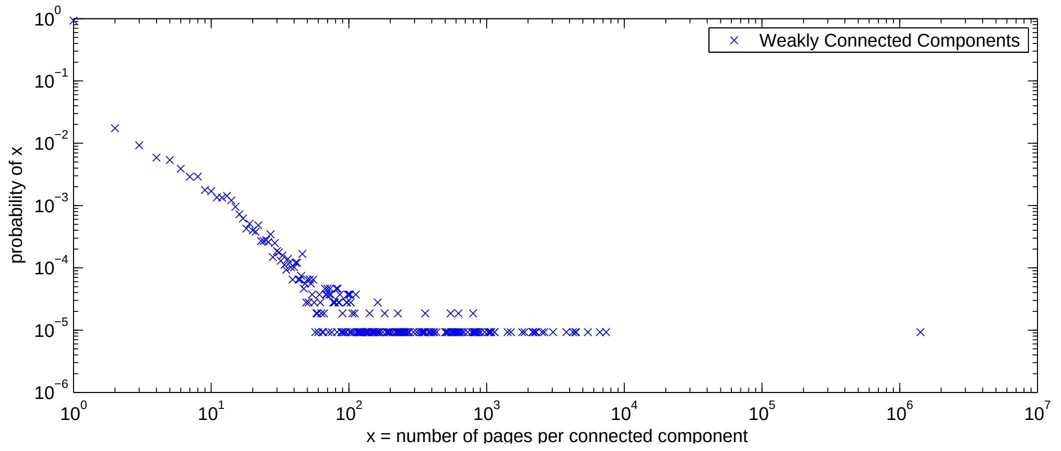


Figure 4.2: Distribution of weakly connected components WT10g

gross level, the WT10g corpus is connected similarly to the web as a whole.

Finally, since CSIRO distributes the WT10g corpus and not us, we do not need to worry about the copyright issues provided we adhere to certain guidelines provided by CSIRO.

4.2 The Queries

A difficulty with using the real web or our own snapshot of the web is identifying all of the pages satisfying some target query. Since our research is not focused on target recognition, but on the search process, spending time developing target identifiers is a distraction. This would force us to either select targets from known target lists such as DMOZ or to use very simplistic target definitions such as a threshold on cosine-angle similarity. The risk of using the known target lists is that they are not exhaustive. There are many pages properly belonging to the target class yet not explicitly listed in the target set. This means that while our spiders are spending their effort finding real targets, these targets won't be recognized as such. The option of using a similarity measure for defining target classes would require that every page in the corpus be tested against every target class. It also defines a very simplistic notion of what a target page is. Also, the task of hand-picking our own query tasks raises

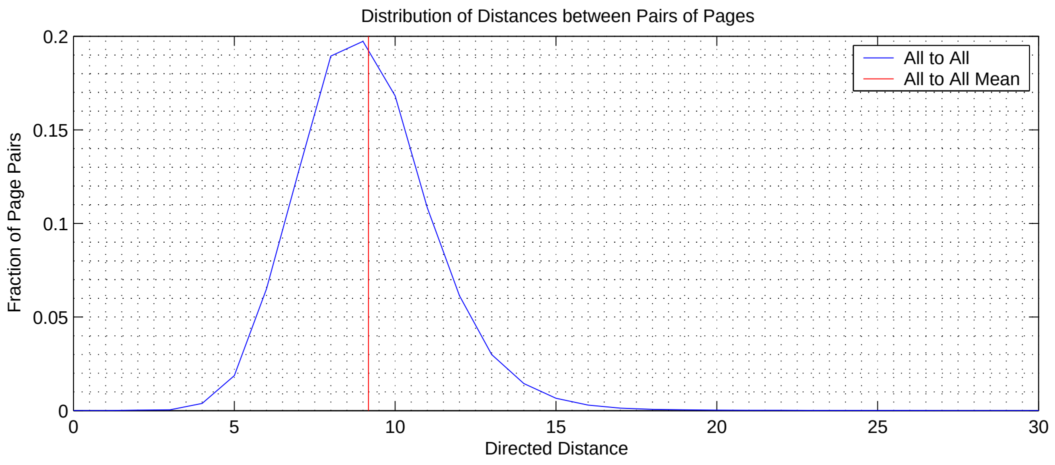


Figure 4.3: Distributions of the Diameter of the WT10g web-graph. The mean directed page-to-page distance is about 9.15. 9.25.

- **Description:** Find documents that describe the roles estrogen plays in the human body

- **Narrative:** Relevant documents will describe the positive effects of estrogen's presence or the negative effects of its absence. Discussions of the benefits of hormones in general are not relevant.

Table 4.1: TREC Task 544

problems with hidden biases...are we designing our queries to show off the features of our spiders?

Fortunately, the TREC competition provides a set of 50 ad hoc queries against the WT10g corpus. The queries have clear semantic definitions and have high quality labellings of the targets. For each query, there is a list of all the pages considered to be targets. These lists have been checked repeatedly: Each page identified as a target by contestants in the TREC competitions but not included in the original target list were hand checked by a panel of human judges. This means that if there exist any unlisted targets in the WT10g corpus, they must be so unusual that none of the techniques used by any competitors were able to find them.

Table 4.1 shows an example of one of the TREC queries. This task, finding documents describing the roles estrogen has in humans, has 324 target pages in the

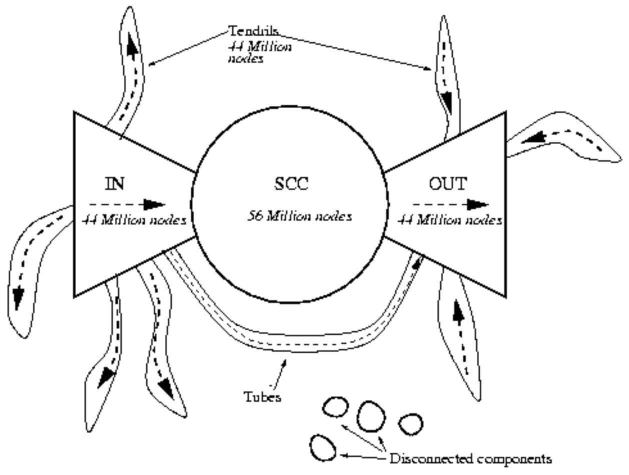


Figure 4.4: The butterfly diagram for a sampling of the world wide web by Brodar et al [8].

Task	Name	Size
541	Weather Instruments	372
544	Estrogen's Roles	324
519	Frog Habitats	149

Table 4.2: Tasks and Targets

WT10g corpus. This number of target pages, approximately 0.02 percent of the corpus, is small enough to be challenging yet large enough to allow gradations in performance between spidering techniques to be seen. If the number of targets is too small, then the task may be too difficult making it hard to discriminate between the performance of the various spidering techniques, i.e. there are so few targets that in most executions the spider will only find either one or no targets forcing the median targets found to be 0 or 1 making comparison difficult. The number of target pages for the TREC tasks ranges from 2 to 372 with a mean of 67.26. From the complete set of queries we arbitrarily select three queries with high target counts, the first, third, and fifth largest having 372, 324, and 149 targets each (Table 4.2), respectively. These are the queries we will focus on for our spider evaluation.

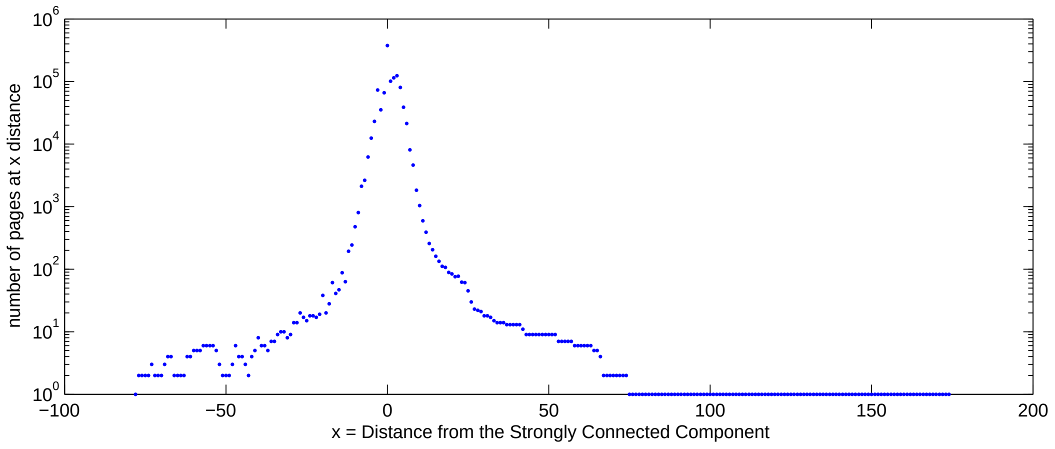


Figure 4.5: The butterfly diagram for the WT10g corpus. Negative x values are distance *to* the strongly connected component, and positive x values are distance *from* the strongly connected component.

4.3 The Target Environment

Now that we have selected the WT10g corpus and the TREC 2001 ad hoc query set, and specifically our three queries in Table 4.2, we examine the environment in the web-graph where these targets are found. Knowledge of this environment is critical for predicting and understanding the performance of spiders on that task.

What characteristics might we be interested in? One characteristic we are interested in is the incestuousness of the data, that is what portion of the descendants of the targets reachable at a certain depth are also members of the set of ancestors of those targets. We might also be interested in the reachability of targets. That is, how is the number of targets reachable from ancestors at each depth distributed?

The previous two characteristics involve the graph structure around the targets. We might also be interested in the vector space characteristics of the terms found in the target pages. For example how do the dictionary sizes and term distributions vary for target sets vs. the dictionary as a whole vs. ancestor pages vs. descendant pages.

In this section we step through the above characteristics showing results from our three TREC queries (Table 4.2) in the WT10g corpus. These characteristics will be further discussed when we analyze actual spidering results in a following chapter.

4.3.1 Pan-TREC properties

Each of the three TREC queries we explored have different local properties when we examine the target pages in the context of the WT10g corpus. In this section we will explore in detail some of these properties. We examine some of the properties of the three test queries as compared with all of the TREC queries. For others, we explore only the properties of the three queries presented in Table 4.2. The properties which we compare across all queries are:

1. Number of targets in the query set
2. Distribution of domains for each query
3. Mean cosine-angle distance between query text and target pages for TREC queries against WT10g corpus.

These properties can be loosely divided into two sets: Those dependent on the graphical structure of the web in the context of the query targets; and those dependent on the textual content, or vector-space representation, of the query pages and their neighbors. We will discuss these separately below.

Graphical Properties

Figure 4.6 shows the number of target pages, i.e., pages whose text matches the textual description in the query, for each of the TREC queries. We can see that the three queries selected are ones with high target counts. The three queries used were chosen arbitrarily from the queries with high target counts. High target count queries were chosen for exploration because we expected to be able to identify finer grade distinctions in quality between spidering techniques if we had a denser distribution of targets in the corpus. In other words, if only a small number of targets are reachable from some starting point, a spidering run might either find one or no targets during a short run. This makes it very difficult to say by what percentage a particular technique performed better than an other. However, if there are 50 or 60 targets

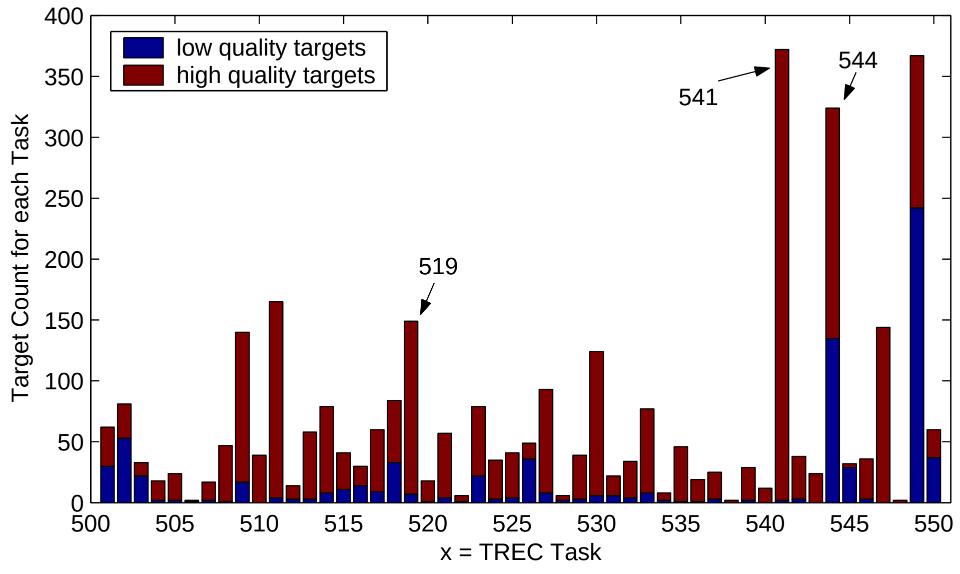


Figure 4.6: The distribution of target counts for the TREC queries. The three tasks explored in detail are indicated.

reachable from some start page, we can expect a large spread in performance across runs, facilitating our comparisons.

In addition to the number of targets for each query, it is also important to have a feeling for how many different domains the targets for the queries are distributed across. This gives us a quick feel for how clustered our targets are. We can see this distribution of targets to domains for the TREC queries in Figure 4.7. Notice the highlighted entries for our three queries of interest.

Textual Properties

In addition to the graphical structure around the target pages, we are also interested in the textual environment of the targets. Since we represent the textual features of the pages with the classic vector-space representation, a natural first feature to examine is the cosine-angle distance between the targets and the TREC query text.

In Figure 4.8 we depict the mean cosine-angle distance between each TREC query and the vectors for each of the target pages for the corresponding query. Each of the tasks has three bars. The first shows the mean cosine angle for the low quality targets, the second shows that of the high quality targets, and the third shows the mean for all

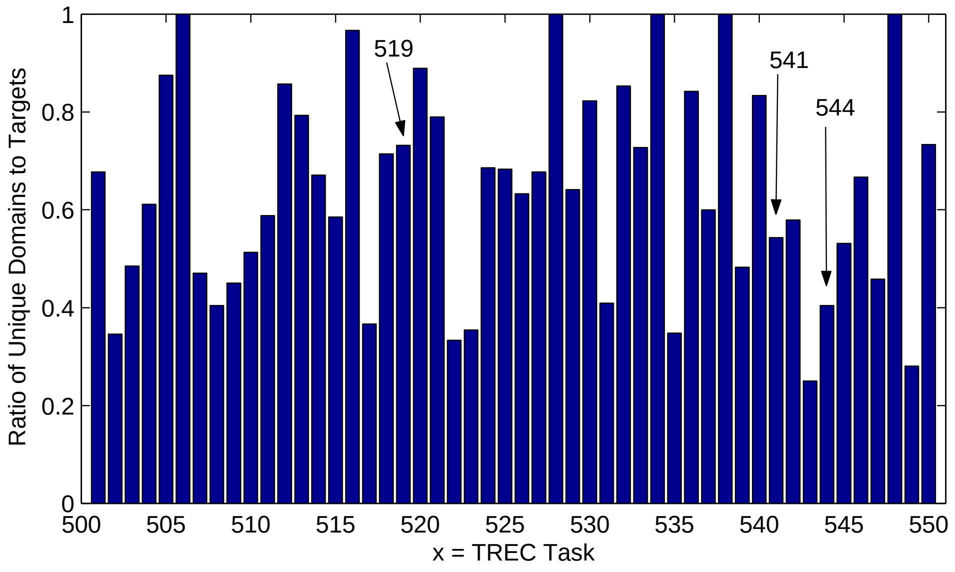


Figure 4.7: The distribution of domains for the TREC queries (normalized by the target counts). The three tasks explored in detail are indicated.

the targets. Notice that the high quality targets almost always have a higher cosine angle.

Graphical Properties

When building the training set for a query in our experiments, we exploit the automatic availability of backlinks. On the web, these backlinks are available from services such as Google and AltaVista for much of the web. In the WT10g corpus, we have direct access to the backlinks as they are an explicit part of the dataset. Under some circumstances however a spider won't have access to the backlinks. This is where our measure if incestuousness comes in. If the percentage of descendant pages which are also ancestors is high enough, then we can approximate the availability of backlinks by performing a breadth-first search through the descendants of our initial target set. Results on the TREC queries indicate approximately 65 percent of the ancestor set of height four is recoverable by expanding the descendants to depth six.

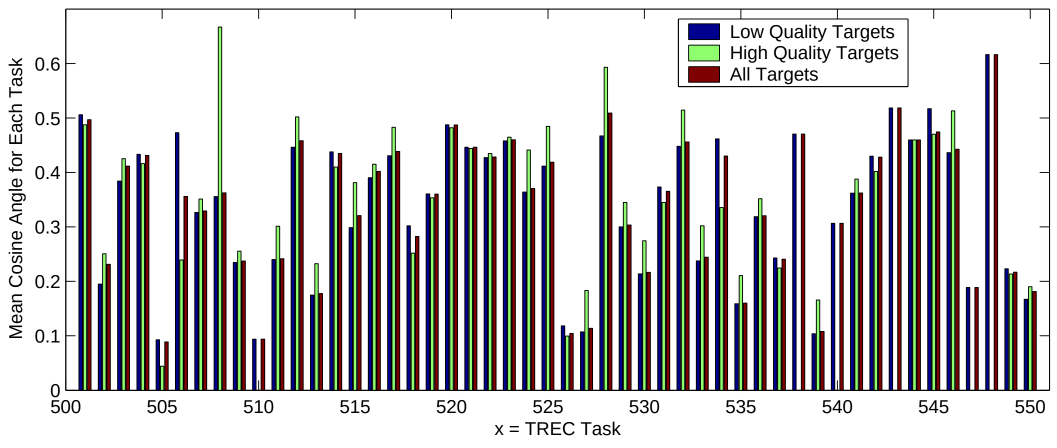


Figure 4.8: The distribution of cosine angle distances between target sets and the TREC query texts.

4.4 Performance Measurements

At a high level, the goal of the web spider is to find target pages as quickly as possible. But what do we mean by "as quickly as possible?" There are two related approaches: measures of the time (actions) to first target and measures of the total number of targets found.

In the planning community one often talks about the *horizon*, either infinite or finite. The *horizon* refers to how far ahead one looks when determining the "payoff" or reward. Since we are working in a domain with a finite set of target pages and we are only allowing the spider to retrieve a fixed number of pages, we won't consider infinite horizons. Once we've decided what to measure regarding spider performance, we need to consider what we will compare the performance to. This is discussed further below.

In addition to directly measuring the performance of the spidering strategies, we might consider the strength of the learned search heuristic itself. In other words, how well does the heuristic perform at ranking and/or ordering the web pages, independently of its use in the spider.

Classically, web-spidering research has focused primarily on evaluating spiders starting relatively close on the web graph to known targets. In our experiments

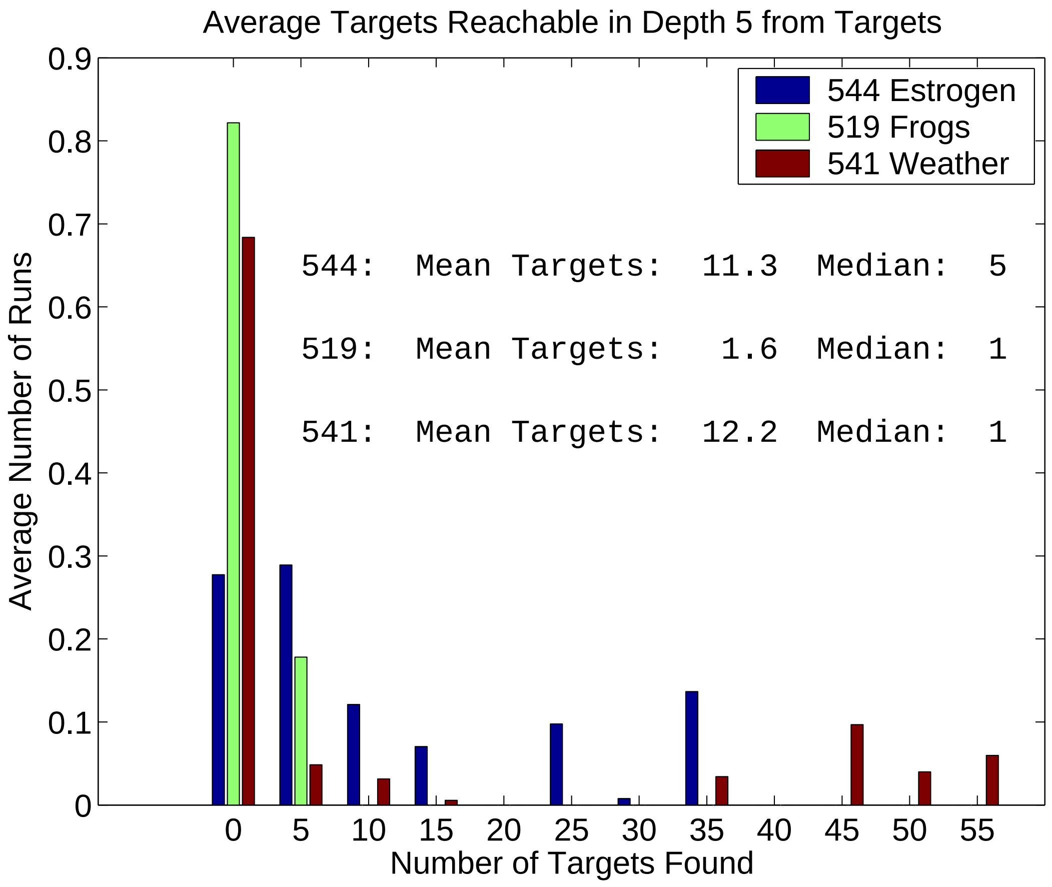


Figure 4.9: Targets reachable in depth 5 starting from targets

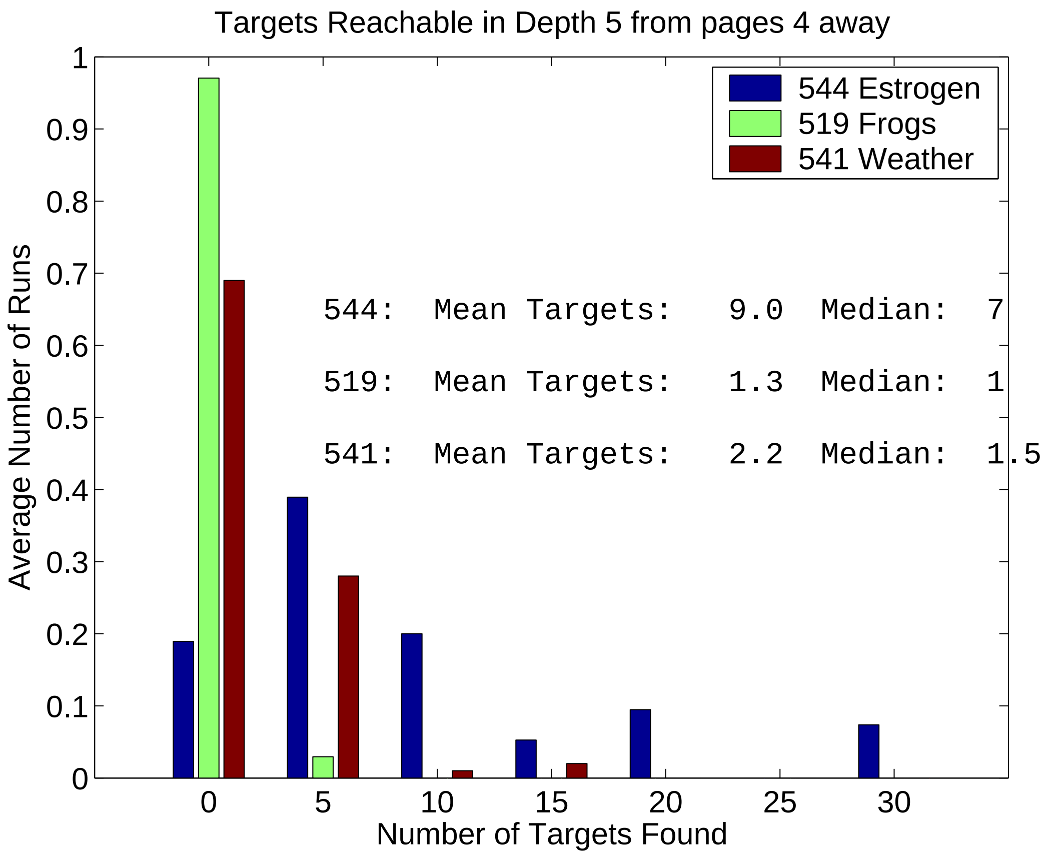


Figure 4.10: Targets reachable in depth 5 starting from sample of pages 4 away

discussed in detail in Chapter 5 we present short-range experiments similar to those experiments discussed in Chapter 2 as well as longer range searches.

The short-range searches evaluate the ability of the spiders to search in domains similar to what they were trained on. They are tend to be searching within the more densely connected neighborhood. Remember that the web-graph is small world. That is, it is sparsely connected globally, densely connected locally, and yet on average short paths exist between any two pages. In our short-range experiments we constrain the spiders to only explore within a fixed distance from the starting page, trying to keep the spider in that densely connected region. The spider is started at some start page selected to be a particular distance from the closest target and then the spider is allowed to search to exhaustion a cone of fixed depth. In the experiments in Chapter 5 we used starting heights of zero, three and four with search depths of three, four, and five. This let us test both short-range exploration and exploitation.

On the other hand, in the long-range searches we test the ability of the spider to find its way through a vast space of unrelated pages, trying to find a path into a target rich area. We select start pages at some height from the closest target and then allow the spider to search for a fixed number of page retrievals. The initial start height varies between five and eight depending on the experiment. Note that a start height of eight is almost the diameter of the WT10g corpus. We typically allow a search horizon of 10 to 20 thousand pages looking for a few hundred targets in a total space of several hundred thousand pages.

4.5 Experimental Software

Figure 4.11 shows the basic software components for the spider toolkit. The `Spider` module provides the main search loop for the spider. Various layered components are provided allowing the user to design a spider with the particular properties needed. It also manages all of the resources used. The `Spider` module communicates with an `http` proxy. The `Proxy` module hides the issues with accessing the real Web or the WT10g corpus or other homegrown corpora. The `Spider` module stores the information about the Web it discovers in the `Graph` and `PageData` modules. The `Graph` module leverages the BOOST Graph Library [54] to allow efficient representation and

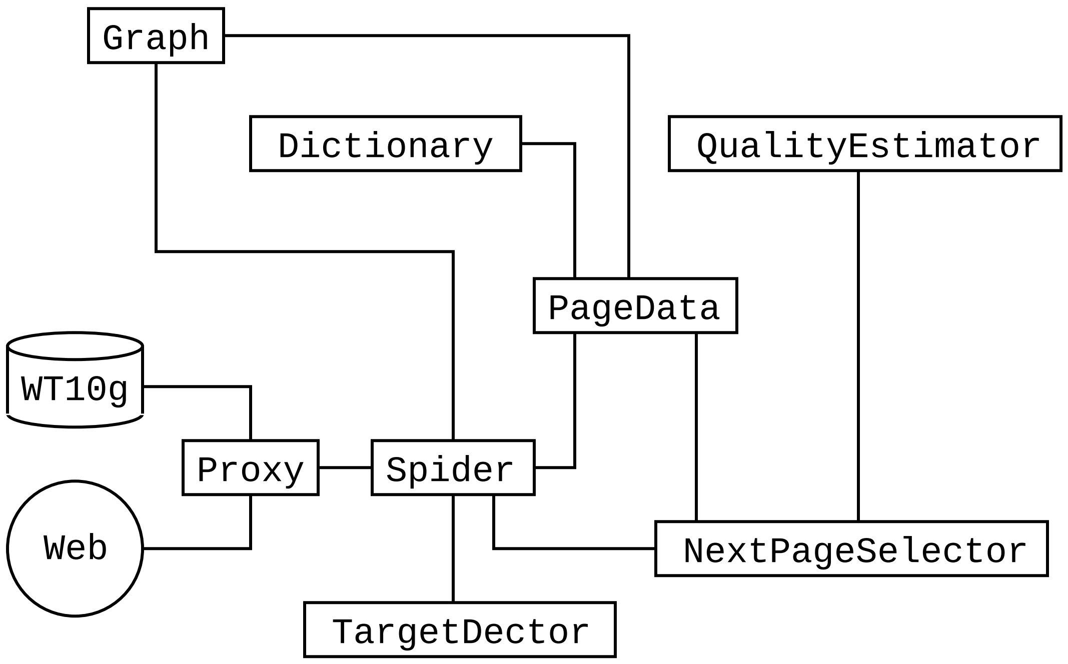


Figure 4.11: Distribution of the number of targets per domain

ready access to a broad collection of graph algorithms. The `Spider` module derives its search strategy from the `NextPageSelector`. The `NextPageSelector` manages the fringe and selects pages from the fringe according to the `QualityEstimator`. Different pluggable `QualityEstimator` modules provide DFS, BFS, Random, SVM, ... The `QualityEstimator` abstracts the spider from the machine learning technique used. By writing a small module, many third party machine learning modules can be attached. Modules have been written for such systems as C4.5, BOW, libsvm, and svmlight. The `TargetDetector` isolates the task of identifying target pages. Modules have been written using a thresholded inner-product similarity as well as an oracle look-up into a known target list.

The modular architecture lets us readily adapt a spider for different environments. We can plug-in a new parser for backlink detection from Google or Alta Vista, we can plug-in different filters to limit the spider's environment, e.g., avoiding binary pages or staying in the `.gov` domain. With a command line switch, we can switch from the live-web to the WT10g corpus.

4.6 Experimental Hardware

Our experiments with spidering on the web required relatively large amounts of compute power and storage. This may seem counter-intuitive when one thinks that the purpose of a web-spider is to provide a relatively low-resource agent for efficiently exploring the web. However, in order to evaluate the performance of non-deterministic spidering strategies many spidering runs from the same starting point are required, only varying the seed values for the random number generators. We also need to run each test from many starting points. This quickly explodes into thousands of test-runs for each experiment. Each of the test executions takes time ranging from a few seconds to several minutes and can generate several megabytes of trace data. In this section we outline the hardware used to run our experiments and analyze the results.

4.6.1 Compute Cluster

Our primary computational resource was an cluster of 20 dual AMD Athlon 1800+ workstations each with 2 gigabytes of ram. The machines are connected on a private 100Mb backbone. While the cluster is running the openMosix [57] clustering system, we don't take advantage of the openMosix features.

4.6.2 Proxy Server

We developed a proxy allowing us to treat the WT10g corpus as simply a view into a snapshot of the web from 1997. Because of the large indices required (600MB resident), we placed the proxy on a dedicated system, as well as scattered on select nodes inside our cluster. The proxy server allows the same software to work on both the real web and on the corpus, saving significant development efforts.

4.7 Summary

In experimental research, repeatability of experiments is crucial. This, combined with issues in dealing with the internet "weather", drove our switch from experiments on the live web to using the WT10g corpus. An added benefit of the switch was access to the TREC queries with human vetted labeled target data.

In this chapter we examined the properties of this corpus and the TREC queries in the context of our spidering experiments. We outlined the performance measurements we use to evaluate the spiders as well as the software and hardware solutions developed for our test harness allowing for the efficient performance of hundreds of thousands of spidering experiments.

Chapter 5

Results

In Chapter 4 we discussed the web environment our spiders are exploring and the search tasks we will evaluate our spiders on. In this chapter we present the spidering results obtained.

Our discussion will focus on tasks performed against TREC task number 544, the role of estrogen in the human body. Once we have detailed our experiments in the context of task 544, we will present the results for our other two tasks, 541 and 519. As we noted in our discussion on the TREC tasks, the web environments of the three tasks (519, 541, 544) are dramatically different. The impact of these differences on our spidering performance will be discussed.

We also briefly discuss some of the other spidering and machine learning strategies we investigated, but didn't pursue to completion.

5.1 Spidering for Estrogen

In this section we explore our spidering results with TREC task 544:

- **Description:** Find documents that describe the roles estrogen plays in the human body.
- **Narrative:** Relevant documents will describe the positive effects of estrogen's presence or the negative effects of its absence. Discussions of the benefits of hormones in general are not relevant.

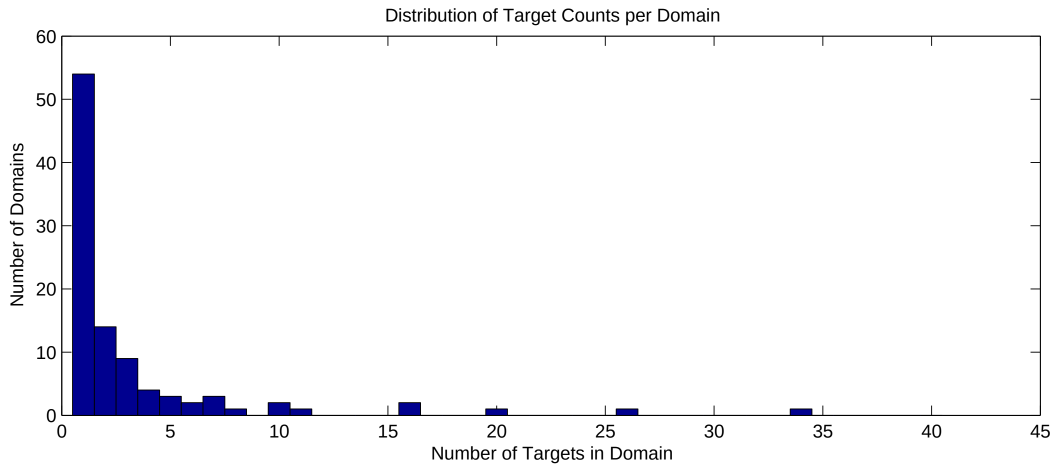


Figure 5.1: Distribution of the number of targets per domain

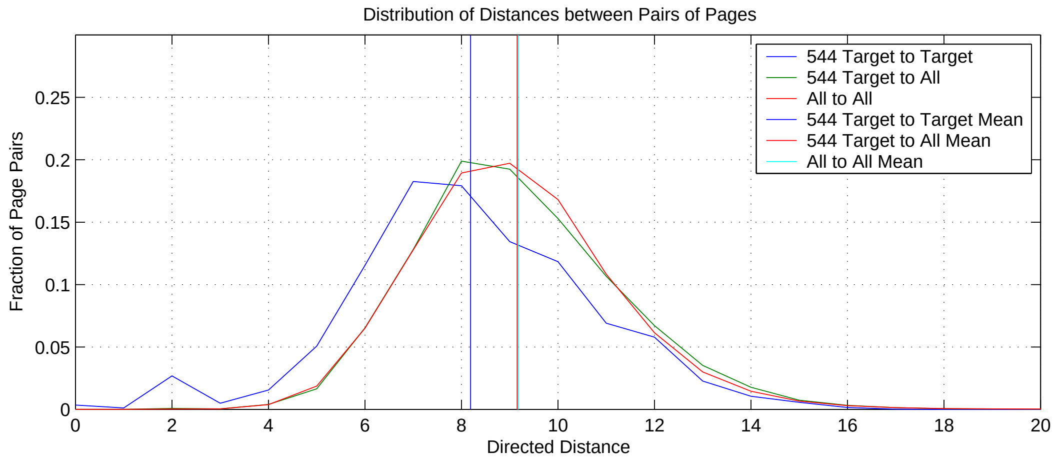


Figure 5.2: Distributions of the Diameter of the Web-graph as observed from the TREC Task 544. The mean target-to-target distance is 8.20 and the mean target-to-all distance is 9.15.

```
http://www.rmca.org/Gazette/spay.htm
http://www.specialtylabs.com/testguid/testfils/BL5.HTM
http://www.specialtylabs.com/testguid/testfils/BL56B.HTM
http://www.sport.ussa.edu/pubs/fitness.htm
http://www.stjoseph.org/~swaszak/nocahrt.htm
http://www.streetlink.com/novn/
http://www.summitmed.com/menopaus.htm
http://www.telemedical.com/~drcarr/Telemedical/CWS/estrogen.html
http://www.wealth-builders.com/frcream.html
http://www.winternet.com/~bobbi/mcclean.htm
http://www1.tpgi.com.au/users/jon/gear.html
http://www9.yahoo.com/headlines/970106/health/stories/estrogen_1.html
http://yarra.vicnet.net.au/~wise/HRT1.htm
```

Table 5.1: Training targets selected against TREC task 544

As discussed in Chapter 4 this task was selected for its large number, 324, of identified target pages spread across 98 domains. This is the third largest TREC task. Figure 5.1 shows the distribution of targets to domains. At one end we have over 50 domains with one target each and at the other end, one domain with 34 targets. In Figure 5.2, we see the distribution of target-to-target distances. The mean target-to-target distance is 8.20 links. The distribution of target-to-all pages distances are also shown (mean 9.15) as well as a sampled estimate of the overall distribution. Note that the distributions for target-to-all pages is very similar to the overall distribution and the means are virtually identical. The mean value of the overall distribution, 9.15, is what is often called the *diameter* of the graph. To put it in context, the overall diameter of the true web is estimated at about 19 [2].

From the 324 target pages in TREC task 544, we arbitrarily picked ten targets from http domains, e.g. `http://www.rmca.org`, containing only one or two targets pages. This restriction was made to keep the training set isolated from the testing set. After trawling the initial target set we found a few more targets. We also found that a few of the pages had significantly ill-formed HTML, bad enough to prevent automated parsing. After adding the discovered targets and dropping the ill-formed targets we had a net gain of three additional targets. This resulted in the target set in Table 5.1.

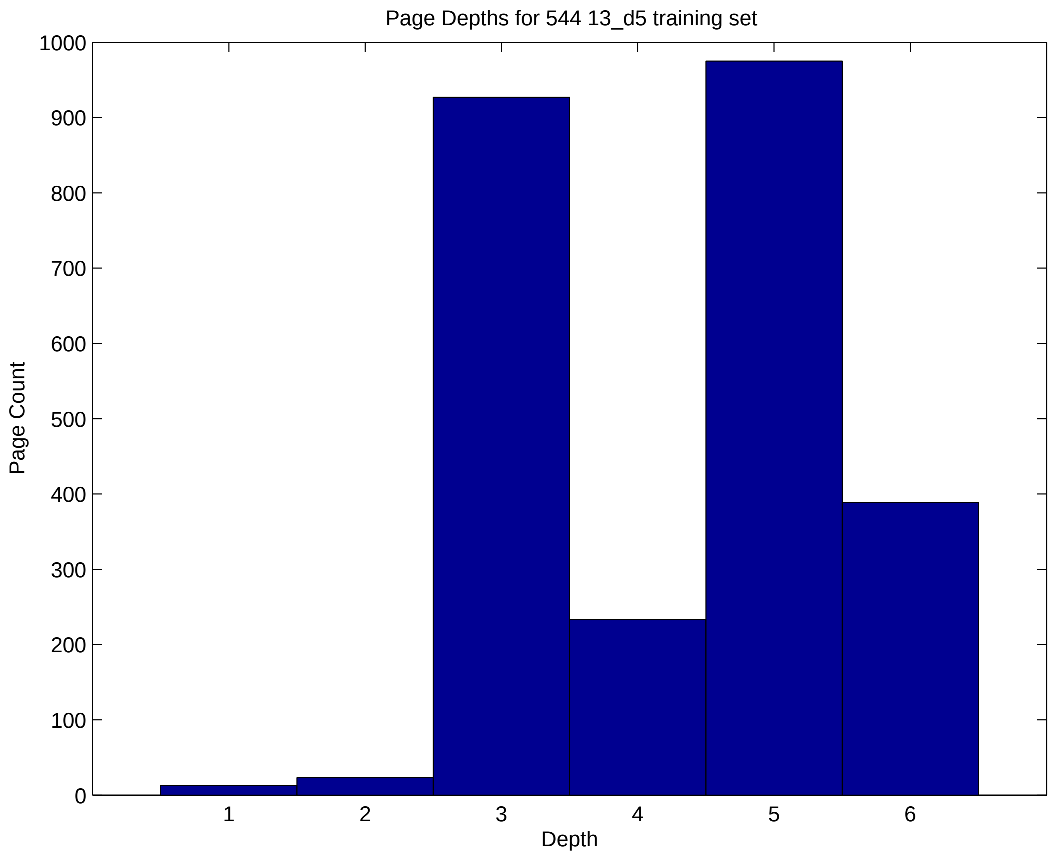


Figure 5.3: Distribution of pages at each level in the parent search-cone for the training set. The hump at height-3 is consists primarily of 817 pages in only two domains.

If we hadn't picked the training targets from hosts with low target counts we would have two problems. First, we might over-bias our results towards success as one might expect the other targets sharing a particular host to be very similar and hence easier to find. Looking at Figure 5.1 we can see that if we had picked any targets from some of the larger domains, we would have instantly exploded into quite a large training set. Second, when trawling the parent search-cone we would have the problem of finding additional targets due to the high connectivity within the domain which would rapidly grow our target set. In any case, this set of thirteen target pages formed the seed for our training target set for virtually all of our experiments on the 544 task.

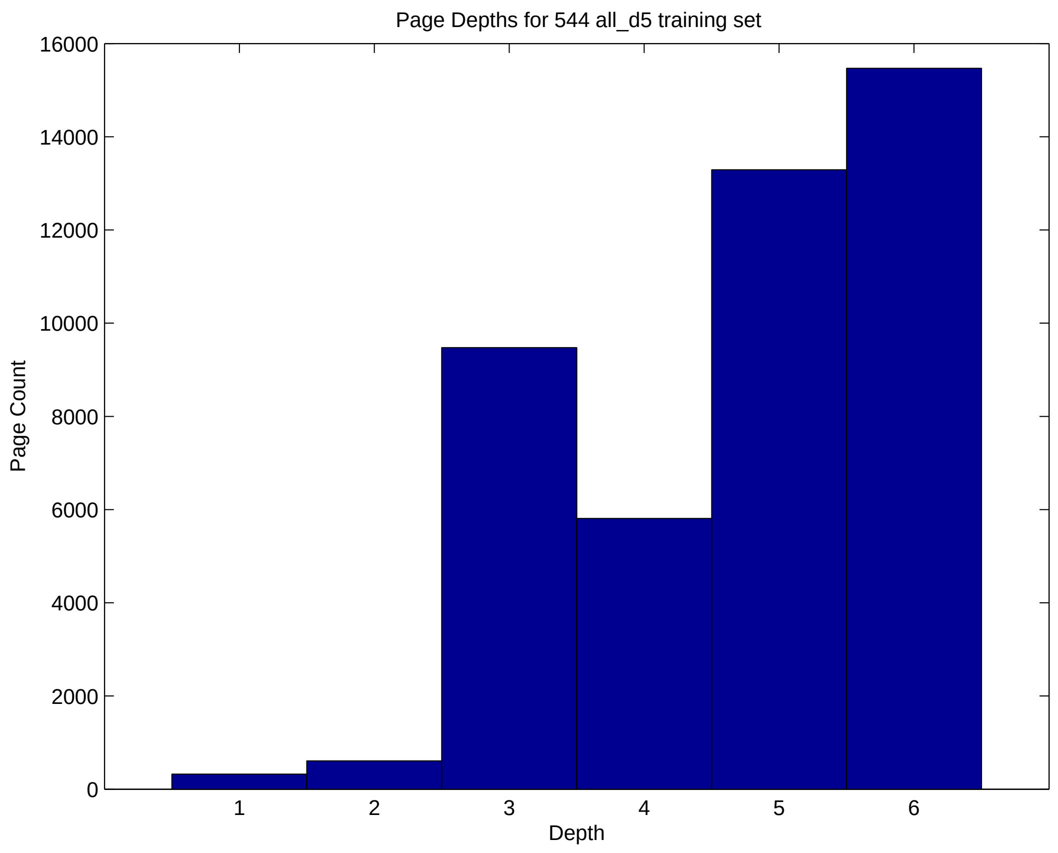


Figure 5.4: Distribution of pages at each level in the parent search-cone for the entire data set

From this target set of thirteen pages (4% of the total), we trawled the parent search-cone to a height of 4 giving us a total of 5 training depths with depth-1 corresponding to targets and depth-5 corresponding to pages having a shortest path of 4 hops to a target. This yields a training set of 2,171 pages distributed according to Figure 5.3. This figure actually shows the distribution of pages to one level further out than we actually used for training. Notice the large hump at height-3. This hump consists of 817 pages from only two domains. The first domain with 545 pages at height-3, `http://www.specialtylabs.com/`, contains a collection of interpretations of medical tests all linked to from a common page reached from our initial targets. The other domain with 217 pages, `http://www.streetlink.com/`, is a collection of Wall Street reports. The hump at height-5 is largely composed of 880 news reports on a Yahoo server. In Figure 5.4 we see the distribution of pages in the parent search-cone for the entire 544 target set.

The trawl of the training set retrieves the term frequency vectors for the pages in the search-cone. The term frequency vectors have the standard stop words removed. The remaining terms are stemmed using the classic Porter stemmer.

5.1.1 Training the Spiders

As discussed previously, our intelligent spiders are taught to perform a regression to guide their search. A particular spider learns to perform a regression from the term-frequency vector for a page to navigational distance to the closest target or to an estimate of the discounted reward available if a spider started its search from that page.

Before we invest in training spiders using either of these heuristics, we test if the heuristic works for spidering. Remember, the heuristics aren't guaranteed to work well as they ignore the non-Markovian nature of the process. How does a spider behave using one of these heuristics? Consider a spider using perfect knowledge of the distance to the nearest target. At the start of the search the spider will greedily move towards (one of) the closest target(s). However once the first target is discovered, the heuristic no-longer gives a perfectly accurate description of the world; the spider may follow red-herring paths towards targets already discovered. Another way to think of it is that the heuristics are perfect knowledge of the marginal distance to targets or

	Machine	Median Δ Discount	Mean Δ Discount	Median Δ First	Mean Δ First
Gold Depth	Random	94	292	-312	-970
	Breadth	139	278	-463	-922
	Depth	98.4	346	-327	-1150
Gold Discount	Random	94.1	292	-312	-970
	Breadth	139	278	-463	-922
	Depth	98.5	346	-327	-1150

Table 5.2: 95 start pages from 4 away to depth 5. Depth and discounted reward oracles vs. Random, BFS, and DFS. Values shown for differences between discounted rewards are differences in the logs of the discounted rewards.

discounted reward. The heuristics, however, do not give you the conditional distance given some portion of the web-graph already explored.

In Table 5.2 we see the results from a set of 95 spidering runs comparing the performance of spiders using the heuristics vs. randomized runs. For each of the uninformed search strategies we ran 20 randomized runs. For each of the 62 starting pages we executed a spider using the depth heuristic and one using the discounted reward heuristic. Remember that these spiders have *perfect* knowledge of the heuristic. We paired the results (discounted-reward and actions-to-first-target) with each of the randomized runs giving us, for each start page and search heuristic, twenty paired samples for each naive search strategy. The "Median Δ Discount" and "Median Δ First" column shows the median difference between these pairs for the log discounted reward (0.5 discount factor) and the page retrievals to first target. We also show the means to give a better feel for the distribution. Let's read an entry off the table: Gold Depth, BFS, Median Δ First: -463. This means that in median, the spider using the Depth heuristic found its first target 463 page retrievals quicker than the breadth first search. It isn't surprising that the both heuristics perform similarly, especially for getting to the first target as they are both guaranteed to get to the first target in the minimum possible number of actions, also the reward for that page will dominate the discounted reward. This domination is particularly strong with a 0.5 discount factor.

According to the Wilcoxon signed rank test of the null hypothesis that the median of the paired differences in performance is zero, the probability of observing the median differences we observe is virtually 0. Here we show results for one set of starting pages; however, for all the starting conditions tried, we observed the same phenomenon: The perfect knowledge heuristics perform very strongly.

Using either the depth or discounted reward heuristics clearly functions as an effective search heuristic. Below we present our spidering results from *approximating* these heuristics with a learned estimate.

We trained the spiders using a linear kernel support vector machine using the SVM^{light} [30] tool set. We used SVM regression with hand-tuned parameters. Other kernels were tried with no performance improvements and dramatically increased training times.

For the 544 dataset using SVM*light* with SVM regression we selected three trained support vector machines which best classified the training data. Two of the machines were trained against the discounted reward calculations and one against the distance. Each of the machines was used in two modes, with and without augmented delta scores for a total of six machines.

```
SVM 1:  1^0^0^13_d4_10000^13_d4.svmlight.svm-z r -w 0.000001 -c 0.000001^13_d4.widx
SVM 2:  1^1^0^13_d4_10000^13_d4.svmlight.svm-z r -w 0.000001 -c 0.000001^13_d4.widx
SVM 3:  1^0^1^13_d4_10000^13_d4.dp_svmlight.svm-z r -w 0.00001 -c 0.0000001^13_d4.widx
SVM 4:  1^1^1^13_d4_10000^13_d4.dp_svmlight.svm-z r -w 0.00001 -c 0.0000001^13_d4.widx
SVM 5:  1^0^1^13_d4_10000^13_d4.dp_svmlight.svm-z r -w 0.00001 -c 0.000001^13_d4.widx
SVM 6:  1^1^1^13_d4_10000^13_d4.dp_svmlight.svm-z r -w 0.00001 -c 0.000001^13_d4.widx
```

The machine name is decrypted as follows: In `a^b^c`, a = 1 indicates a SVM based machine, b = 1 indicates to use an augmented delta score (See Section 3.1.3), and c = 1 indicates to reverse the sorting for heuristics for which good is big (discounted reward) as opposed to those for which small is good (navigational distance); The `13_d4_10000` indicates the training set used (13 source pages to a height of 4 from target, filtered to top 10,000 infogain terms). The remaining portion gives the parameterization to svmlight to train the machine. If there is a `dp_`", then the machine is trained against the dynamic programming computed discounted rewards. The `-z r` selects for regression, `-c XXXX` gives the trade-off between the margin and the training error, and finally `-w XXXX` gives the width of the tube.

Machines 5 and 6 were not run for all experiments.

5.1.2 Short-range Evaluation

Now that we've selected the machines we will use to evaluate the spidering hypothesis, we step through our results in short-range spidering experiments. Some of the results in this section may appear counter-intuitive: the SVM spiders don't always perform significantly better than the naive spiders. In the sections below we will note and discuss these discrepancies.

Our discussion focuses on the two classic phases in a search problem. We often start a search in a new environment as an exploration. Then once we *have a feel* for the environment we can shift to exploitation. A common view in spidering [42] is to consider the process of getting to initial targets as exploration, and then searching from targets as exploitation. Thus our short-range testing has two components: Exploration: reaching targets from a short distance, either 3 or 4 hops to target; and Exploitation: searching from targets for a limited depth to find additional targets. In all of these short-range experiments, all of the spider strategies are competing in exactly the same space and they *will* all find the same targets. The only issues are how quickly they find the first target and what their cumulative rewards are.

In the short-range tests we constrain the depth of the search to some fixed value. What this means is that as the spider is exploring, at no time will it follow a link which exceeds the current known distance from the start page. What this means is that the spider may discover an edge in the graph which could lead to a target, but it will not follow the edge as it would exceed the depth bound. At some later point it might discover an alternate path to that page with a shorter path-length from the starting page and can then take the action of transitioning to the target. The alternative choice (free search) is to constrain the spider to explore only the search-cone at the fixed depth, but to explore the cone freely and without regard to the current known shortest path length.

We did not choose the alternative discussed above for our experiments as it does not correspond to *real* spider experiments on the live web. Allowing free search in the search-cone requires a priori knowledge of the cone. As a check, we did run limited experiments using the free search alternative and found that it universally improved the relative performance of our smart spiders.

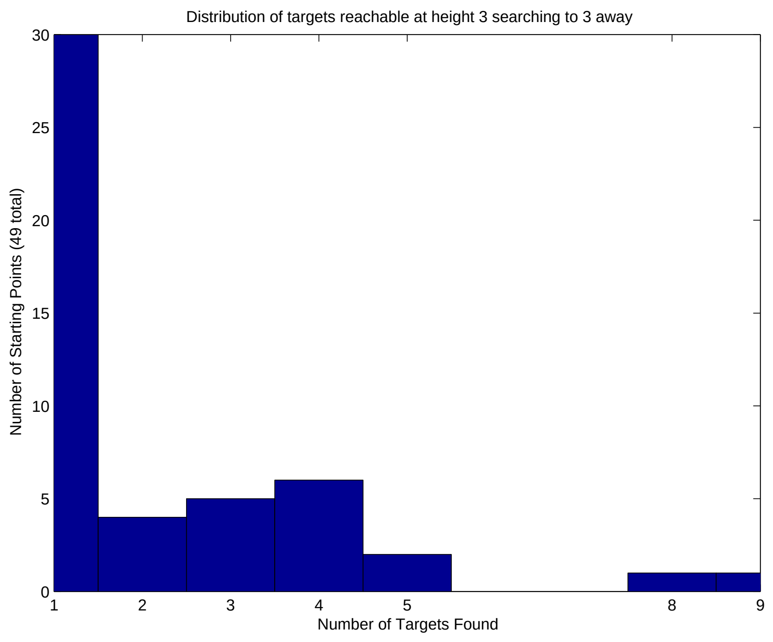


Figure 5.5: TREC Task 544: Starting at pages 3 away (49 starts) to depth 3 searching to exhaustion. Distribution of targets found. Mean number of targets: 2.1, median: 1. Mean number of pages in search-cone: 407, median: 315.

3 away

In this section we review our exploration results for very short-range searches. We are starting from 49 pages, all able to reach at least one target with a minimum length path of three steps. There are no targets reachable in fewer than three steps (page retrievals). We evaluate the spider in this environment in three phases. We will first run a pure exploration mode search in which we only search the web-graph to a depth of three. We will then increase the depth a couple more times, increasing the exploration phase as well as introducing the opportunity for some exploitation.

Consider Figure 5.5. This shows the distribution of targets reachable from our start pages with a depth-3 search cone. In more than half of the trials we will have only one target page and hence only one source of reward. Now consider Table 5.3.

This table is read similarly to Table 5.2 but is somewhat more compact. The "Δ Discount" column shows the median difference in log discounted reward between the spider all the way to the left and that of the particular row. The mean is also

	Machine	Δ0.05 Discount	0.95 Significance	Δ0.09 Discount	0.95 Significance	Δ First Target	0.95 Significance
Gold Depth	Random	31.6(54.8)	yes	5.09(8.46)	yes	-104(-182)	yes
	Breadth	50.6(63.7)	yes	7.76(9.8)	yes	-168(-211)	yes
	Depth	26.5(50.7)	yes	4.21(7.82)	yes	-87.5(-168)	yes
Gold Discount	Random	31.7(54.8)	yes	5.11(8.46)	yes	-104(-182)	yes
	Breadth	50.6(63.7)	yes	7.76(9.8)	yes	-168(-211)	yes
	Depth	26.5(50.7)	yes	4.21(7.82)	yes	-87.5(-168)	yes
SVM 1	Random	3.61(-4.92)	yes	0.569(-0.724)	yes	-12(16.4)	yes
	Breadth	7.83(4.04)	yes	1.16(0.615)	yes	-26(-13.4)	yes
	Depth	0.602(-9.03)	no	0.0915(-1.37)	no	-2(30)	no
SVM 2	Random	5.12(-3.42)	yes	0.778(-0.493)	yes	-17(11.4)	yes
	Breadth	8.73(5.55)	yes	1.33(0.846)	yes	-29(-18.4)	yes
	Depth	1.74(-7.52)	no	0.275(-1.13)	no	-5.5(25)	no
SVM 3	Random	3.14(-6.01)	no	0.503(-0.883)	no	-10.5(20)	no
	Breadth	6.92(2.96)	yes	1.05(0.456)	yes	-23(-9.81)	yes
	Depth	0.301(-10.1)	yes	0.0915(-1.52)	no	-1(33.6)	yes
SVM 4	Random	3.91(-5.42)	yes	0.641(-0.791)	yes	-13(18.1)	yes
	Breadth	7.53(3.55)	yes	1.16(0.548)	yes	-25(-11.7)	yes
	Depth	0.903(-9.52)	no	0.17(-1.43)	no	-3(31.7)	no
SVM 5	Random	3.31(0.483)	yes	0.503(0.103)	yes	-11(-1.57)	yes
	Breadth	7.42(9.45)	yes	1.16(1.44)	yes	-24.5(-31.4)	yes
	Depth	0.602(-3.62)	no	0.0942(-0.538)	no	-2(12)	no
SVM 6	Random	3.3(0.754)	yes	0.545(0.151)	yes	-11(-2.45)	yes
	Breadth	7.83(9.72)	yes	1.24(1.49)	yes	-26(-32.3)	yes
	Depth	0.301(-3.35)	no	0.0915(-0.49)	no	-1(11.2)	no

Table 5.3: TREC Task 544: 3 away to depth 3. Median (and mean) differences in log discounted reward with 0.95 significance. Note lack of significance on results against depth first search.

provided in parenthesis. Because the discounted rewards have values typically smaller than 1.0, the logs will have negative value, with the greater the value, the better the result. As a consequence, a positive result for the Discount column indicates that the machine indicated on the far left outperformed the indicated naive search. On the other hand, in the "Δ First Target" column we would expect to see a negative value in that same case as a successful search will have a low number of page retrievals before finding the first target.

Note that almost across the board, we just barely outperform in median the depth first search. Also, we don't have better than 0.95 confidence that the median's are different. Also note that if we consider the means, the depth first search is actually winning. Also, the random search comes close to winning for SVM 3. What is happening here?

Remember how the depth first search works and remember where all of the targets are in this task. The depth first search will rapidly navigate to the base of the search cone and then it will stay there, backing up only the minimum amount needed to stay

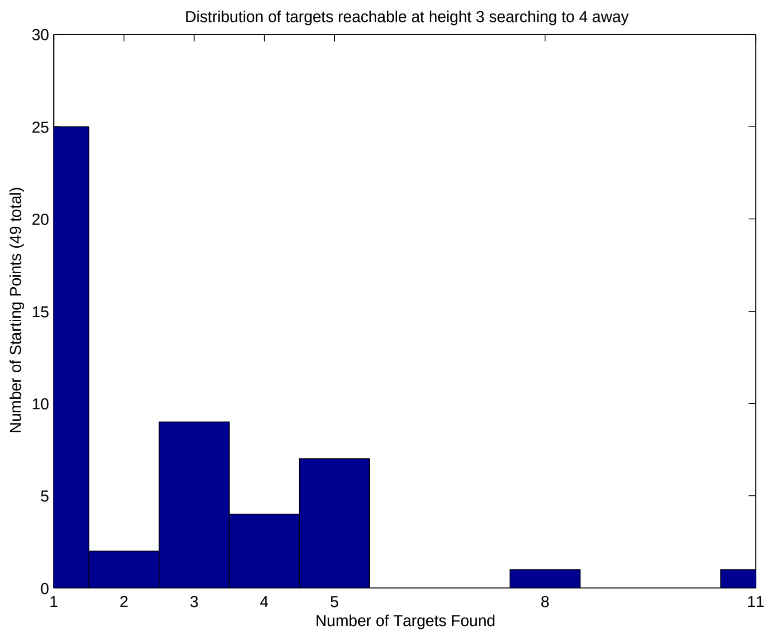


Figure 5.6: TREC Task 544: Starting at pages 3 away (49 starts) to depth 3 searching to exhaustion. Distribution of targets found. Mean number of targets: 2.6, median: 1. Mean number of pages in search-cone: 550, median: 461.

at depth 3. And, depth 3 is exactly where all the targets are so it isn't surprising that in this artificial scenario the depth first search can perform strongly. As discussed previously, the random search performs in some ways like a breadth first and in some ways like a depth first search so it is also not surprising that it performed well in this scenario.

Table 5.4 shows the same experiments, this time run to a search depth of 4. This introduces an additional opportunity for finding targets and also changes the shape of the search space. Notice in this set of results that the SVM machines uniformly outperform the naive searches in median, with 95% confidence. Note, however, that sometimes the means of the deltas indicate that the naive searches, particularly breadth-first search, are gaining advantage, with slight advantage also going to the random search. This also isn't particularly surprising.

We have added a fourth layer of pages, potentially exponentially larger than the previous layer. This new layer contains a few targets, however the ratio of non-targets to targets is high. The breadth-first search quickly spins through the depth-1 and

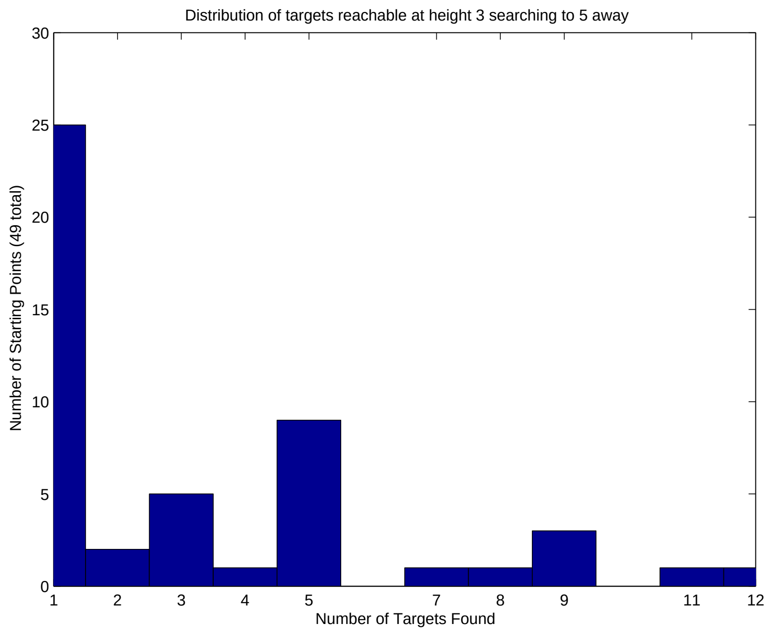


Figure 5.7: TREC Task 544: Starting at pages 3 away (49 starts) to depth 5 searching to exhaustion. Distribution of targets found.

	Machine	Δ Discount	Significance	Δ First Target	Significance
Gold Depth	Random	42(68.5)	Yes	-139(-227)	Yes
	Breadth	50.6(63.7)	Yes	-168(-211)	Yes
	Depth	37.8(72)	Yes	-125(-239)	Yes
Gold Discount	Random	42(68.5)	Yes	-139(-227)	Yes
	Breadth	50.6(63.8)	Yes	-168(-211)	Yes
	Depth	37.9(72)	Yes	-125(-239)	Yes
SVM 1	Random	7.53(-1.93)	Yes	-25(6.39)	Yes
	Breadth	6.32(-6.66)	Yes	-21(22.1)	Yes
	Depth	6.62(1.56)	Yes	-22(-5.22)	Yes
SVM 2	Random	7.38(-3.29)	Yes	-24.5(10.9)	Yes
	Breadth	6.32(-8.02)	Yes	-21(26.6)	Yes
	Depth	7.14(0.204)	Yes	-24(-0.708)	Yes
SVM 3	Random	6.92(-3.65)	Yes	-23(12.1)	Yes
	Breadth	4.82(-8.38)	Yes	-16(27.8)	Yes
	Depth	7.22(-0.154)	Yes	-24(0.496)	Yes
SVM 4	Random	8.13(-2.79)	Yes	-27(9.27)	Yes
	Breadth	6.02(-7.52)	Yes	-20(25)	Yes
	Depth	7.83(0.702)	Yes	-26(-2.34)	Yes
SVM 5	Random	7.98(1.04)	Yes	-26.5(-3.49)	Yes
	Breadth	6.32(-3.7)	Yes	-21(12.2)	Yes
	Depth	6.02(4.53)	Yes	-20(-15.1)	Yes
SVM 6	Random	8.28(1.9)	Yes	-27.5(-6.37)	Yes
	Breadth	6.92(-2.83)	Yes	-23(9.36)	Yes
	Depth	7.22(5.4)	Yes	-24(-18)	Yes

Table 5.4: TREC Task 544: 3 away to depth 4. Median (and mean) differences in log discounted reward with 0.95 significance.

	Machine	Δ Discount	Significance	Δ First Target	Significance
Gold Depth	Random	49.3(76.6)	Yes	-164(-254)	Yes
	Breadth	50.6(63.7)	Yes	-168(-211)	Yes
	Depth	48.3(81)	Yes	-160(-269)	Yes
Gold Discount	Random	49.3(76.6)	Yes	-164(-254)	Yes
	Breadth	50.6(63.8)	Yes	-168(-211)	Yes
	Depth	48.4(81)	Yes	-160(-269)	Yes
SVM 1	Random	9.09(6.61)	Yes	-30.5(-21.9)	Yes
	Breadth	5.72(-6.26)	No	-19(20.8)	No
	Depth	7.68(11)	Yes	-25.5(-36.4)	Yes
SVM 2	Random	9.03(5.35)	Yes	-30(-17.7)	Yes
	Breadth	4.52(-7.52)	No	-15(25)	No
	Depth	7.67(9.7)	Yes	-25.5(-32.2)	Yes
SVM 3	Random	8.35(4.94)	Yes	-27.5(-16.4)	Yes
	Breadth	3.61(-7.93)	No	-12(26.4)	No
	Depth	7.83(9.3)	Yes	-26(-30.9)	Yes
SVM 4	Random	9.27(4.82)	Yes	-31(-16)	Yes
	Breadth	4.21(-8.05)	No	-14(26.8)	No
	Depth	8.44(9.18)	Yes	-28(-30.4)	Yes
SVM 5	Random	8.88(8.47)	Yes	-29.5(-28.1)	Yes
	Breadth	4.91(-4.4)	Yes	-16(14.6)	Yes
	Depth	9.03(12.8)	Yes	-30(-42.6)	Yes
SVM 6	Random	8.88(8.18)	Yes	-29.5(-27.2)	Yes
	Breadth	4.82(-4.69)	Yes	-16(15.6)	Yes
	Depth	9.08(12.5)	Yes	-30(-41.7)	Yes

Table 5.5: TREC Task 544: 3 away to depth 5. Median (and mean) differences in log discounted reward with 0.95 significance.

depth-2 pages, and then find the targets which are guaranteed to be at depth-3, without wasting any time on the pages at depth-4. Unless the smart spiders have a very strong advantage they will have a hard time overcoming this problem.

If we expand out to another layer in our search cone we would expect the relative performance of the BFS spiders to continue to improve, as the ratio of targets to non-targets decreases further, with easy, for BFS, to find targets at depth-3. This is exactly what we see when we examine Table 5.5. We also can see that the SVM spiders are still beating the naive spiders in median; however this time the differences

Strategy	Depth	Δ Discount	0.95 Significance	Δ First Target	0.95 Significance
Random	3	5.12(-3.42)	yes	-17(11.4)	yes
	4	7.38(-3.29)	Yes	-24.5(10.9)	Yes
	5	9.03(5.35)	Yes	-30(-17.7)	Yes
Breadth	3	8.73(5.55)	yes	-29(-18.4)	yes
	4	6.32(-8.02)	Yes	-21(26.6)	Yes
	5	4.52(-7.52)	No	-15(25)	No
Depth	3	1.74(-7.52)	no	-5.5(25)	no
	4	7.14(0.204)	Yes	-24(-0.708)	Yes
	5	7.67(9.7)	Yes	-25.5(-32.2)	Yes

Table 5.6: Summary of Tables 5.3, 5.4, and 5.5 For SVM 2.

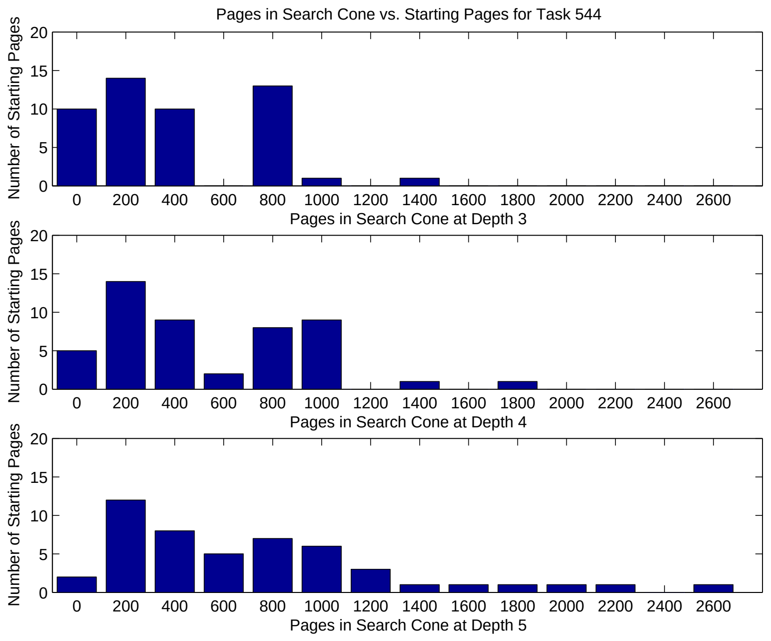


Figure 5.8: TREC Task 544: Starting from height and searching to depths 3, 4, and 5. Distribution of search-cone sizes.

in performance are frequently not significant when compared to the breadth-first search spiders.

Now if we go back through Tables 5.3, 5.4, and 5.5 and examine the time-to-first-target performance of the gold depth based spider vs. the naive breadth-first search spider we notice that the results are identical. This is completely to be expected as increasing the search depth has no impact on either algorithm, at least in the first stage of the search. Table 5.6 shows a summary of these three tables for SVM 2, the depth-trained SVM spider. As the search depth increases, we see steady improvement in the SVM spider's performance vs. both the random and depth-first strategies, with a steady drop in performance as compared to the breadth-first strategy.

Figure 5.9 shows the distribution of time-to-first-target normalized by the size of the search-cone for each of the naive search strategies along with depth-trained and discounted reward-trained SVM spiders (SVM 2 and SVM 4 respectively). Notice the distinctive pattern in each of the naive searches. The cumulative distribution function (CDF) is shown for the same spiders is shown in Figure 5.10.

Here we have examined the performance of our spiders starting very close to known

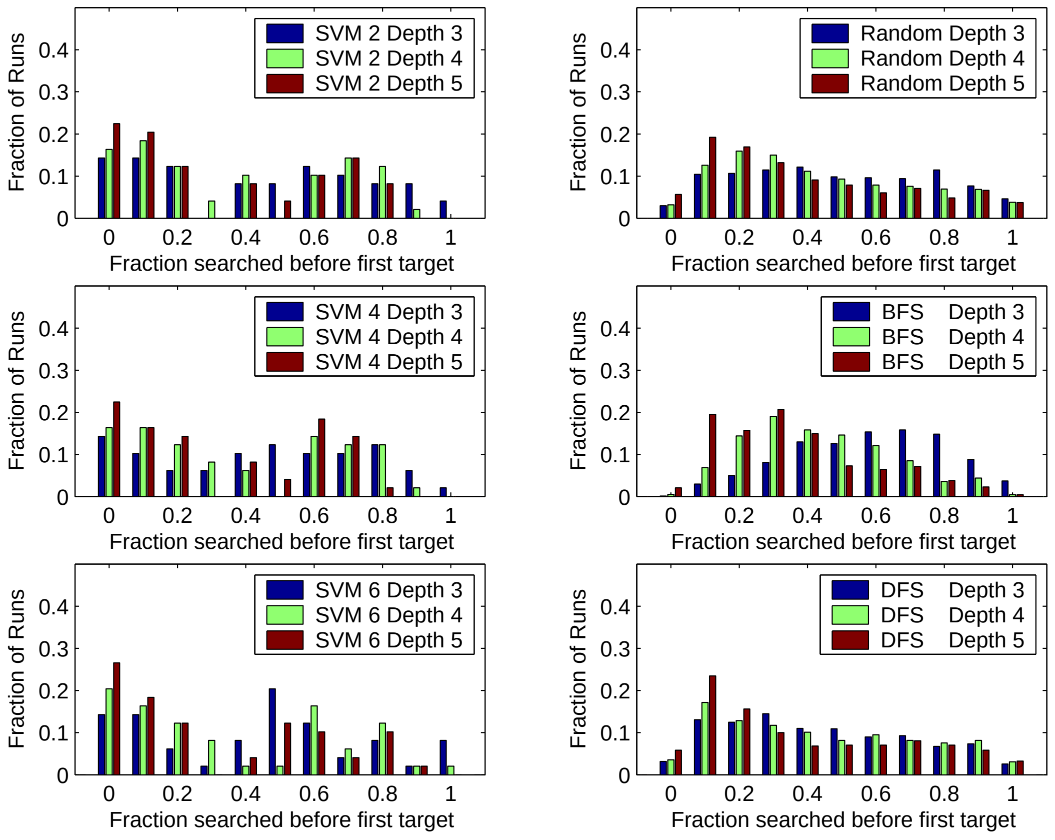


Figure 5.9: TREC Task 544: Starting at pages 3 away (49 starts) from targets and searching to depths 3, 4, and 5. Distribution of page retrievals to find first target divided by total pages in search cone.

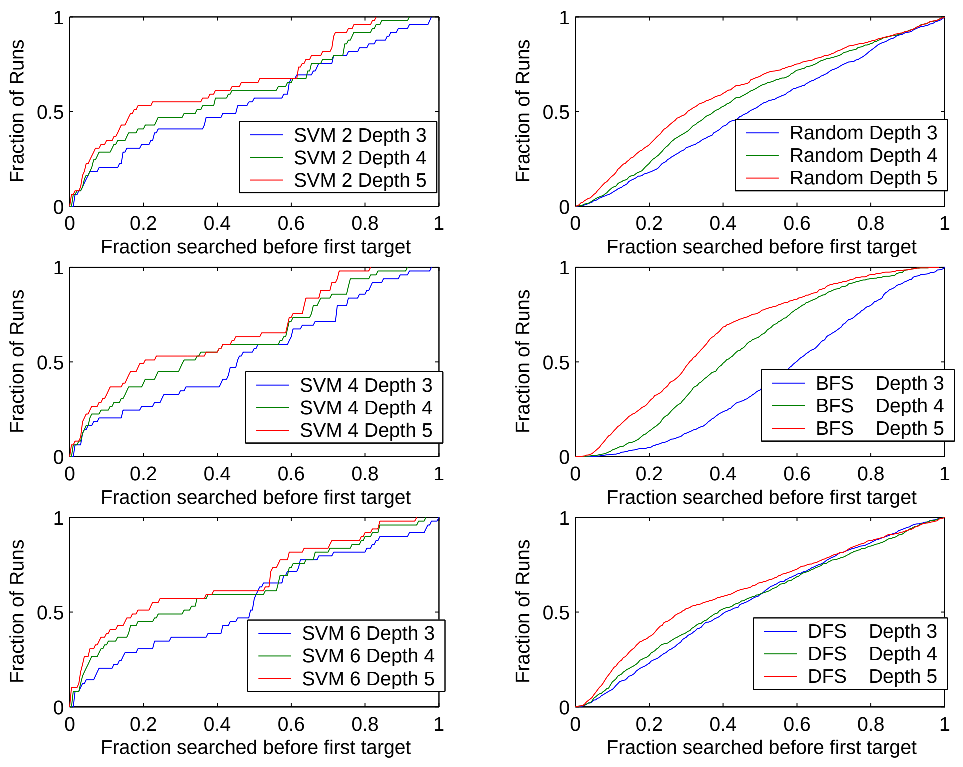


Figure 5.10: TREC Task 544: Starting at pages 4 away (95 starts) from targets and searching to depths 4 and 5. Cumulative distribution of page retrievals to find first target divided by total pages in search cone.

targets. We observe what at first blush appears to be a surprising result: The naive spiders perform particularly well. When the depth of search is tightly constrained, the DFS spider performs well, focusing its efforts on the target-bearing layer. As the depth of search is increased, we notice a shift in performance towards the BFS spider as the BFS spider ignores the target-sparse depths until it has first exhausted the target bearing layer. Next we will examine our spidering performance from a height of four and consider how this changes our relative performances.

4 away

We saw previously that if we have a target at three hops away from our starting pages then the naive searches give our SVM spiders a run for their money. We also saw that as we the allowed search depth increases, the performance advantage shifts from depth-first search towards breadth-first search. We also saw that the relative performance of our SVM spiders improved as the depth increased, aside from the issue with breadth-first search which we explained above. This increase in performance as the depth increases suggests that our SVM spiders may do better if we increase our starting distance from the target. We also can predict that increasing the starting distance will immediately reduce the breadth-first search advantage as the size of the empty search-cone is greatly increased.

In the tests discussed here, we randomly selected 95 starting pages each having the closest target exactly 4 hops away. We then ran two batches of experiments. First the spiders were constrained to search only in those pages reachable in 4 hops. Next we opened up the search slightly and constrained the search to those pages reachable in 5 hops. Unlike in the previous tests, we did not run the experiments at depth-3 as there are guaranteed to be known targets within the search-cone. We didn't run to depth-6 due to run-time constraints. The search-cones are just too big. The spiders were run until they had completely exhausted the allowed pages. Also in these experiments, we didn't run the second discounted reward spider. The SVM trained spiders we selected had very similar performance characteristics so in the name of time savings we did not run the additional tests.

Our first set of results appear in Table 5.7. Just as we would expect and as we saw with the height-3 depth-3 experiment, the depth-first search does best of all the naive

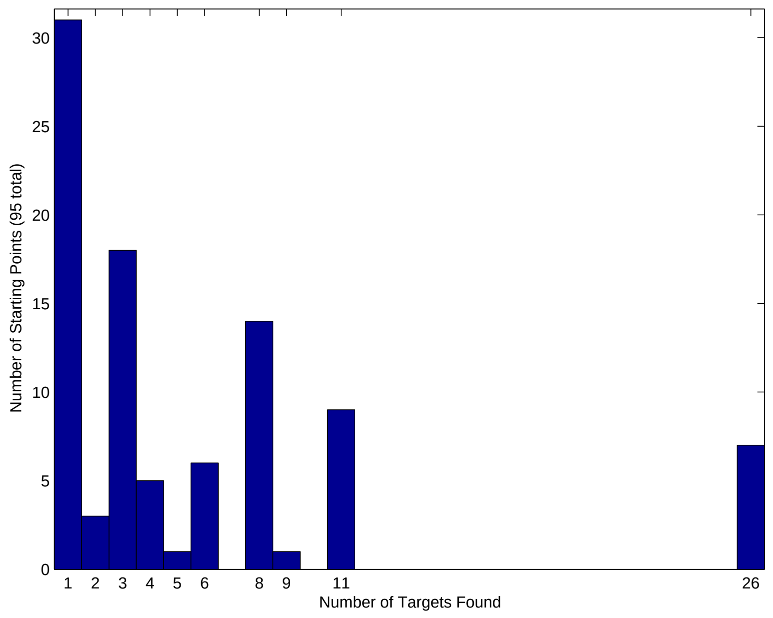


Figure 5.11: TREC Task 544: Starting from pages 4 away (95 starts) to depth 4 searching to exhaustion. Distribution of targets found. Mean number of targets: 5.8, median 3. Mean number of pages in search-cone: 1726, median 1086.

	Machine	Δ Discount	0.95 Significance	Δ First Target	0.95 Significance
Gold Depth	Random	82.3(207)	Yes	-273(-688)	Yes
	Breadth	139(278)	Yes	-463(-922)	Yes
	Depth	76.4(183)	Yes	-253(-608)	Yes
Gold Discount	Random	81.3(207)	Yes	-270(-686)	Yes
	Breadth	136(277)	Yes	-451(-920)	Yes
	Depth	75.1(183)	Yes	-249(-606)	Yes
SVM 1	Random	12.9(85.3)	Yes	-43(-283)	Yes
	Breadth	31.2(156)	Yes	-104(-518)	Yes
	Depth	10.8(61.3)	Yes	-36(-204)	Yes
SVM 2	Random	12.3(83.6)	Yes	-41(-278)	Yes
	Breadth	30.5(154)	Yes	-101(-512)	Yes
	Depth	9.52(59.6)	Yes	-31.5(-198)	Yes
SVM 3	Random	11.9(82.5)	Yes	-39.5(-274)	Yes
	Breadth	28.6(153)	Yes	-95(-508)	Yes
	Depth	9.03(58.5)	Yes	-30(-195)	Yes
SVM 4	Random	12.9(84.1)	Yes	-42.5(-279)	Yes
	Breadth	30.7(155)	Yes	-102(-514)	Yes
	Depth	9.4(60.1)	Yes	-31(-200)	Yes

Table 5.7: TREC Task 544: 4 away to depth 4. Median (and mean) differences in log discounted reward with 0.95 significance.

	Machine	Δ Discount	0.95 Significance	Δ First Target	0.95 Significance
Gold Depth	Random	94(292)	Yes	-312(-970)	Yes
	Breadth	139(278)	Yes	-463(-922)	Yes
	Depth	98.4(346)	Yes	-327(-1150)	Yes
Gold Discount	Random	94.1(292)	Yes	-312(-970)	Yes
	Breadth	139(278)	Yes	-463(-922)	Yes
	Depth	98.5(346)	Yes	-327(-1150)	Yes
SVM 1	Random	16.6(149)	Yes	-55(-495)	Yes
	Breadth	41(135)	Yes	-136(-447)	Yes
	Depth	16.2(203)	Yes	-54(-674)	Yes
SVM 2	Random	16.6(148)	Yes	-55(-493)	Yes
	Breadth	40.4(134)	Yes	-134(-445)	Yes
	Depth	19.2(202)	Yes	-64(-672)	Yes
SVM 3	Random	16.9(143)	Yes	-56(-474)	Yes
	Breadth	37.6(128)	Yes	-125(-426)	Yes
	Depth	18.1(197)	Yes	-60.5(-653)	Yes
SVM 4	Random	17.8(148)	Yes	-59(-492)	Yes
	Breadth	40.2(134)	Yes	-133(-444)	Yes
	Depth	21.1(202)	Yes	-70(-670)	Yes

Table 5.8: TREC Task 544: 4 away to depth 5. Median (and mean) differences in log discounted reward with 0.95 significance.

searches. Close in performance is the random search followed by the breadth-first search. However, unlike at height-3, the SVM trained spiders solidly outperform the naive searches. Note that the means favor our trained spiders unlike the trend in the height-3 experiments.

Now let's examine our results as we increase our search depth. Figure 5.12 shows the distribution of targets for the 95 runs in these tasks searching to depth-5. If things follow the pattern seen at height-3, we would expect the breadth first search to increase in performance relative to the other naive searches. If we look at the performance of the naive searches against the gold standard searches we see that the breadth-first search holds its ground (as it must since it searches exactly as before until it finishes depth-4) while the random and depth-first lose ground. Unlike at height-3, however, the SVM spiders do not lose ground to the naive searches: there is more "room" for the SVM spiders to leverage their advantage.

We might also like to examine the relative performance of each of the spider strategies against each of the others, along with the significance of the difference in medians. Table 5.9 gives the relative performance of our different spidering strategies. Above the diagonal axis we have the Wilcoxon Paired Signed-Ranked Test measure of significance for H_0 equivalent medians. Below the diagonal we have we have the median difference in log_{10} of the 0.5 discounted reward with the mean of the same

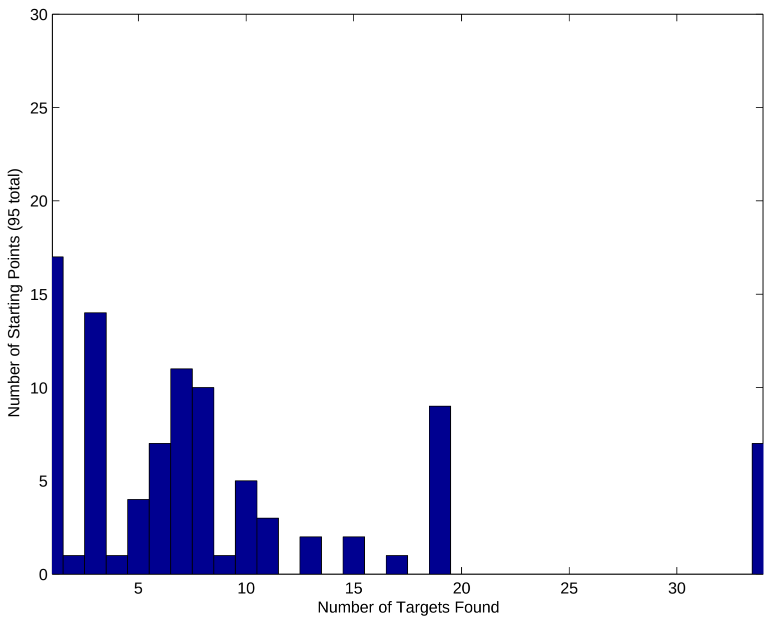


Figure 5.12: TREC Task 544: Starting from pages 4 away (95 starts) to depth 5 searching to exhaustion. Distribution of targets found.

	Gold Depth	Gold Discount	Depth SVM 2	Discount SVM 4	Random	BFS	DFS
Gold Depth	0	5.03e-06	2.6e-17	2.59e-17	0	0	0
Gold Discount	0(-0.0116)	0	2.6e-17	2.59e-17	0	0	0
Depth SVM 2	65.7(144)	65.7(144)	0	0.869	4.07e-23	8.5e-90	1.53e-30
Discount SVM 4	62(144)	62(144)	0(0.387)	0	4.51e-25	5.36e-96	1.02e-31
Random	94(292)	94.1(292)	16.6(148)	17.8(148)	0		
Breadth First	139(278)	139(278)	40.4(134)	40.2(134)		0	
Depth First	98.4(346)	98.5(346)	19.2(202)	21.1(202)			0

Table 5.9: TREC Task 544: 95 starting pages at 4 away to depth 5. Median (and mean) differences in log discounted reward with Wilcoxon statistical significance

in parenthesis. We read the table as follows: The median difference in performance between the "Gold Depth" spider and the SVM trained spider estimating depth is 65.7 with a mean difference of 144. The probability of actually observing this difference given the medians were the same and observing the experimental data we observed is less than or equal to 2.6×10^{-17}, i.e., we should reject the null hypothesis of equivalent medians. We can see from this table that the SVM trained spiders outperformed, with significance, all of the brute force search strategies. We can also see that the depth trained and discounted reward trained SVM spiders performed equivalently with significance.

	Gold Depth	Gold Discount	Depth SVM 2	Discount SVM 4	Random	BFS	DFS
Gold Depth	0	0	2.58e-17	2.58e-17	0	0	0
Gold Discount	0(-0.0105)	0	2.58e-17	2.58e-17	0	0	0
Depth SVM 2	-218(-477)	-218(-477)	0	0.835	4.57e-23	1.01e-89	1.01e-89
Discount SVM 4	-206(-478)	-206(-478)	0(-1.2)	0	4.86e-25	5.48e-96	5.48e-96
Random	-312(-970)	-312(-970)	-55(-493)	-59(-492)	0		
Breadth First	-463(-922)	-463(-922)	-134(-445)	-133(-444)		0	
Depth First	-463(-922)	-463(-922)	-134(-445)	-133(-444)			0

Table 5.10: TREC Task 544: 95 starting pages at 4 away to depth 5. Page retrievals to first target

In Table 5.10 we see the same table for the same experiments with the number of actions to first target shown instead. For example, we can see that the discounted reward trained SVM spider finds in median the first target page in 59 (492 for the mean) steps faster than a random spider and that there is strong reason to reject the null hypothesis. Negative value for the difference indicates that the column machine performed better than the row machine.

Figure 5.13 summarizes the performance (time-to-first-target) of a couple of the SVM based spiders as well as that of the naive spiders. Unlike in the tables, the results in this plot are normalized by the size of the search cone so that all results lie in the range of 0 to 1. Where a result close 0 would indicate a search which found the first target very quickly relative to the search-cone and a result close to 1 would indicate that almost the entire space was searched before the first target was found. Just as we saw in Figure 5.9 for the height-3 searches, the naive search distributions have very unique and regular shapes. The distributions for the SVM searches are not as smooth as there are one twentieth the samples and are also very similar to each other. They appear to be bi-modal, with a portion having very rapid target acquisition, and another group having late time-to-first-target. Figure 5.14 is the cumulative density function for the same data. Notice the high slope at the very left hand edge in the CDFs for the SVM spiders: In about 40% of the runs, the first target is found earlier than 10% of the way into the run.

In the above experiments we showed that the SVM-trained spiders out-perform or at worse perform equivalently in median to the naive spiders in the challenging exploitation phase with target locations favoring the naive searches. We can also see that we have room for massive improvement in the performance of the SVM-trained spiders as they lag far behind the gold standard heuristic spiders they are trained to

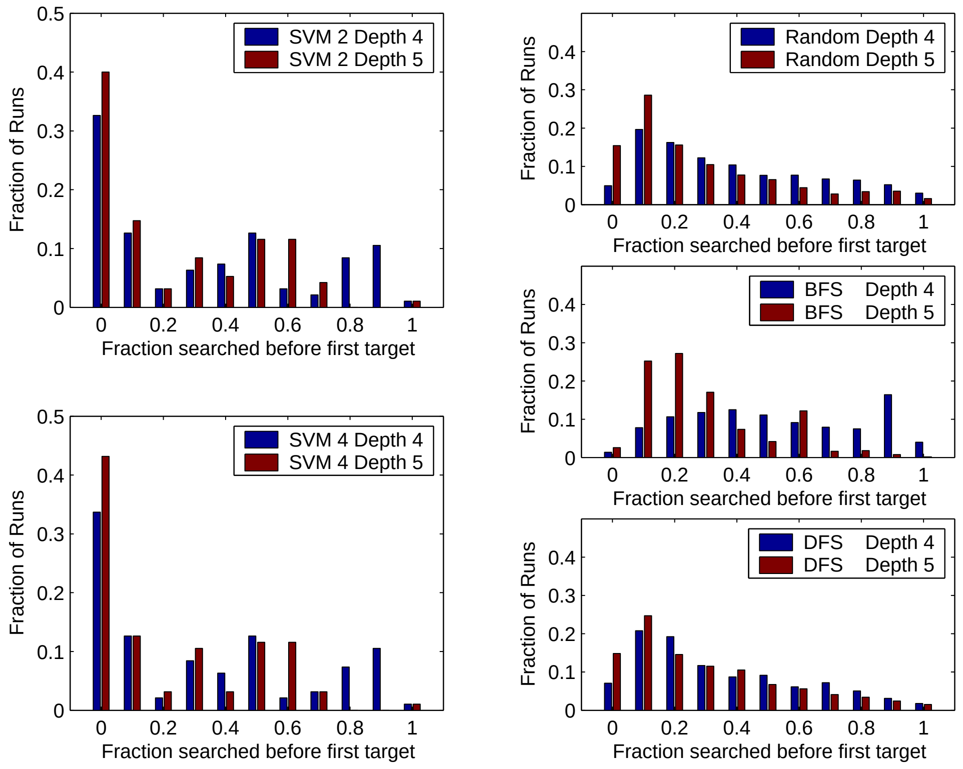


Figure 5.13: TREC Task 544: Starting at pages 4 away (95 starts) from targets and searching to depths 4 and 5. Distribution of page retrievals to find first target divided by total pages in search cone.

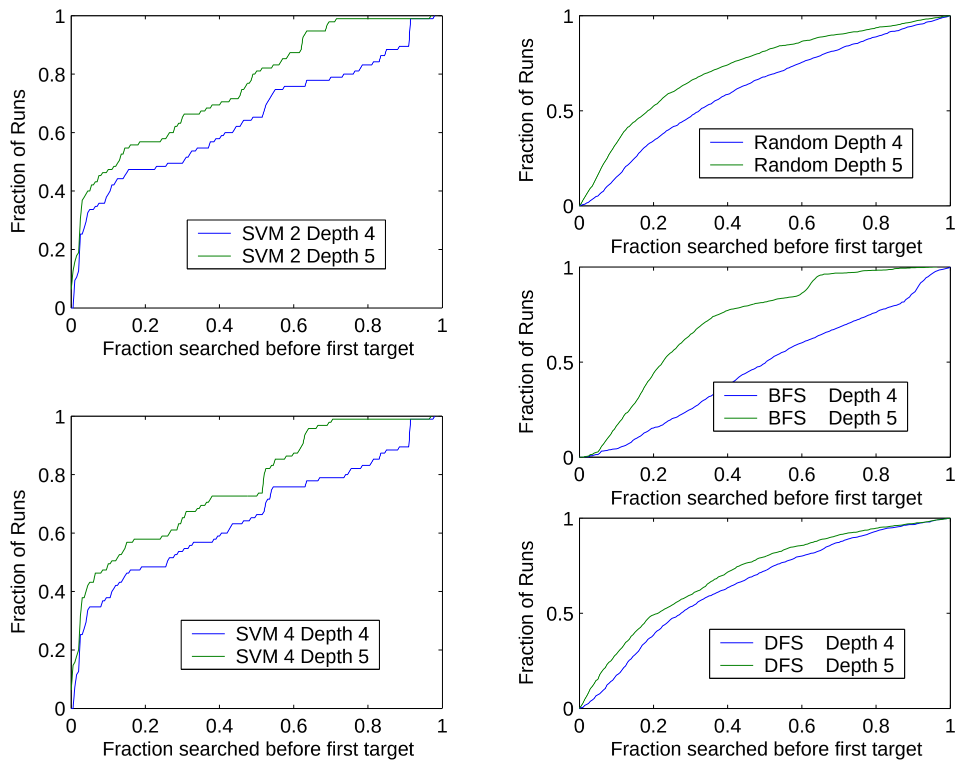


Figure 5.14: TREC Task 544: Starting at pages 4 away (95 starts) from targets and searching to depths 4 and 5. Cumulative distribution of page retrievals to find first target divided by total pages in search cone.

emulate.

Starting from Targets

We showed above in the 544 dataset that our SVM trained spiders can successfully explore the environment finding target pages faster, or at worse competitively, with the naive search strategies. One aspect of the performance obtained above is that exploitation is inextricably intermixed with the exploration: Once the first target is found, we automatically are mixing exploitation with additional exploration. In this subsection we try to directly consider the exploitation problem by starting from target pages.

We selected as our initial test set all of the targets not used in our training set. We then discarded all those targets from which no other targets are reachable within a search depth of 4 as well as a few that had only one target reachable with that one target as the entire search space, i.e., a few of the search-cones had only two pages, the initial target, and an additional target. These pages were rejected as their performance is completely predetermined, either there will be no reward, or there will be a large and instantaneous award. After removing these cases we were left with a total of 186 starting pages for our test set.

What kind of space are we searching in? If we trawl out from all the targets to a fixed depth we find a loosely exponentially growing search space. We can see the distribution of pages at each search depth in Figure 5.15. If we trawl out far enough, we won't actually get exponential growth because the fan-out will be damped by the limited size of the WT10g corpus, i.e., as the search depth increases, a given link has a greater chance of either linking back into the explored graph or of linking out of the corpus entirely. While the diagram shows the distribution to a considerable depth, we won't actually search out that far in our short-range experiments.

If we search out to a small depth, how many targets can we find. Figures 5.16 and 5.17 show the distribution of targets reachable in searches with depth-4 and depth-5 respectively starting from the 186 initial targets. Figure 5.18 shows the distribution of sizes of search-cones within which the reachable targets are found.

Unlike with our tests at height 3 and 4 where there were no targets near our starting points, in these experiments we would expect there to be targets nearby.

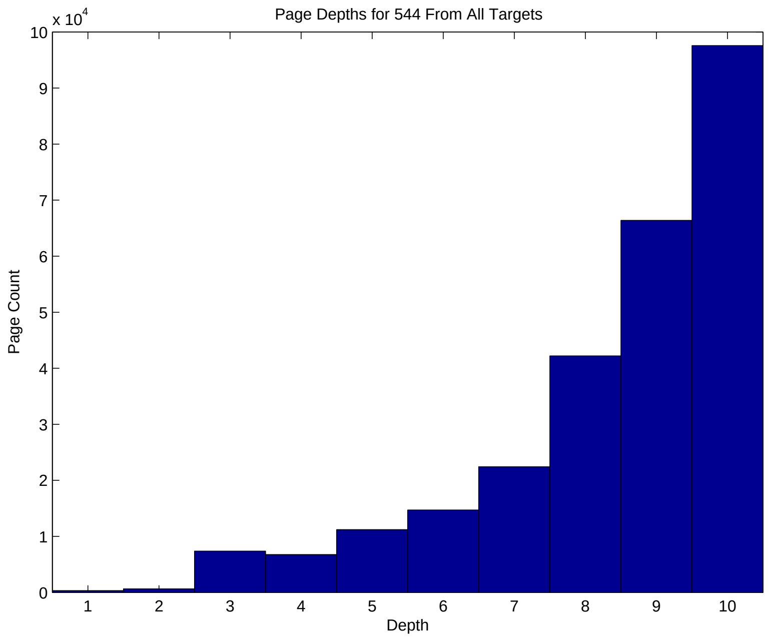


Figure 5.15: TREC Task 544: Starting from all of the targets in the 544 task and trawling the search-cone. Count of pages discovered at each depth. In the diagram, the target pages are labeled as depth-1.

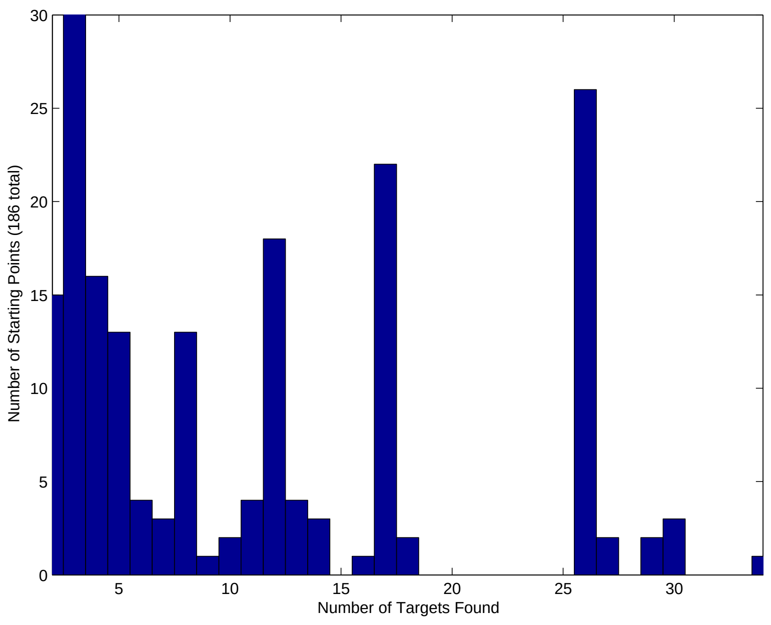


Figure 5.16: TREC Task 544: Starting from targets (186 starts) and searching to depth 4 to exhaustion. Distribution of targets found.

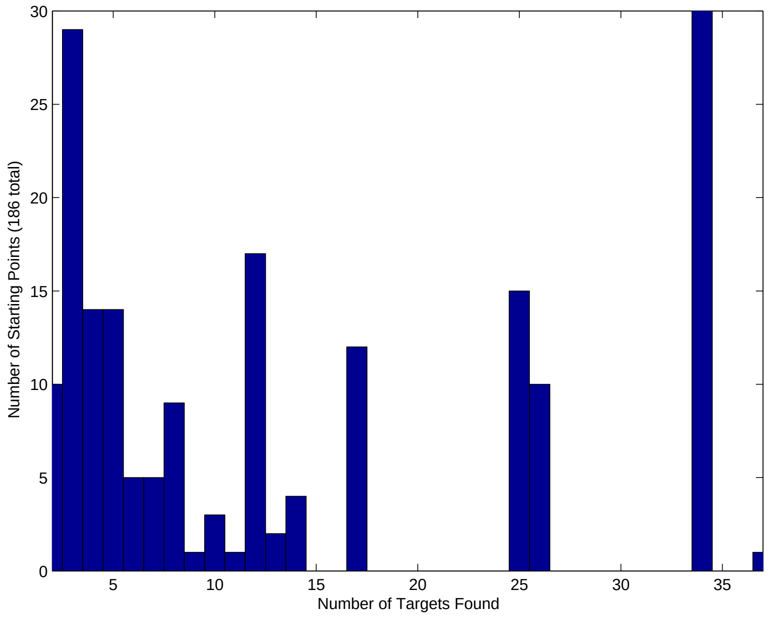


Figure 5.17: TREC Task 544: Starting from targets (186 starts) and searching to depth 5 to exhaustion. Distribution of targets found.

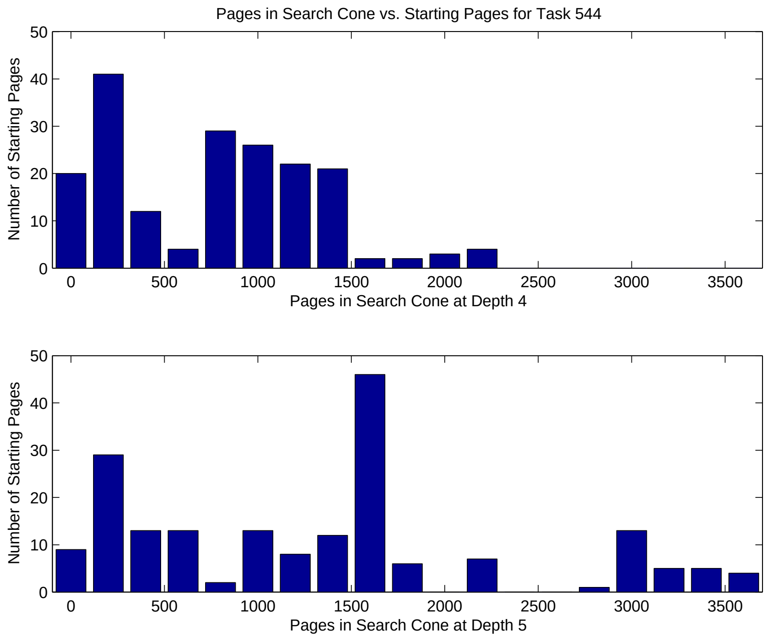


Figure 5.18: TREC Task 544: Starting from targets (186 starts) and searching to depths 4 and 5. Distribution of search-cone sizes.

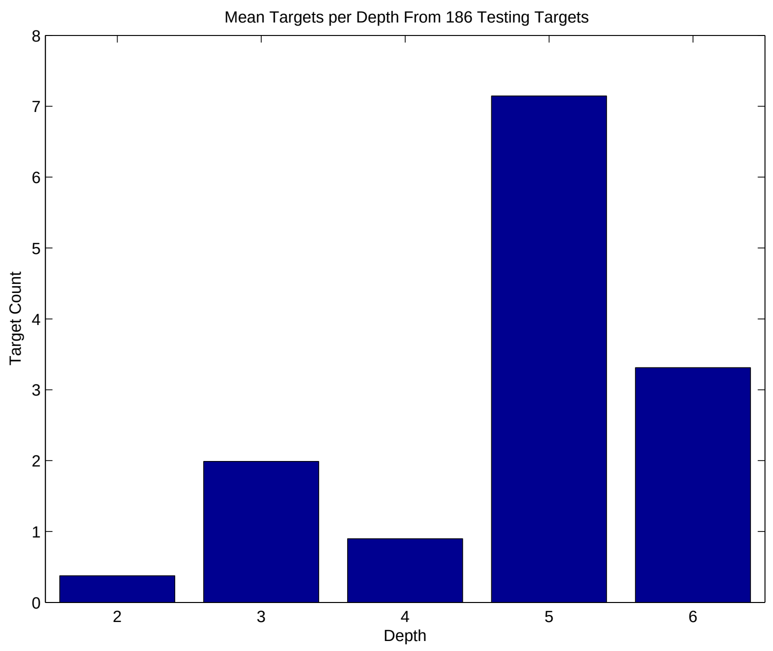


Figure 5.19: TREC Task 544: Starting from targets (186 starts) and searching to depth 5. Average number of targets per search-cone.

According to the thematically unified cluster (TUC) theory, we would expect a greater density of targets close to our targets than further away. In Figure 5.19 we see the average number of targets in the search-cones used in our tests. We consider the initial targets to be at depth-1 so our plot begins at depth-2 and goes to depth-6 for a search depth of 5. Notice that we are finding targets immediately adjacent to the starting pages. At first blush, this chart would indicate that the targets increase as strongly as our depth increases, violating the TUC theory. However, we need to keep in mind that the number of non-target pages increases exponentially with each depth. In Figure 5.20 we show, instead, the ratio of targets to total pages at each depth. Here we can see that the target density falls off rapidly as we get further away from our initial targets.

We can expect the target density to impact our performance in several ways. First the power of our gold-standard heuristic may be slightly reduced. Why is that? The heuristic spiders use exact knowledge of the depth of the nearest spider for selecting the page to explore next. However this knowledge is based on the entire web-graph,

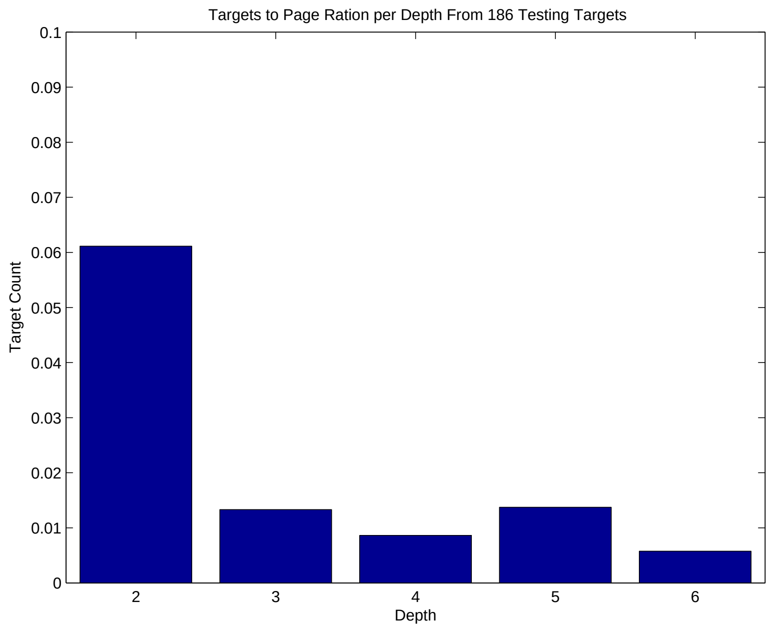


Figure 5.20: TREC Task 544: Starting from targets (186 starts) and searching to depth-5. Ratio of reachable targets to pages in each depth. Depth-5 is labeled ”6” in the graph.

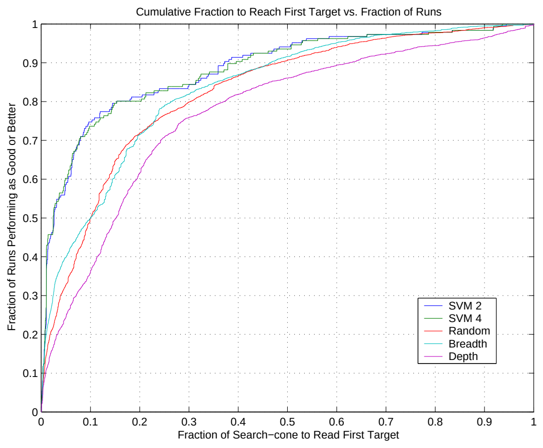
	Machine	Δ Discount	Significance	Δ First Target	Significance
Gold Depth	Random	15.6(27.5)	Yes	-51(-90.8)	Yes
	Breadth	10.2(24.7)	Yes	-34(-81.6)	Yes
	Depth	23.6(37.9)	Yes	-78(-125)	Yes
Gold Discount	Random	15.6(27.6)	Yes	-51(-91.1)	Yes
	Breadth	10.2(24.8)	Yes	-34(-81.9)	Yes
	Depth	23.7(38)	Yes	-78(-126)	Yes
SVM 1	Random	6.02(5.7)	Yes	-20(-18.9)	Yes
	Breadth	2.11(2.92)	Yes	-7(-9.7)	Yes
	Depth	11.7(16.1)	Yes	-39(-53.4)	Yes
SVM 2	Random	6.02(5.66)	Yes	-20(-18.8)	Yes
	Breadth	2.1(2.88)	Yes	-7(-9.57)	Yes
	Depth	11.9(16)	Yes	-39.5(-53.3)	Yes
SVM 3	Random	6.01(6.47)	Yes	-20(-21.5)	Yes
	Breadth	2.41(3.69)	Yes	-8(-12.3)	Yes
	Depth	11.4(16.9)	Yes	-38(-56)	Yes
SVM 4	Random	6.02(6.8)	Yes	-20(-22.6)	Yes
	Breadth	2.47(4.02)	Yes	-8(-13.4)	Yes
	Depth	11.7(17.2)	Yes	-39(-57.1)	Yes

Table 5.11: TREC Task 544: Target pages to depth 4. Median (and mean) differences in log discounted reward with 0.95 significance.

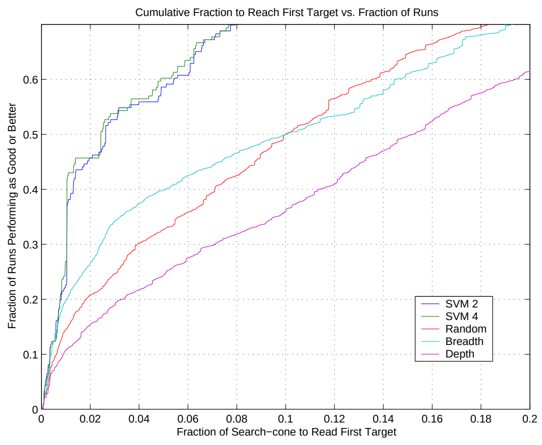
not on only the unexplored region. So the heuristic could say that a given page is 1 hop away from a target, but that target might already be explored! In fact that target might even be the page we started from (also expected with TUC theory). In the results we see a slight, but insignificant difference in the performance in the two gold-standard spiders, with the total-reward being more robust to this issue. Another factor we may encounter is that because there are targets within a few hops, the breadth first search may perform very well.

For this test we only apply two of the SVMs, one trained against the depth heuristic and the other against discounted reward. We ran the spiders to depth-4 and to depth-5. We will continue to test the SVM spiders without (SVM 1 and SVM 3) and with amplification of the difference in regression between the base page and the base page with extra weight for the link's anchor text (SVM 2 and SVM 4). SVM 1 and SVM 2 use the same depth-trained SVM machine used in the experiments at heights three and four above with the same name. Likewise, SVM 3 and SVM 4 both use the same discounted reward-trained machine as above. We ran the tests at search depths of 4 and 5 just as we did in our exploration test at height-4.

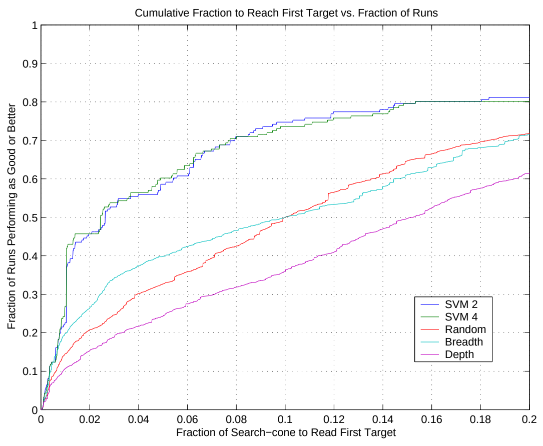
In Table 5.11 we see that indeed the breadth first search does quite well in these experiments and that the depth first search does quite badly. We also see that our SVM trained spiders do even better. In fact the gap in performance between the



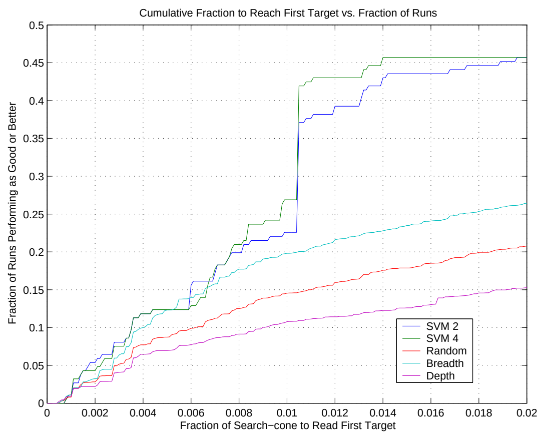
(a) Full Distribution



(b) Zoomed in Closer



(c) Zoomed in Closer



(d) Zoomed in Closer Still

Figure 5.21: TREC Task 544: Starting from targets (186 starts) and searching to depth 4. Cumulative performance normalized by cone size viewed at different scales.

	Machine	Δ Discount	Significance	Δ First Target	Significance
Gold Depth	Random	16.3(33)	Yes	-54(-109)	Yes
	Breadth	10.2(24.7)	Yes	-34(-81.6)	Yes
	Depth	28.5(54.5)	Yes	-94(-180)	Yes
Gold Discount	Random	16.3(33.1)	Yes	-54(-109)	Yes
	Breadth	10.2(24.8)	Yes	-34(-81.9)	Yes
	Depth	28.5(54.6)	Yes	-94(-181)	Yes
SVM 1	Random	6.72(15.2)	Yes	-22(-50.4)	Yes
	Breadth	2.08(6.93)	Yes	-7(-23)	Yes
	Depth	15.9(36.7)	Yes	-53(-122)	Yes
SVM 2	Random	6.76(14.9)	Yes	-22(-49.5)	Yes
	Breadth	1.93(6.65)	Yes	-6(-22.1)	Yes
	Depth	15.7(36.4)	Yes	-52(-121)	Yes
SVM 3	Random	7.22(14.6)	Yes	-24(-48.4)	Yes
	Breadth	2.11(6.31)	Yes	-7(-21)	Yes
	Depth	15.7(36.1)	Yes	-52(-120)	Yes
SVM 4	Random	7.22(15)	Yes	-24(-49.7)	Yes
	Breadth	2.11(6.71)	Yes	-7(-22.3)	Yes
	Depth	16(36.5)	Yes	-53(-121)	Yes

Table 5.12: TREC Task 544: Target pages to depth 5. Median (and mean) differences in log discounted reward with 0.95 significance.

gold-standard and the heuristic spiders is an order of magnitude smaller than in the exploration tests presented previously.

Examining Figure 5.21 we see the relative cumulative performance of a couple of the SVM spiders combined with the naive spiders. Each subsequent sub-figure zooms in a little bit closer to the origin to reveal the structure of the plot. The x-axis gives the fraction of the search-cone spidered before the first target was discovered and the y-axis gives the fraction of runs. Reading a point off these plots we see that, for both the random and breadth-first spiders, in 50 percent of the runs, the first target was discovered within the first 10 percent of the run. Likewise for the SVM spiders, in 50 percent of the runs the first target was discovered within about the first 2.5% of the size of the search cone. As we predicted the breadth-first search outshines the depth-first search, with the random search resting in the middle.

In these results we see that our spiders do well competing against the naive searches, even against the breadth-first search which truly thrives when targets are nearby. What do we expect will happen when we open up our search depth one more notch? Just as in the exploration tests, the absolute performance of the breath-first search will not change. The random and depth-first searches strategies should degrade in performance.

Table 5.12 shows the comparison of our results against the naive spiders at depth 5.

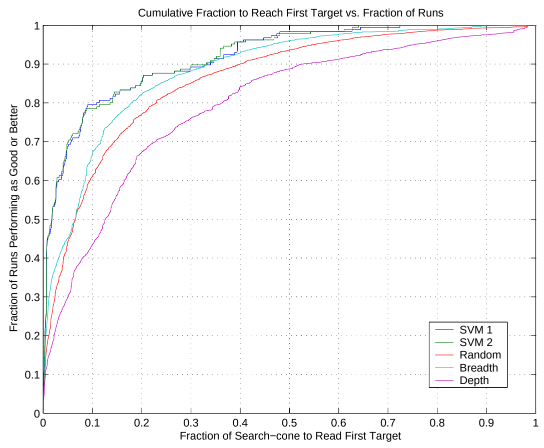
	Gold Depth	Gold Discount	Depth SVM 2	Discount SVM 4	Random	Breadth First	Depth First
Gold Depth	0	1.29e-13	8.78e-32	8.78e-32	0	0	0
Gold Discount	-0.00345(-0.098)	0	8.64e-32	8.64e-32	0	0	0
Depth SVM 2	2.87(18.1)	3(18.2)	0	0.0825	1.09e-201	2.55e-110	1.57e-289
Discount SVM 4	2.68(18)	2.68(18.1)	0(-0.059)	0	6.87e-201	3e-108	2.11e-288
Random	16.3(33)	16.3(33.1)	6.76(14.9)	7.22(15)	0		
Breadth First	10.2(24.7)	10.2(24.8)	1.93(6.65)	2.11(6.71)		0	
Depth First	28.5(54.5)	28.5(54.6)	15.7(36.4)	16(36.5)			0

Table 5.13: TREC Task 544: 186 Target pages to depth 5. log_{10} of 0.5 Discounted Reward

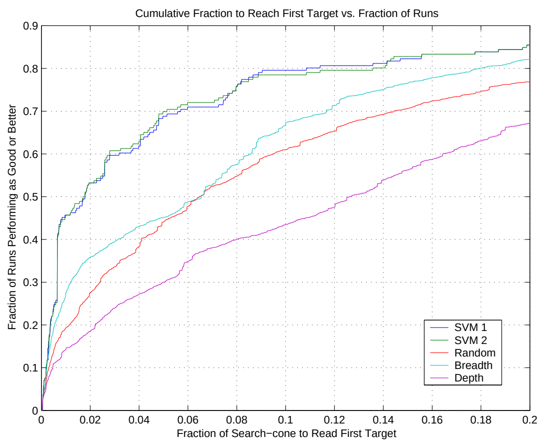
Tables 5.13 and 5.14 show the relative performance of each of the spidering strategies (below the diagonal) and the corresponding probability that the null hypothesis of equal medians is correct (above the diagonal).

Figure 5.22 shows the normalized performance of our spiders searching to depth-5 at different zooms. Looking at Sub-Figure 5.22(a), we see that for all of the search-cones, all of the spidering strategies converge to finding the first target by the end of the run. This is of course necessary as all of the search-cones are searched to exhaustion and they also all contain at least one target, otherwise they were eliminated from the experiment. The y-axis shows the fraction of spidering runs (one run per search-cone) for each strategy in which the first target is found in the corresponding fraction of the search-cone as measured on the x-axis. For example, we can see in Sub-Figure 5.22(b) that in about 53% of the SVM based runs the first target is found within the first 2% of the pages examined. Contrast this with the best performing naive searches (breadth and random) which need to search about 7% of the search-cone in order have the same 53% chance of discovering the first target. When we examine Sub-Figure 5.22(d) we see how early in the search the SVM based spiders start to pull away from the naive ones. At about 0.1% into the search the random and depth-first spiders are left behind and then at about 0.3% the breadth-first search is left behind until it briefly catches up at about 30%.

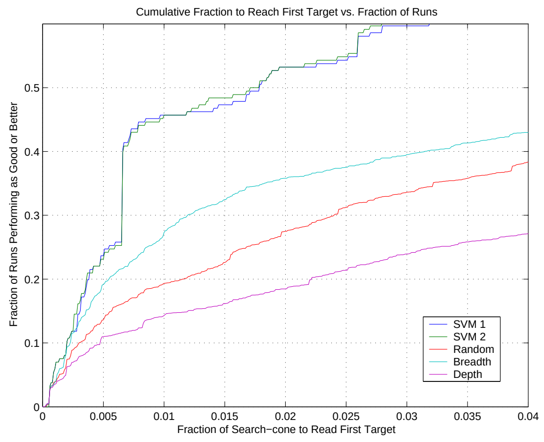
Figure 5.23 has the same results but with an unnormalized x-axis showing the raw number of page retrievals. In this we can see that our SVM-based spiders are finding the first target in half the runs within the first 12 pages retrieved while the breadth-first search is requiring three times the page retrievals to get the same performance level!



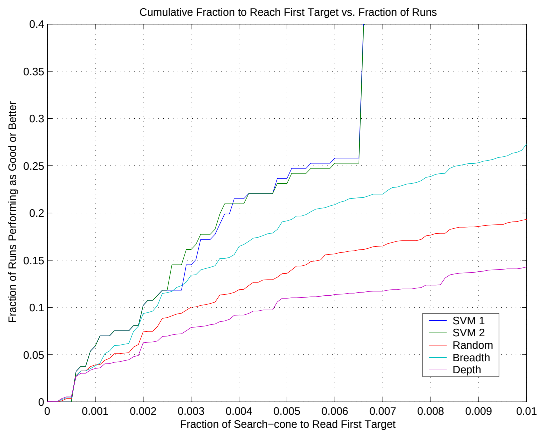
(a) Full Distribution



(b) Zoomed in Closer



(c) Zoomed in Closer



(d) Zoomed in Closer Still

Figure 5.22: TREC Task 544: Starting from targets (186 starts) and searching to depth 5. Cumulative performance normalized by cone size viewed at different scales.

	Gold Depth	Gold Discount	Depth SVM 2	Discount SVM 4	Random	BFS	DFS
Gold Depth	0	0.101	1.1e-29	1.17e-29	0	0	0
Gold Discount	0(0.253)	0	1.52e-29	1.6e-29	0	0	0
Depth SVM 2	-9(-59.5)	-9.5(-59.8)	0	0.121	7.29e-201	3.51e-110	1.88e-288
Discount SVM 4	-8(-59.3)	-8(-59.6)	0(0.237)	0	6.32e-200	1.86e-107	1.16e-287
Random	-54(-109)	-54(-109)	-22(-49.5)	-24(-49.7)	0		
Breadth First	-34(-81.6)	-34(-81.9)	-6(-22.1)	-7(-22.3)		0	
Depth First	-94(-180)	-94(-181)	-52(-121)	-53(-121)			0

Table 5.14: TREC Task 544: 186 Target pages to depth 5. Actions to First Target comparing

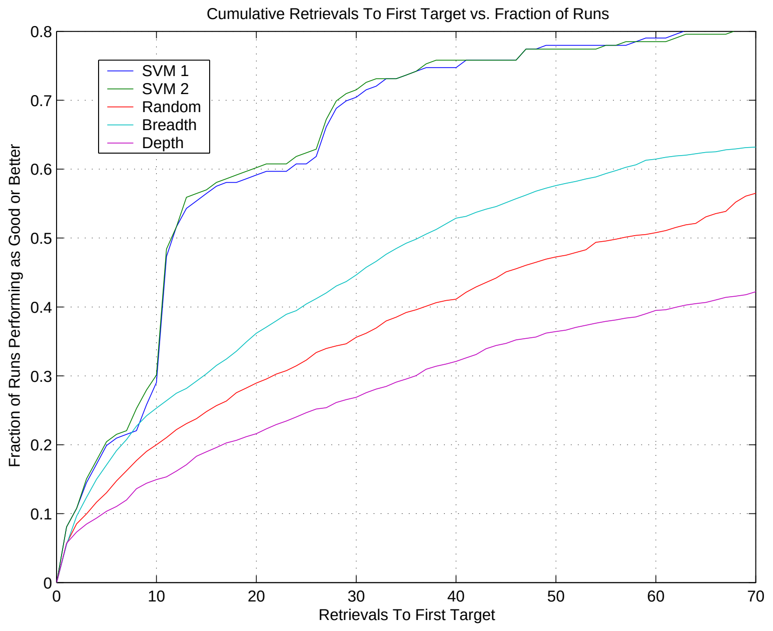


Figure 5.23: TREC Task 544: Starting from targets (186 starts) and searching to depth 5. Cumulative performance.

Figure 5.24 summarizes our exploitation results. This chart dramatically emphasizes the performance of our SVM based spiders. We can see that the SVM spiders are finding the first target 20% more often in the first 10% of the search vs. the best naive search, the breadth-first search.

Discussion

In the above experiments, we analyzed the ability of the various spiders to exhaustively search a constrained environment. This limits the ability of a spider to exploit the learned signal—it can't follow the scent as it is blocked by the depth boundary— It's kind of like a blood-hound that has to work on a leash: It can run down the scent until it is pulled up short by the leash and then it has to back up and try to track in from a different direction.

We found that across the board the SVM based spiders were able to out-perform in median (sometimes without significance) the naive spiders. The most interesting and most challenging task was searching from very close (height-3) from the target pages.

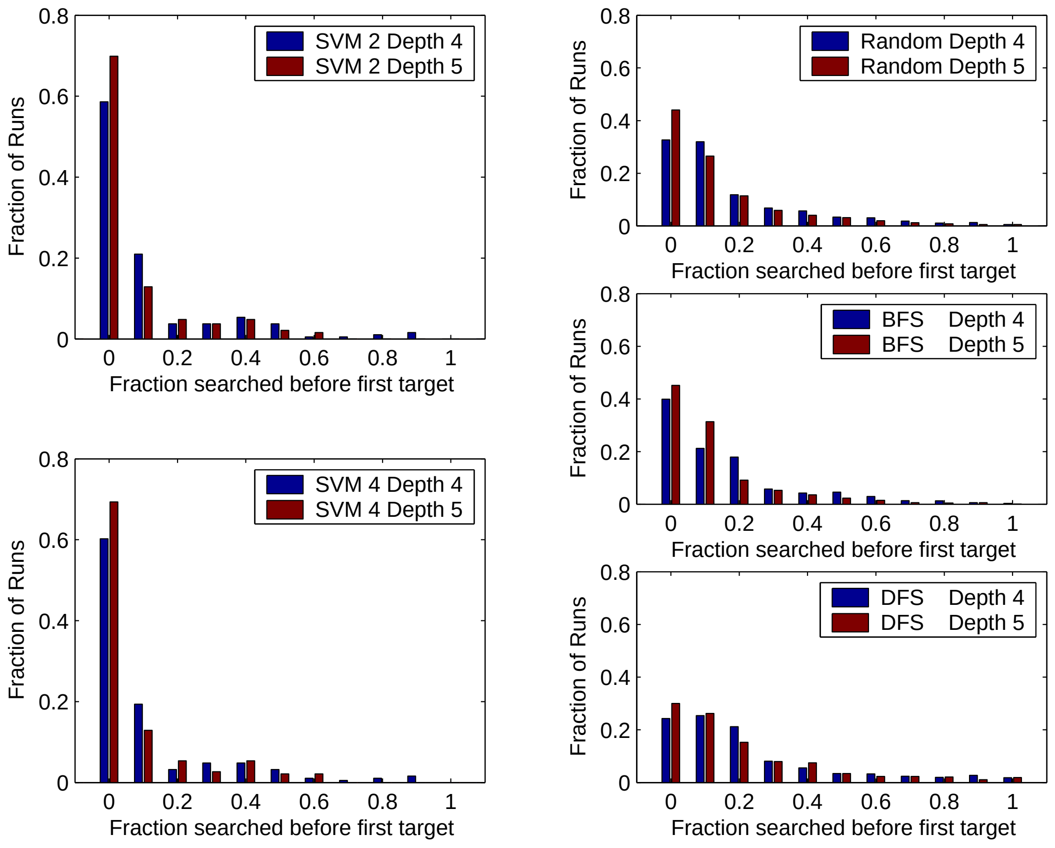


Figure 5.24: TREC Task 544: Starting from targets (186 starts) and searching to depths 4 and 5. Distribution of page retrievals to find first target divided by total pages in search cone.

Here, with a shallow search-cone, the depth-first search has a natural advantage; with a deeper search-cone, the breadth-first search comes into its own. Even so, our smart spiders still do well in this environment.

The spiders were tested in both exploration and exploitation tasks. In both classes of tasks the smart spiders performed well. This is important if we consider macro actions for optimizing our search. One macro action might be to run a fixed depth exploitation search every time a target is found. If we don't have a smart spider available we might select between depth-first and breadth-first based on our budgeted depth of search-cone for the macro action. We might even consider a random-search, perhaps with adjusted weights pushing towards breadth- or depth-first. The random-search is interesting as it universally performed between depth- and breadth-first search on all experiments and might make a good compromise when the nature of the target environment is unknown.

On the other hand, if a smart spider is available, it is better to use that for the macro action. The macro action might not even be helpful as the smart-spider might be better off exploring higher scoring links in other parts of the graph, since the smart-spider has a more (as compared with the naive spiders) global view of the state of the fringe. This will be discussed more in our *Future Work* chapter.

You probably noticed that the smart spiders performed very close to identically whether they simply scored links using the base page with amplified anchor text or they used the amplified gradient scores. Also, the spiders trained against the depth gold standard and the discounted reward gold standard heuristics performed very similarly. The only difference was that the discounted reward gold standard was a touch more robust in the exploitation phase towards wasting time following links to already discovered targets.

5.1.3 Long-range Evaluation

We saw above that, in the short-range searches, sometimes the naive search strategies were competitive with or even out-performed (in mean, and without significance in median) the SVM trained spiders. The constrained search area gives guaranteed performance to the naive spiders while preventing the SVM spiders from taking advantage of paths to targets potentially discoverable outside of the fixed-depth cone.

In this section we explore the performance of the various spider types once we relax the depth constraints. As the search space grows, we hope that the SVM spiders using their estimates of the gold standard heuristics will begin to vastly outperform the naive searches. What can we expect to see happen? As we increase the allowed search depth, the search space increases (loosely) exponentially, dramatically overwhelming the number of targets reachable. In other words, as the depth of the search-cone increases, the ratio of target pages to non-target pages approaches zero. Hence we would expect the depth first search to start doing especially bad. It should just wonder off into limbo.

On the other hand, if we start out close enough to a target, the breadth first search is guaranteed to find targets through exhaustive search. *Close enough* just means that the search horizon is larger than the number of pages closer to the initial page vs. the first target. So if we start our search only 4 hops away from a target we **will** find at least one target within a few hundred to a thousand retrievals. If our search horizon isn't large enough, then the breadth first search will never find any targets.

We'd expect the random spider to perform somewhere in between the performance of the depth first and breadth first spiders: It's randomness forces it to spend effort expanding the fringe in the neighborhood of the starting pages thereby finding the close targets while also searching deeper for other targets and hence also risking getting lost.

We perform several long-range experiments. First we will start at the same distance from targets as we used in our highest depth exploration phase (height-4), however we will triple our search depth to 15. Also we will no longer exhaust our search space as this would require reading and parsing hundreds of thousands of web pages many many times. Instead we will fix our horizon. That is, we retrieve a fixed number of web pages, vs. a fixed depth cone as we did in the short-range tests. We start the experiments relatively close to the target as in the short-range tests, but with greater allowed search depths. In subsequent experiments, we increase the distance from target as well as the size of the horizon.

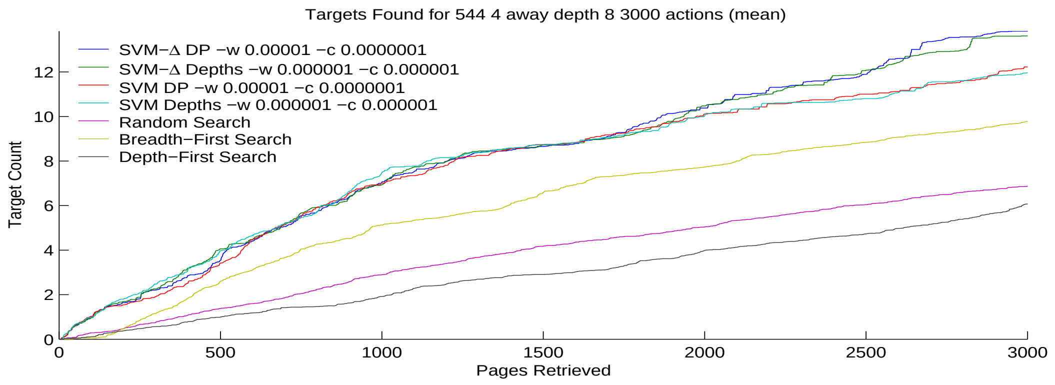
In each set of the experiment sets we run each of the SVM based spiders once. In some cases we run with only the augmented-delta and sometimes with both. We don't always evaluate the non-delta version as the performance doesn't appear significantly different from the augmented-delta version. Also for each set of experiments we run two tests each for breadth-first, depth-first, and random search. In the breadth-first and depth-first search, the randomization shuffles the nodes discovered so each execution comes out different. The seeds are fixed to allow reproducible results. We only run two of these tests vs. the 20 in the short-range tests for speed of execution, especially as we increase the horizon. We do not make multiple runs of the SVM based spiders as they are fully deterministic once the machine is trained. As we step through our experiments, you will notice that some differently parameterized SVM machines are used in some of the tests. These results are presented for completeness, and to emphasize that parameterization *does* impact spidering efficacy.

From 4 Away

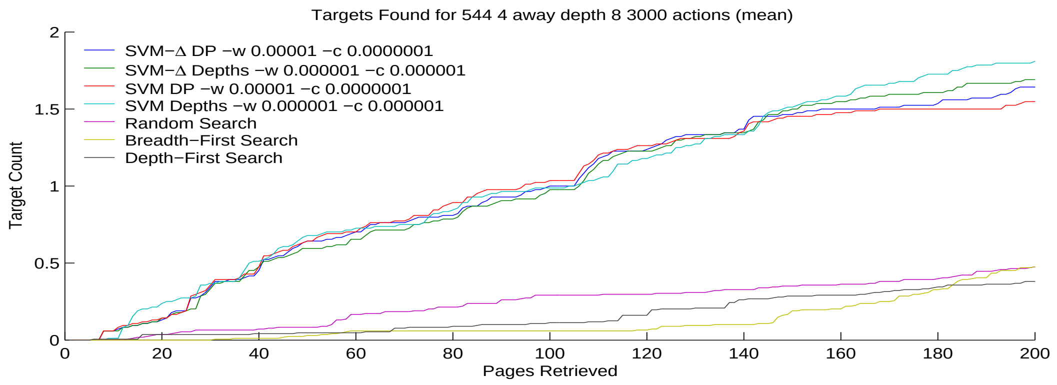
While the short-range results discussed above did support our efficient spidering hypothesis, the performance advantage for the trained spiders was not as strong as we might have hoped. In addition, much of the spidering research to date has focused on starting relatively short distances from targets. Before extending the distance to targets we decided to do a prototype run to see if our spiders could take advantage of additional search depth while starting from the same initial conditions. We limited the horizon to only $3,000$ page retrievals in interest of speed of testing.

In our first test we opened up the search-cone to depth-8 instead of only 4 or 5. We used the same starting pages as were used in Section 5.1.2 with one modification: Those starting pages whose search-cones "dead-ended" in less than $3,000$ pages were removed from the experiment. Since we are starting close to the target, the breadth-first search will be guaranteed to find targets relatively early in the search process and may prove to be tough competition. Let's see how it worked out.

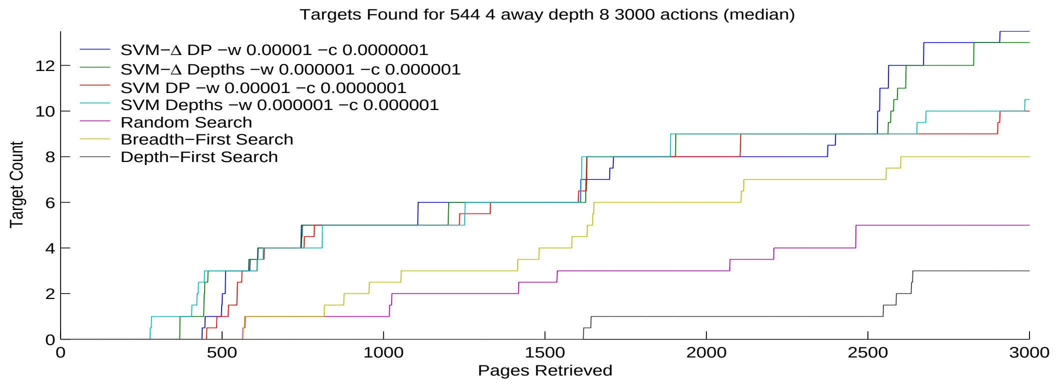
Figure 5.25 shows the performance of our spiders searching in a depth-8 search-cone with a $3,000$ page horizon. Notice that the breadth first, depth first, and random spiders perform just as we would expect with the breadth first search outperforming the other naive spiders. Also the SVM trained spiders out-perform even the breadth first search spider by an almost 50% margin. The difference in median between the SVM spiders and the best performing naive search is significant according the



(a) All 3000 retrievals



(b) The first 200 retrievals



(c) All 3000 retrievals, Median

Figure 5.25: TREC Task 544: Starting at pages 4 away (86 starts) to depth 8 for 3000 page retrievals. Mean and median performance of trained and naive spiders.

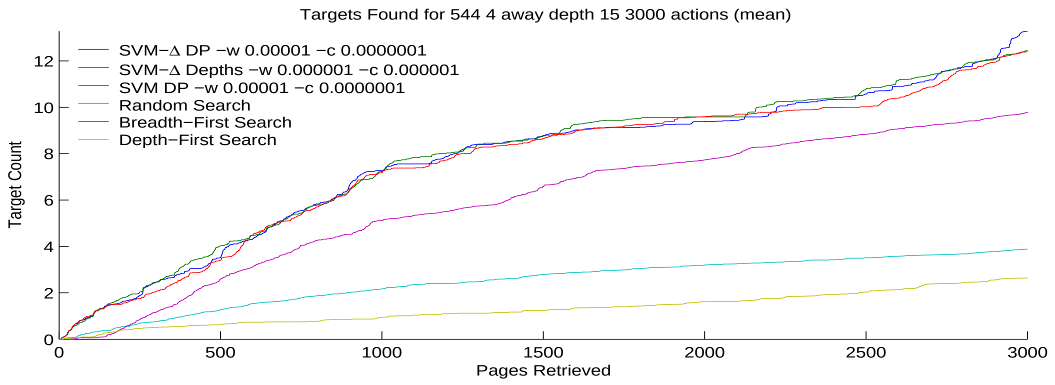
Wilcoxon test with far better than the 95% confidence. In the second portion of the figure we zoom in on the first 200 page retrievals. We can see that the SVM spiders "take off running" and perform several times better then the naive spiders in the first portion of the run. Look at the breadth-first search trace (in yellow). Notice that it passes the random and depth-first traces at just under 200 retrievals. That is, it passes random and depth-first once it has exhausted the empty layers before the target layer. In the end the breadth-first search performs at about 75% relative to the SVM-based searches.

The final plot in Figure 5.25 shows the median number of targets found. The median is very difficult to read because of its discrete nature, the plot of means is much easier to read. One thing that can be read off of this plot is the median time to first target. Looking closely, we can see that the SVM spiders are finding the first target much quicker than the naive spiders.

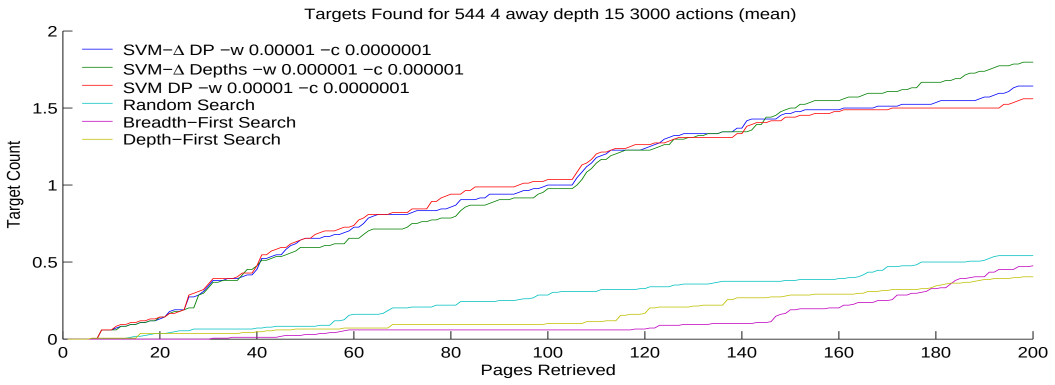
Next we run the same experiment, but this time allowing the spiders to search a depth-15 search-cone, still with a $3,000$ page horizon. We can see the results of this search in Figure 5.26. The first thing we notice is that the performance of the breadth-first spider is unchanged and that the other naive spiders have dropped off greatly in performance. The other results are basically identical to that of the previous depth-8 test.

In Figure 5.27 we compare the results for the depth-8 and depth-15 searches. Here we can directly compare the results of the two tests. Now that we've opened the depth up, the SVM spiders perform about the same in either search cone and in fact seem to do slightly worse (insignificantly) in the deeper test, although they seem to be narrowing the gap at the end of the horizon. As mentioned above, the main difference is in the performance of the naive algorithms. The depth-first and random searches plummet in performance. The cause of this is obvious: The search space is several times larger at depth-15 vs depth-8.

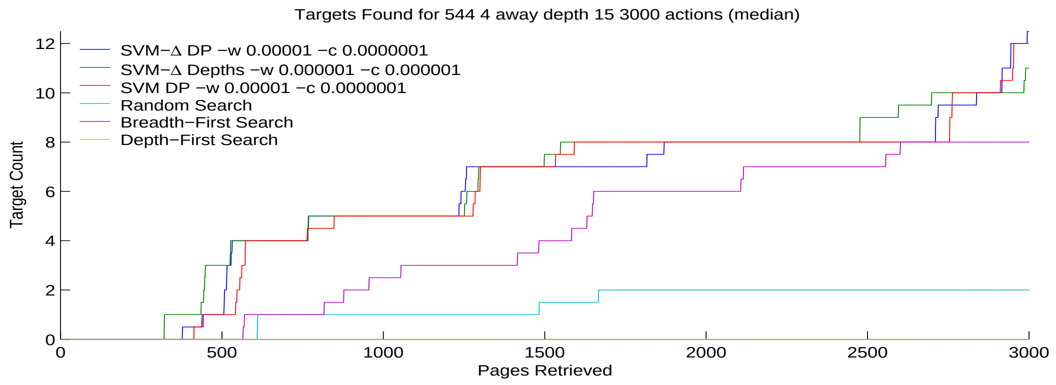
These results give us an indication that opening up the search will show off the capabilities of our learned spiders. It emphasizes what we learned in the short-range tasks; that if we suspect there are targets "nearby" then we should prefer the breadth-first search over the other naive searches and that the random search always seems to perform between the depth-first and breadth-first searches.



(a) All 3000 retrievals

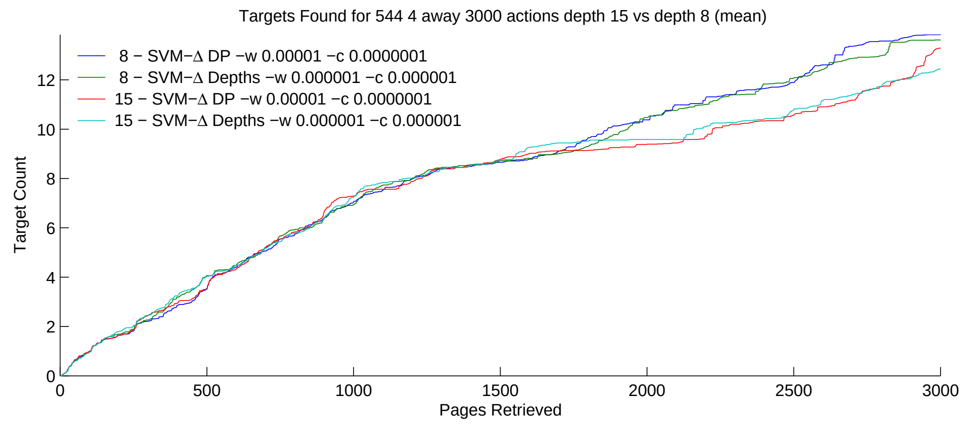


(b) The first 200 retrievals

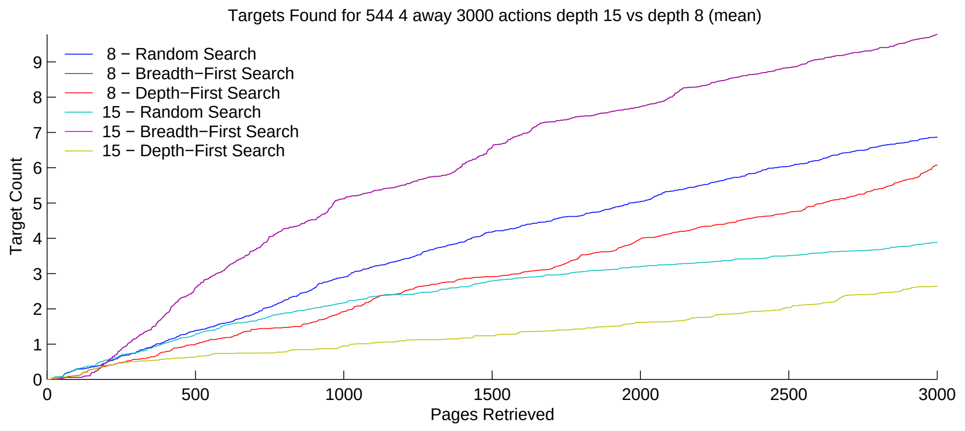


(c) All 3000 retrievals, Median

Figure 5.26: TREC Task 544: Starting at pages 4 away (86 starts) to depth 15 for 3000 page retrievals. Mean and median performance of trained and naive spiders.



(a) Comparison of SVM performance



(b) Comparison of Naive Performance

Figure 5.27: TREC Task 544: Starting at pages 4 away (86 starts) to depths 8 and 15 for 3000 page retrievals. Mean performance comparing the results for the SVM spiders and the Naive spiders. Note that for the naive spiders there are six labels and only 5 plots. This is because, as must be the case, the two breadth-first searches had identical performance.

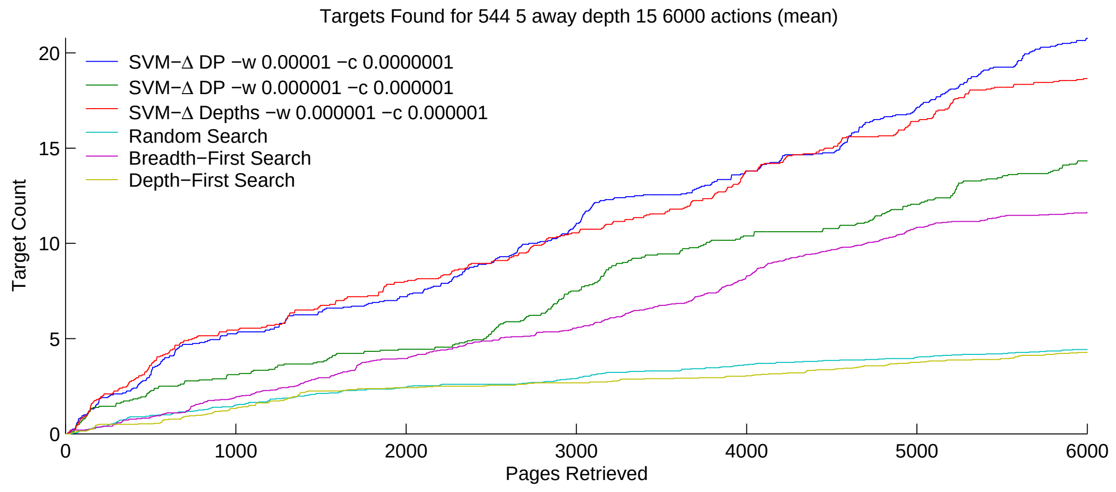


Figure 5.28: TREC Task 544: Starting at pages 5 away (20 starts) to depth 15 for $6,000$ page retrievals. Mean performance of trained and naive spiders.

From 5 away

In the next set of experiments, we move one step further away from the targets such that our starting pages have a target five links away, and none closer. We run our spiders with the search-cone limited to depths of 15 and 30. These experiments were run as a quick exploration to confirm the results with a slight increase in distance to target.

These are the first set of experiments where we are testing the ability of the smart spiders to search in regions of the web-graph they were not trained on. Remember that the spiders were trained on parent-cones of height-4 and we are now starting at height-5. These tests were run primarily as a quick test to confirm the smart spiders still work outside the training domain—akin to training a chess program on end-games and evaluating it on openings. In other words, we are testing the ability of the spiders to generalize their ability to classify pages within the height-4 region they were trained on to be able to get a useful gradient on the navigational distance or discounted reward, even looking at classes of pages they weren't trained on. Being able to do this is critical to successful long-range searching. For these reasons, we only use 20 starting pages.

Figure 5.28 shows the mean number of targets discovered by our spiders running from height-5. We see that the random and depth-first searches have almost leveled

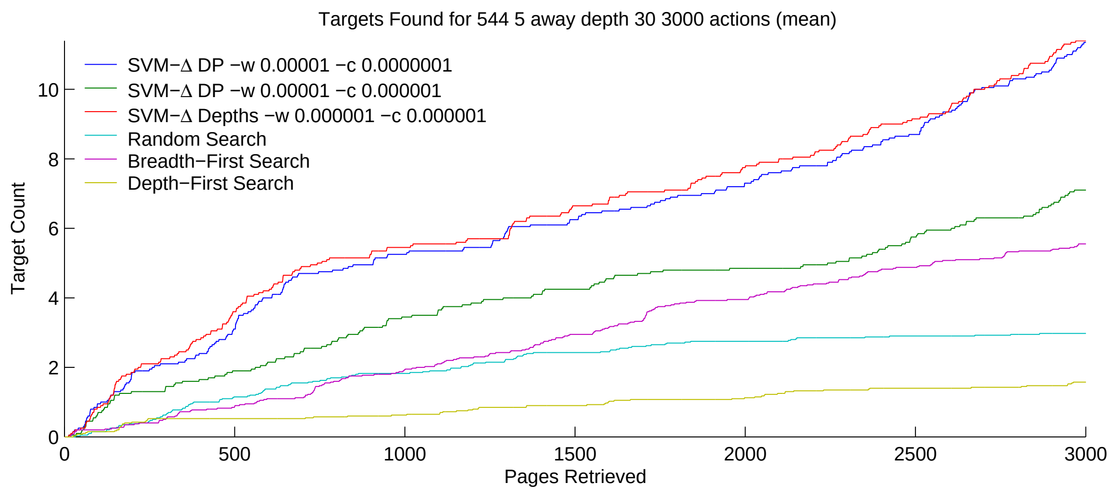
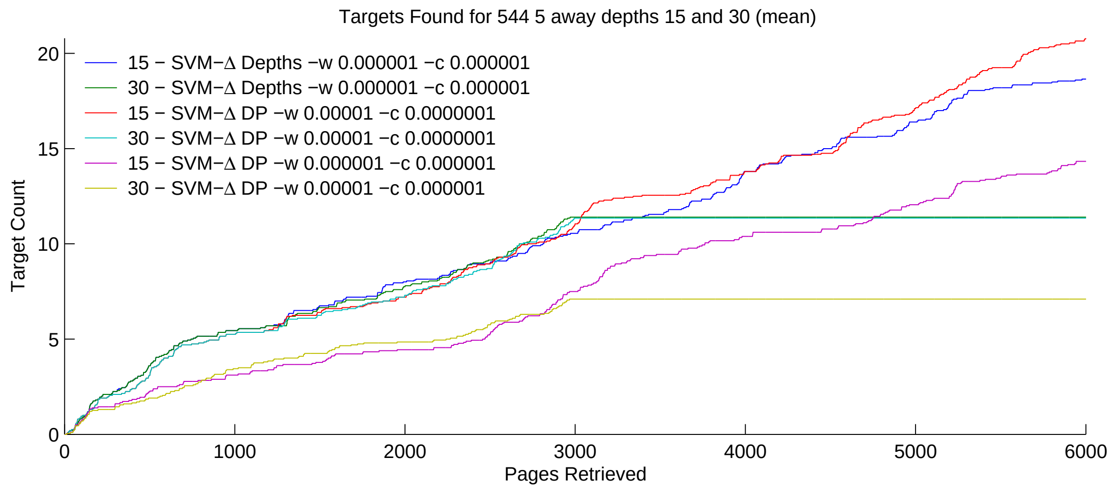


Figure 5.29: TREC Task 544: Starting at pages 5 away (20 starts) to depth 30 for $3,000$ page retrievals. Mean performance of trained and naive spiders.

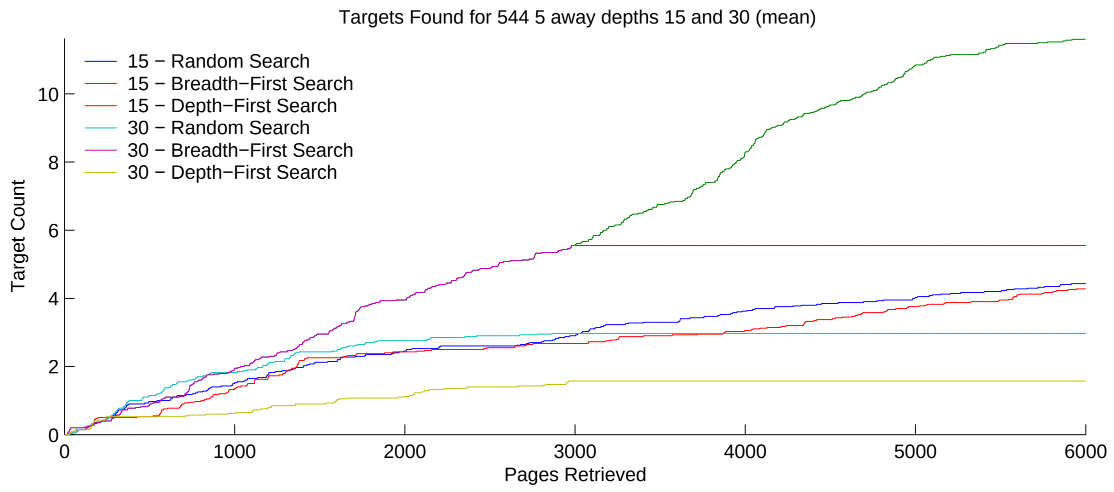
out after only $1,500$ page retrievals. The breadth-first search continues to perform strongly, at about half the rate of the best SVM spiders, but that it is losing ground as the starting height increases. Notice the green plot. This SVM was not used in other tests and is shown here primarily to test an alternately trained machine. Notice that it is performing just slightly better than the breadth-first search, in other words, not all SVM trained machines perform super strongly. Looking at the lower left portion of the graph, we see that even this machine performs very strongly in the initial phase of the search.

Next we relax our constraint on the depth of our search-cone. We will allow the spiders to search out to a depth of 30. First off, we know that the breadth-first search results will not change at all. We would expect the random and depth-first searches to drop off as the size of the search space has increased dramatically (bigger haystack). We only run these tests with a horizon of $3,000$ pages. Figure 5.29 shows our results. The red and blue traces are identical machines to those in Figure 5.28. The green trace is another alternate SVM being tested. Again note that its asymptotic performance drops off, but it still starts out very strongly. Our predictions for the naive spiders pan out. Notice that it now takes almost $1,000$ actions for the depth-first spider to pass the random spider. Remember it was only 200 actions at height-4 (Figure 5.26).

In Figure 5.30 we see a comparison of the performance with the search-cone limited



(a) Comparison of SVM performance



(b) Comparison of Naive Performance

Figure 5.30: TREC Task 544: Starting at pages 5 away (20 starts) to depths 15 and 15 for up to 6000 page retrievals. Mean performance comparing the results for the SVM spiders and the Naive spiders. Note that for the naive spiders there are six labels and only 5 plots. This is because, as must be the case, the two breadth-first searches had identical performance.

to depths 15 and 30. For the depth-30 runs, we only ran with a horizon of 3,000 which is why some of the plots flat-line. Notice that, as must happen, the breadth-first search performs identically regardless of the depth of search-cone. And that the long-range depth-first search performance dropped off quite a bit with the increased depth. For the smart spiders, we see that the change in search depth has little impact on our performance, with an insignificant bias favoring the deeper search depth.

This difference between the smart spiders and the naive spiders is important. The performance of the smart spiders is not being negatively impacted by opening up our search domain, in fact it may be slightly improved. However the depth-first and breadth-first searches are extremely dependent on this constraint and are liable to go wondering off into never-never land.

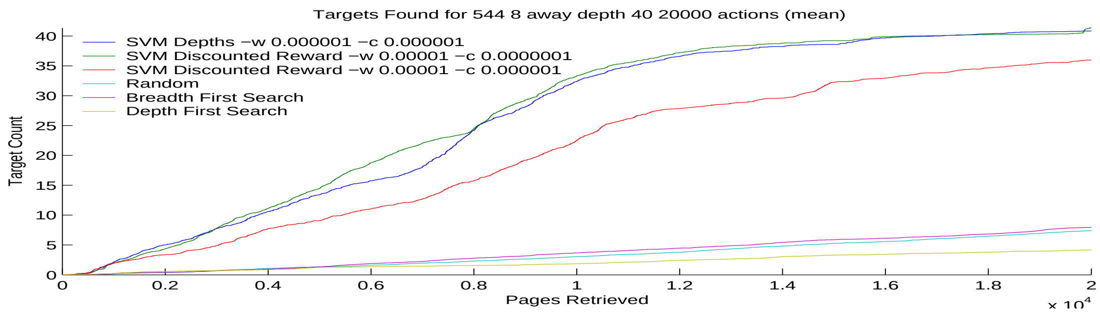
From 8 away

In our previous two experiments we were still starting fairly close to targets but we could already see how the naive searches were being impacted by the increased search distance. We also saw that the performance of the smart spiders was not falling off as quickly.

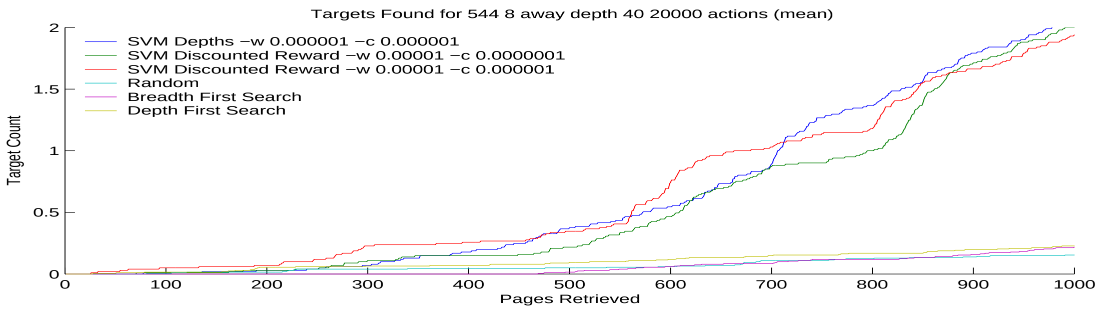
In the experiments in this section, we move out to a distance of 8 hops away from the closest targets and select 101 initial pages at random from the testing dataset. We open up the search-cone to a depth of 40. At the time these experiments were run, we had an artificial limitation of the software in which we had to specify some maximum depth. A depth of 40 is much larger than the expected distance between any two pages in the corpus (only about 19 in the full web). We run each of our spiders with a horizon of 20,000 page retrievals. The spiders are now searching in a space of about two hundred-thousand pages containing at-most 300 targets. A veritable needles-in-a-hay-stack problem.

In Figure 5.31 we show the performance of our three SVM trained spiders as well as that of the naive spiders. As before the SVM spiders outperform the naive spiders, this time with a very heavy margin. Note that the asymptotically, the relative performance of the naive searches are as they always are, breadth-first is best, followed very closely by random with depth-first far behind.

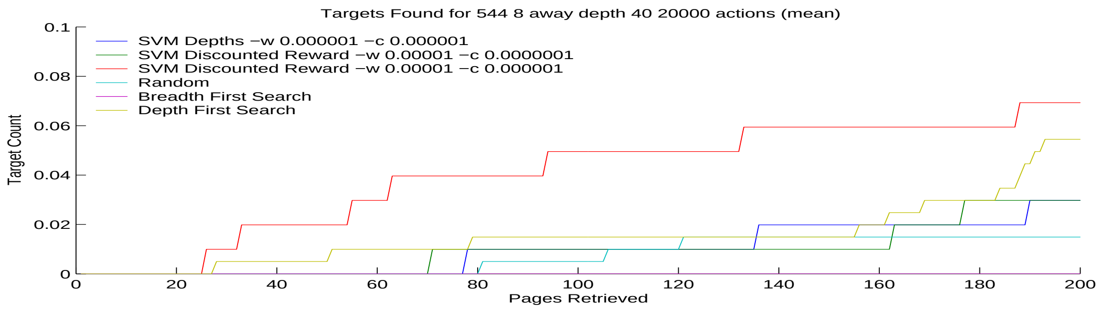
As we zoom in, looking at only the first 1000 retrievals, we see that the SVM



(a) Performance across 20000 actions

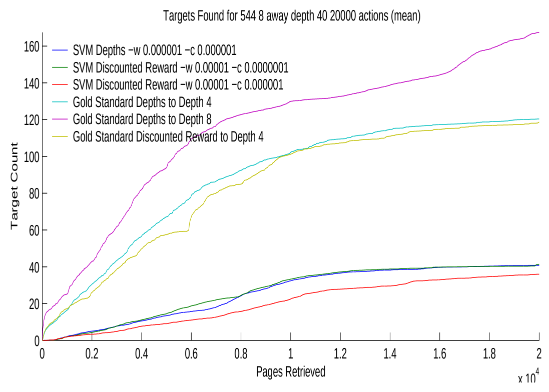


(b) Zoomed in to show detail

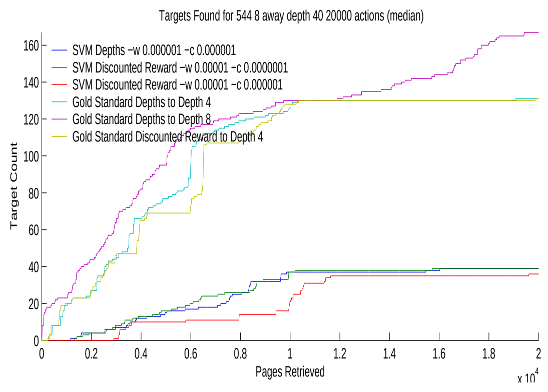


(c) Zoomed in more

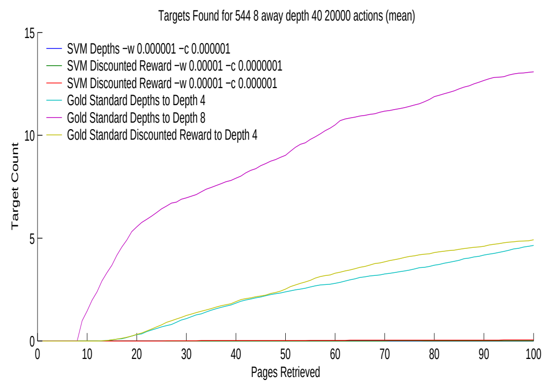
Figure 5.31: TREC Task 544: Starting from 101 starting pages 8 away from first target searching to depth 40 for 20000 page retrievals. Mean performance comparing the results for the SVM spiders and the Naive spiders. The second figure zooms in on the first plot to show the initial performance.



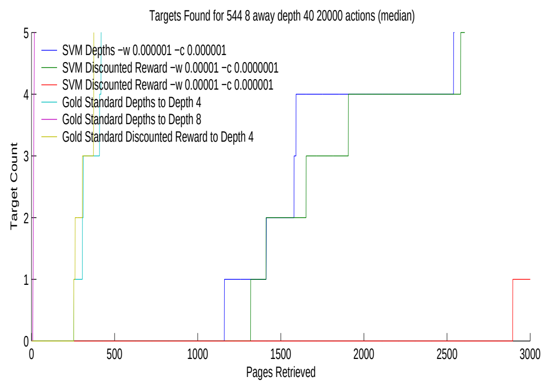
(a) Mean Performance Across 20000 Actions



(b) Median Performance Across 20000 Actions



(c) Mean Zoomed in to show detail



(d) Median Zoomed in to show detail

Figure 5.32: TREC Task 544: Starting from 101 starting pages 8 away from first target searching to depth 40 for 20000 page retrievals. Mean and median performance comparing the results for the SVM spiders and the Gold Standard spiders. The second figure shows zooms in on the first plot to show the initial performance.

based spiders are hitting their stride at around 500 pages leaving the naive searches behind. At around 1000 page retrievals the breadth first search passes the depth first search. When we look at the first 200 retrievals we see that the depth-first search starts picking up targets in the same range as our SVM based searches.

Our SVM based spiders are clearly performing much better than the naive spiders. However, our SVM spiders are trying to emulate the gold-standard spiders which they were trained against. Figure 5.32 shows the performance of the Gold Standard spiders vs. our SVM trained spiders. Since the SVM spiders were trained to calculate regressions against either the depth or discounted reward to height-4 in the training set, we use gold spiders with perfect knowledge of height-4 for both depth and discounted reward. One difference is that these gold spiders get no signal when they are more than four hops away from a target whereas the SVM spiders might be able to estimate some signal. We also include a gold standard spider using perfect knowledge of the depth information out to a height of 8 away from targets. This is the violet line in the plots. The plots on the right side show the median performance. Looking at the zoomed in median plot we can see the median number of pages retrieved to find the first target. The gold-standard depth-8 spider is finding the first target in only 8 actions. The other two are finding the first target at about 250 actions, and then the SVM spiders are finding the first target much later, between about 1200 to 2800 page retrievals depending on the spider.

This difference in performance between the gold-standard spiders and the SVM spiders shows that we have a lot of room for improvement. In the same horizon the gold-standard spiders are finding three to four times the targets. On the other hand, looking back at Figure 5.31 our SVM spiders are finding about eight times as many targets as the best naive spider.

5.2 The Other Tasks: Frogs and Weather

In Section 5.1 we presented our results on the TREC 544 task. In this section we present our results for tasks 541 (Weather Instruments) and 519 (Frog Habitats).

Our learning techniques on these TREC Tasks were far less effective than on the 544 task. We first present our results on the Weather Instruments task followed by

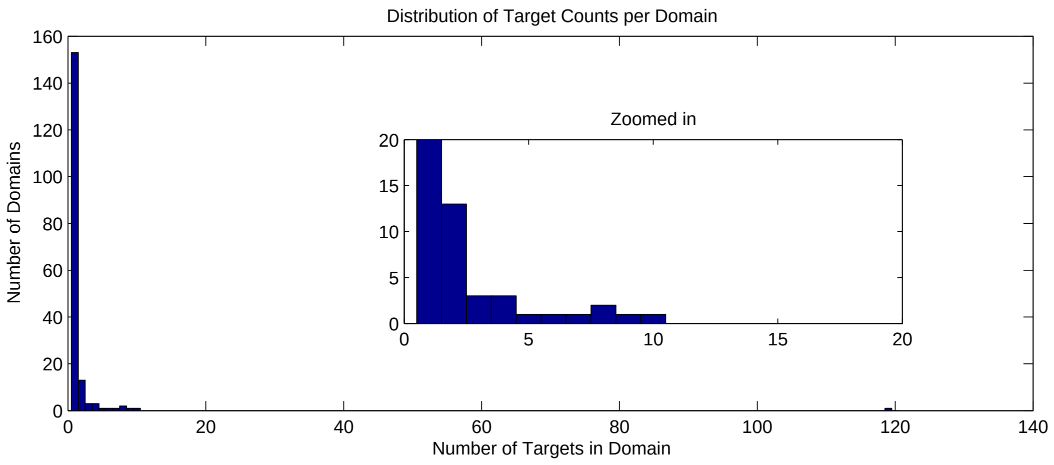


Figure 5.33: Distribution of the number of targets per domain.

our discussion on the Frog Habitats task. In each discussion we develop our training techniques used and our spidering results.

Our discussion is grounded in the differences between these tasks and the 544 task. The distribution of targets in these tasks as well as the web environments of the target pages for these tasks are rather different from those of task 544. These differences may explain some of the difficulty in developing effective spiders.

5.2.1 Searching for Instruments for Forecasting Weather

In this section we discuss our results for TREC task 541:

- **Description:** What instruments are used to forecast the weather?
- **Narrative:** A relevant document will state that a particular instrument is used to forecast weather. Weather forecasts themselves are not relevant.

This task has 372 different labeled targets distributed across 180 domains. It is the largest of the 50 TREC tasks. This number of targets is about the same as that for the 544 task presented above, however there are about twice as many domains. Thus the targets are more diffusely distributed. Because targets are distributed across more domains, it may be harder to learn a regression. Figure 5.2.1 depicts the distribution

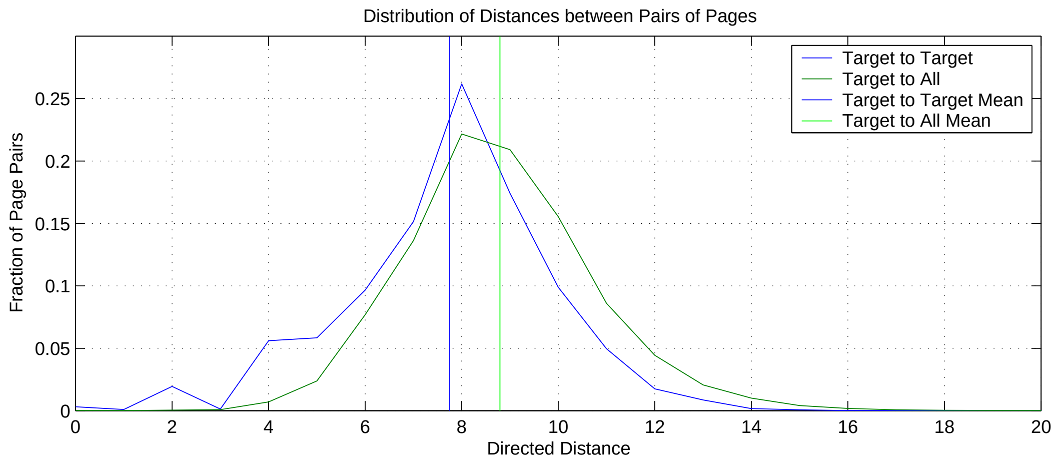


Figure 5.34: Distributions of the Diameter of the Web-graph as observed from the TREC Task 541. The mean target-to-target distance is 7.75 and the mean target-to-all distance is 8.79.

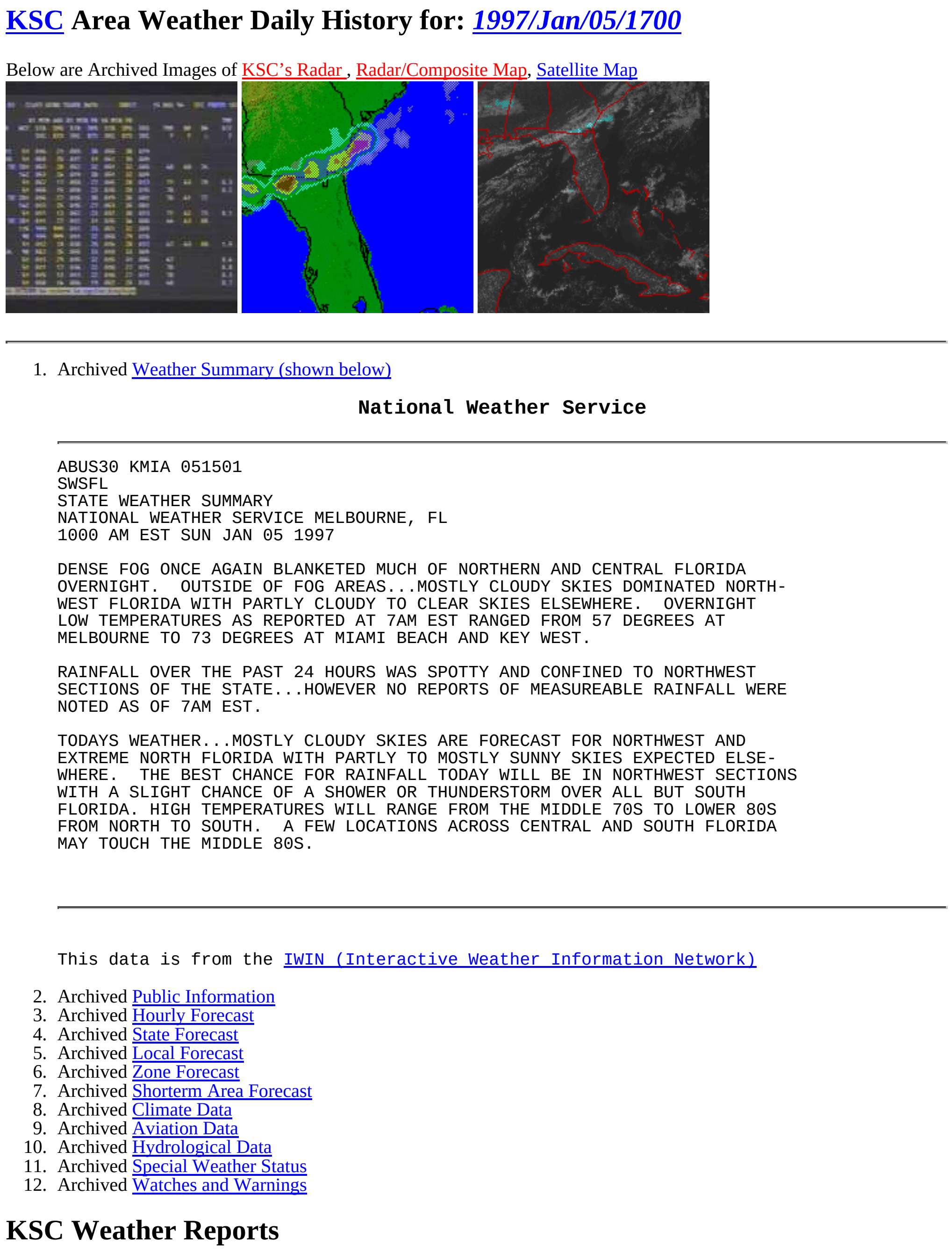
of targets across domains. For example, there is one domain containing almost 120 targets, and about 150 domains each containing only one target. The sub-plot shows the same data zoomed in to the origin. Comparing against the distribution for Task 544 shown in Figure 5.1, we see that Task 541 has far fewer pages per domain, discounting the one huge domain.

The 120 target domain is a NASA collection of weather reports for 1996 and 1997. An example report is shown in Figure 5.35. The URL is at:

`http://science.ksc.nasa.gov/weather/1997/Jan/05/1700/wx.html`

and is still active on the live web at the time of writing. Upon examining these pages, I disagree these pages satisfy the target criteria discussed above although it does mention the term radar on the same page as a weather report. Other pages in this task are similar, that is, weather reports that contain the word radar; while others are merely lists of links to various weather resources. There is an indication that the TREC authors recognize the low quality of most of the targets for this task.

The labeled pages for each task are classed into three categories: 0 for non-targets, 1 for weak targets, and 2 for strong targets [58]. For the Weather Instruments task, only 2 pages were labeled as strong targets. Figure 5.36 shows



KSC Area Weather Daily History for: *1997/Jan/05/1700*

Below are Archived Images of KSC's Radar , Radar/Composite Map, Satellite Map

1. Archived Weather Summary (shown below)

National Weather Service

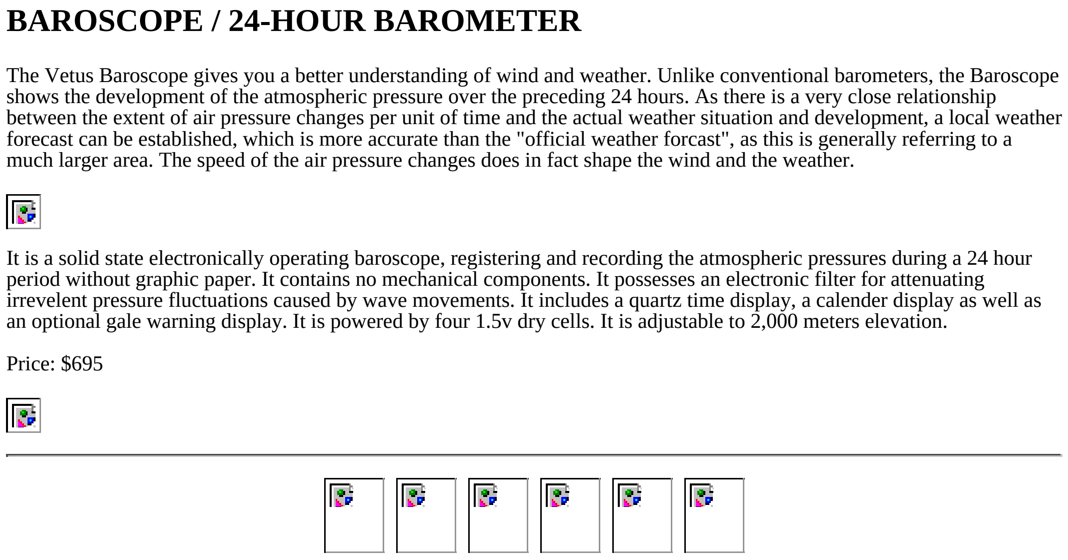
```
ABUS30 KMIA 051501
SWSFL
STATE WEATHER SUMMARY
NATIONAL WEATHER SERVICE MELBOURNE, FL
1000 AM EST SUN JAN 05 1997

DENSE FOG ONCE AGAIN BLANKETED MUCH OF NORTHERN AND CENTRAL FLORIDA
OVERNIGHT.  OUTSIDE OF FOG AREAS...MOSTLY CLOUDY SKIES DOMINATED NORTH-
WEST FLORIDA WITH PARTLY CLOUDY TO CLEAR SKIES ELSEWHERE.  OVERNIGHT
LOW TEMPERATURES AS REPORTED AT 7AM EST RANGED FROM 57 DEGREES AT
MELBOURNE TO 73 DEGREES AT MIAMI BEACH AND KEY WEST.

RAINFALL OVER THE PAST 24 HOURS WAS SPOTTY AND CONFINED TO NORTHWEST
SECTIONS OF THE STATE...HOWEVER NO REPORTS OF MEASUREABLE RAINFALL WERE
NOTED AS OF 7AM EST.

TODAYS WEATHER...MOSTLY CLOUDY SKIES ARE FORECAST FOR NORTHWEST AND
EXTREME NORTH FLORIDA WITH PARTLY TO MOSTLY SUNNY SKIES EXPECTED ELSE-
WHERE.  THE BEST CHANCE FOR RAINFALL TODAY WILL BE IN NORTHWEST SECTIONS
WITH A SLIGHT CHANCE OF A SHOWER OR THUNDERSTORM OVER ALL BUT SOUTH
FLORIDA. HIGH TEMPERATURES WILL RANGE FROM THE MIDDLE 70S TO LOWER 80S
FROM NORTH TO SOUTH.  A FEW LOCATIONS ACROSS CENTRAL AND SOUTH FLORIDA
MAY TOUCH THE MIDDLE 80S.
```

This data is from the IWIN (Interactive Weather Information Network)

2. Archived Public Information
3. Archived Hourly Forecast
4. Archived State Forecast
5. Archived Local Forecast
6. Archived Zone Forecast
7. Archived Shorterm Area Forecast
8. Archived Climate Data
9. Archived Aviation Data
10. Archived Hydrological Data
11. Archived Special Weather Status
12. Archived Watches and Warnings

KSC Weather Reports

Figure 5.35: An example weather report from the large 120 target NASA weather report collection. Is it really a target for TREC Task 541? Note that the radar images are not in the WT10g corpus. The images come from the live web page and are shown to give a better indication of the content of the page.



BAROSCOPE / 24-HOUR BAROMETER

The Vetus Baroscope gives you a better understanding of wind and weather. Unlike conventional barometers, the Baroscope shows the development of the atmospheric pressure over the preceding 24 hours. As there is a very close relationship between the extent of air pressure changes per unit of time and the actual weather situation and development, a local weather forecast can be established, which is more accurate than the "official weather forcast", as this is generally referring to a much larger area. The speed of the air pressure changes does in fact shape the wind and the weather.

It is a solid state electronically operating baroscope, registering and recording the atmospheric pressures during a 24 hour period without graphic paper. It contains no mechanical components. It possesses an electronic filter for attenuating irrevelent pressure fluctuations caused by wave movements. It includes a quartz time display, a calender display as well as an optional gale warning display. It is powered by four 1.5v dry cells. It is adjustable to 2,000 meters elevation.

Price: $695

Figure 5.36: An example strong target page for TREC Task 541. The missing image markers are shown as this page no longer exists on the live web and the WT10g corpus does not contain images.

one of these pages. Note how much better this page fits the task. The URL is `http://www.bkbank.com/Users/creative/barometr.htm`.
Compare this with the ratio of strong to weak targets in the 544 Task. The ratio there was 135 to 189—almost 1 for 1 vs 1 to 150 in the 541 task.

Building the Machines

In our first attempts to train machines for the Weather Instruments task we followed the same pattern as worked so well for the Role of Estrogen task. We formed our training set by picking ten pages from the target set and built a parent search-cone around those pages to a height of four above the targets.

```
http://black.missouri.edu/mu/other_weather.html
http://hpcc1.hpcc.noaa.gov/oar.html
http://inspire.ospi.wednet.edu:8001/curric/weather/intlweat/mexicar.html
http://minfonet.com/forecast.html
http://telnet.newsnet.com/libiss/ae08.html
```

```
Machine 1:  1^0^0^ss_10000^ss.svmlight.svm-z r -w 0.001 -c 0.00005^ss.widx
Machine 2:  1^1^0^ss_10000^ss.svmlight.svm-z r -w 0.001 -c 0.00005^ss.widx
Machine 3:  1^0^1^10_d4_10000^10_d4.dp_svmlight.svm-z r -w 0.00001 -c 0.0000001^10_d4.widx
Machine 4:  1^1^1^10_d4_10000^10_d4.dp_svmlight.svm-z r -w 0.00001 -c 0.0000001^10_d4.widx
Machine 5:  1^0^0^10_d4_10000^10_d4.svmlight.svm-z r -w 0.000001 -c 0.000001^10_d4.widx
Machine 6:  1^1^0^10_d4_10000^10_d4.svmlight.svm-z r -w 0.000001 -c 0.000001^10_d4.widx
```

Table 5.15: Support Vector Machine trained machines used for TREC task 541. Machines 3 and 4 are trained against discounted reward while all the others are trained against depths. The even machines use augmented delta scores when evaluating links during spidering.

`http://www.awc-kc.noaa.gov/publications/Evenson/dry.html`

`http://www.dircon.co.uk/lockes/forecast.htm`

`http://www.ksc.nasa.gov/weather/1996/Dec/30/0100/wx.html`

`http://www.ksc.nasa.gov/weather/1996/Jul/13/wx.html`

`http://www.s-t.com/daily/03-96/03-31-96/1weathyu.htm`

The height-4 search cone for these ten pages contained an additional 8 pages all from the same domain:

`http://inspire.ospi.wednet.edu:8001/curric/space/solterr/spacewx.html`

`http://inspire.ospi.wednet.edu:8001/curric/space/solterr/todaysp.html`

`http://inspire.ospi.wednet.edu:8001/curric/weather/adptcty/cantask1.html`

`http://inspire.ospi.wednet.edu:8001/curric/weather/index.html`

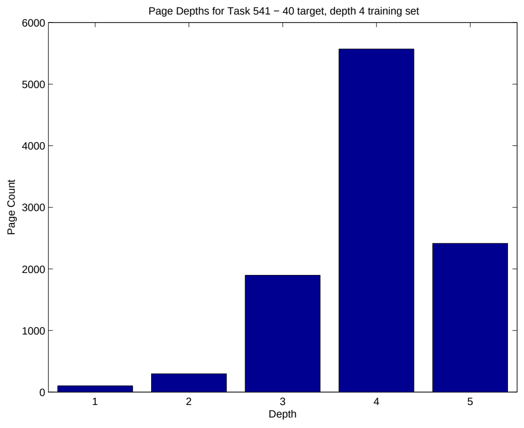
`http://inspire.ospi.wednet.edu:8001/curric/weather/intlweat/africa.html`

`http://inspire.ospi.wednet.edu:8001/curric/weather/intlweat/asia.html`

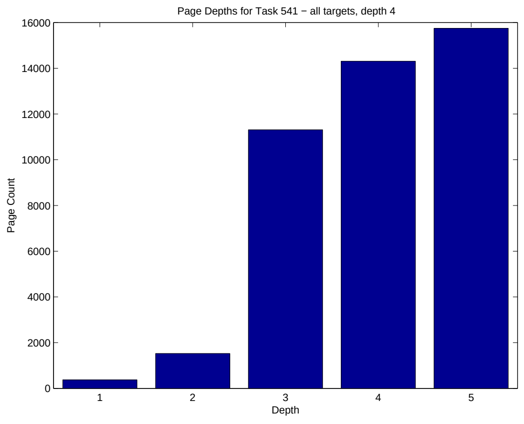
`http://inspire.ospi.wednet.edu:8001/curric/weather/intlweat/aust.html`

`http://inspire.ospi.wednet.edu:8001/curric/weather/intlweat/canada.html`

This gave us a training set of 18 pages, approximately the same size as used in our experiments with the 544 task. The resulting search cone had a total of $1,012$ pages vs. $2,171$ pages in 544. As in 544 we tried training the linear support vector machines to perform regression against both the depth and approximate discounted rewards. After training many machines with different parameters, we selected a couple that got the least error against the training set and used them for spidering. Looking in Table 5.15, these are machines 3 through 6. These machines performed no better than, and often worse than the naive spiders. They just didn't seem to be able to learn a good signal.



(a) Training Set



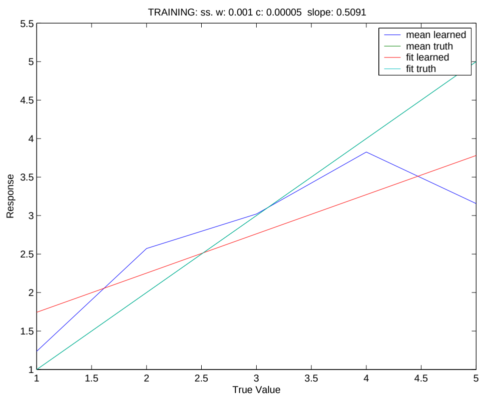
(b) All Targets

Figure 5.37: Distribution of pages at each level in the parent search-cone for the 40 starting target training set and for starting from all targets.

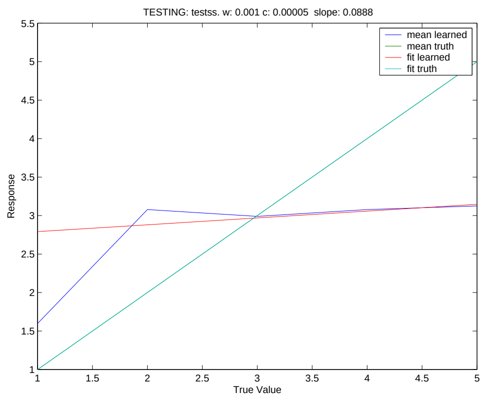
One problem we considered was that perhaps the depth labels on the training pages were exaggerated. This can happen in a parent search-cone when a page, say at height-4, has a nearby, say at depth-2, descendant which is a target. By running through several cycles of alternately pulling the parent search-cone from the target set followed by performing a child search from each page found looking for targets, we built a new target set with completely accurate depth labels. Unfortunately this dataset was largely useless for us as obtaining it covered a significant portion of the WT10g dataset and contained 169 of the 372 targets. In other words, it violated our criteria for our training set desired to demonstrate the spidering hypothesis.

To work around this, we instead selected a larger starting set of targets containing 40 targets selected in an ad hoc manner. The targets were spread across 31 domains. From this set of targets we pulled the training search-cone to a height of 4 above the targets. This found an additional 62 targets. This cone consisted of $10,281$ pages distributed as shown in Figure 5.37. Note that this training set consists of about 25 percent of the full height-4 dataset. Although this training set contains 102 targets, about a third of the total, it did not require exploring an overly large portion of the corpus to find. We did not reiterate growing the parent search-cone from the 62 new targets.

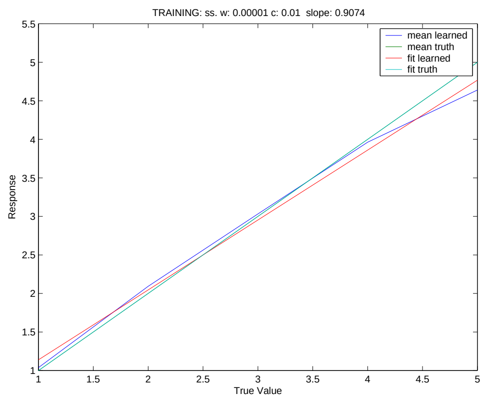
We tried training SVMs directly against this dataset however, because so much of



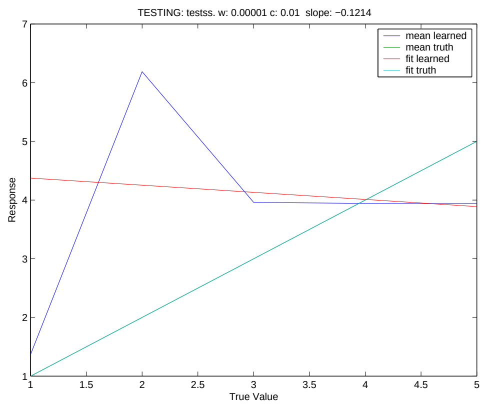
(a) Training Set



(b) Testing Set



(c) Training Set



(d) Testing Set

Figure 5.38: SVM performance against training testing parent search-cones. The top pair of results is for the machine selected for our experiments presented below. The bottom pair of results is for the machine with the lowest L1 loss against the training data set and demonstrates the problem of over-fitting.

the mass is concentrated at one height, we found that the support vector machines tended to converge to returning the mean height (very close to 4) for the regression regardless of the input. To overcome this lack of discrimination, we sampled with replacement $1,000$ pages from each depth; oversampling from the low-height pages and under sampling from the high-height pages.

Using this $5,000$ page training set we trained 15 different machines with SVM^{light} using different parameters for the regression. Particularly we varied the epsilon width of the tube for regression ("-w") and the trade-off between training error and margin ("-c"). We tested each of these machines, both against the training set and against a test set created by subtracting pages found in the parent search-cone searching from the starting 40 targets from the search-cone created from all of the targets.

In order to select a good proof of concept machine to test spidering on this task, we selected a machine trying to avoid over fitting the training data. In Figure 5.38 we see the output of the learned regression for the training set and testing set for two different machines. The top machine is the one referred to as "Machine 1" and "Machine 2" in Table 5.15. The bottom machine is one with the lowest training error. Each figure has four plots. The blue plot shows the mean response for each input having a particular true height. The dark green plot shows the response of a perfect regression, i.e., response equals input. The red line is a best-fit line fitting the response for the learned machine. Note that it fits the response not the mean shown in the blue plot. The light blue plot (always coincident in these diagrams with the green plot) shows the best-fit line against the perfect regression.

Remember that in our spidering we do not need an accurate regression. What we need is a regression with a slope with the same orientation (up or down) as that of the input since the regression is used to provide a total order for the priority queue. So looking at 5.38(b) we see that the output curve for the testing set doesn't match the input very well, it does have the same slope. Now looking at 5.38(a) and at 5.38(b), notice that the SVM learned the input data very well, however it was unable to generalize to the testing set—the slope is actually negative.

As we will see in the next two sections on short- and long-range evaluation the machine learned here actually does work, albeit not quite as dramatically as the easier to build machines for the estrogen task.

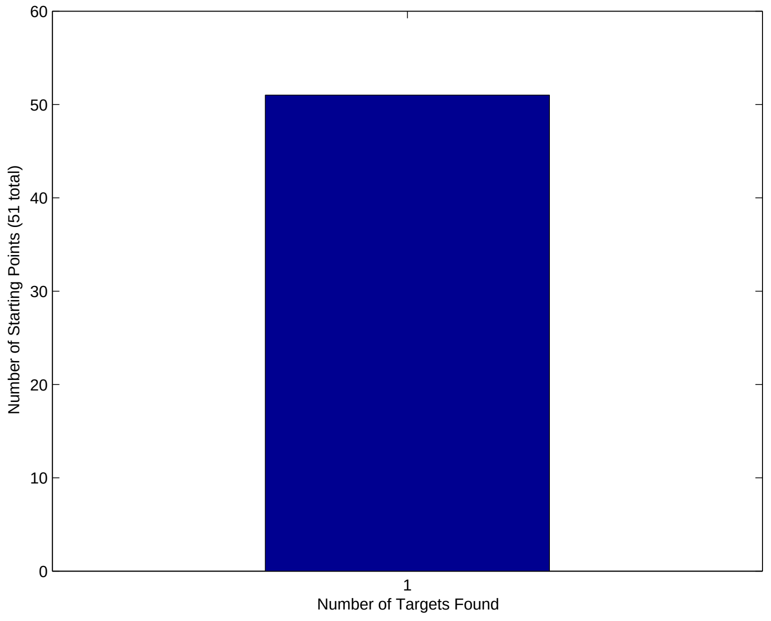


Figure 5.39: TREC Task 541: Starting at pages 3 away (51 starts) to depth 3 searching to exhaustion. Distribution of targets found. Mean number of targets: 1, median: 1. Mean number of pages in search-cone: 1078.8, median: 473.

Short-range Evaluation

In these experiments we use the same testing strategies as used in for testing the spiders against Task 544 as discussed in 5.1.2. In the first set of experiments, we are testing the exploration capabilities of the spiders. We start our experiments very close to the targets and gradually increase our search depth and our starting height. After the exploration tests, we examine the exploitation abilities of our spiders against the "Weather Instruments" task.

We begin our experiments at a height of three above the closest target and search to a depth of three. We start from 51 randomly selected pages from our test set. Examine Figure 5.39. This figure shows the distribution of targets found in our search vs. the number of starting points. What we see is that each search can find one and only one target—a veritable needle-in-the-haystack. Contrast this against Figure 5.5 in which on average each of the searches finds a touch over two targets, with a max of 9 targets found on one search while searching smaller search-cones (mean size 1078 for this task vs. 407 pages for 544)!

		Δ0.05	0.95	Δ0.09	0.95	Δ	0.95
	Machine	Discount	Significance	Discount	Significance	First Target	Significance
Gold Depth	Random	56.1(152)	yes	8.53(23.2)	yes	-186(-506)	yes
	Breadth	71.9(195)	yes	10.9(29.7)	yes	-239(-649)	yes
	Depth	45.5(151)	yes	6.91(22.9)	yes	-151(-501)	yes
Gold Discount	Random	56.1(152)	yes	8.53(23.2)	yes	-186(-506)	yes
	Breadth	71.9(195)	yes	10.9(29.7)	yes	-239(-649)	yes
	Depth	45.5(151)	yes	6.91(22.9)	yes	-151(-501)	yes
SVM 1	Random	15.7(65.7)	yes	2.38(9.98)	yes	-52(-218)	yes
	Breadth	42.6(109)	yes	6.47(16.5)	yes	-142(-361)	yes
	Depth	10.4(64.1)	yes	1.58(9.74)	yes	-34.5(-213)	yes
SVM 2	Random	16.6(66.1)	yes	2.52(10)	yes	-55(-219)	yes
	Breadth	44.3(109)	yes	6.73(16.6)	yes	-147(-362)	yes
	Depth	12.6(64.4)	yes	1.92(9.79)	yes	-42(-214)	yes
SVM 3	Random	-27.1(-57.4)	yes	-4.12(-8.72)	yes	90(191)	yes
	Breadth	-10.7(-14.5)	yes	-1.62(-2.2)	yes	35.5(48.1)	yes
	Depth	-24.7(-59)	yes	-3.75(-8.97)	yes	82(196)	yes
SVM 4	Random	-29.5(-59.8)	yes	-4.48(-9.08)	yes	98(198)	yes
	Breadth	-11.7(-16.9)	yes	-1.78(-2.56)	yes	39(55.9)	yes
	Depth	-27.2(-61.4)	yes	-4.14(-9.33)	yes	90.5(204)	yes
SVM 5	Random	-24.8(-40.4)	yes	-3.77(-6.14)	yes	82.5(134)	yes
	Breadth	-9.78(2.47)	yes	-1.49(0.38)	yes	32.5(-8.3)	yes
	Depth	-22.9(-42.1)	yes	-3.48(-6.39)	yes	76(140)	yes
SVM 6	Random	-25.9(-43.2)	yes	-3.94(-6.55)	yes	86(143)	yes
	Breadth	-10.5(-0.254)	yes	-1.6(-0.0345)	yes	35(0.754)	yes
	Depth	-24.1(-44.8)	yes	-3.66(-6.8)	yes	80(149)	yes

Table 5.16: TREC Task 541: 3 away to depth 3. Median (and mean) differences in log discounted reward with 0.95 significance (Wilcoxon Sign-Rank).

Table 5.16 summarizes the results from this experiment. Since each of the individual runs only find one run, the three columns (0.5 discounted reward, 0.9 discounted reward, and time to first target) convey identical information only formulated differently. We show all three only for comparison against other experiments. Machines one and two are the only trained machines that outperform the naive spiders. The other machines (trained like those for Task 544) do not perform well. For the well performing machines, note that, as for the same experiment in Task 544, that the depth-first search performs best, followed by random, and then by breadth-first search. We will expect this to reverse as the search-depth increases. Notice that the gold standard searches (depth and discounted reward) perform identically. This is expected with only one target to be found.

Next we increase our search depth by one. Now, as we can see in Figure 5.40, there are a few more targets to be found, but the mean is still very close to 1 (only 8 of 51 starts finds more than one target). Examining Table 5.17 we can see how our performance margin increases (for the good spiders) as the depth increases, other than for breadth first search whose absolute performance doesn't change with increased

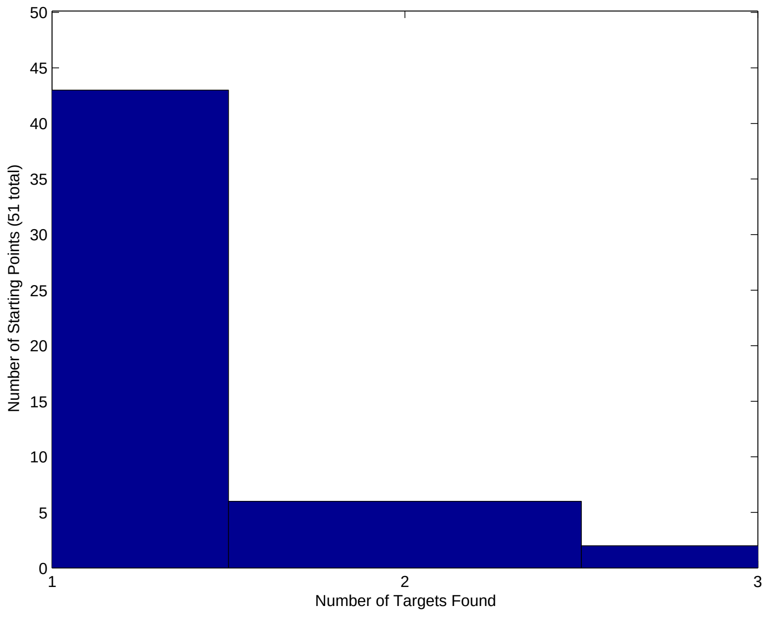


Figure 5.40: TREC Task 541: Starting at pages 3 away (51 starts) to depth 4 searching to exhaustion. Distribution of targets found. Mean number of targets: 1.2, median: 1. Mean number of pages in search-cone: 1999.7, median: 825.

	Machine	Δ0.05 Discount	0.95 Significance	Δ0.09 Discount	0.95 Significance	Δ First Target	0.95 Significance
Gold Depth	Random	81.3(245)	yes	12.4(37.3)	yes	-270(-814)	yes
	Breadth	71.5(194)	yes	10.9(29.5)	yes	-238(-644)	yes
	Depth	85.5(274)	yes	13(41.6)	yes	-284(-909)	yes
Gold Discount	Random	81.3(245)	yes	12.4(37.3)	yes	-270(-815)	yes
	Breadth	71.5(194)	yes	10.9(29.5)	yes	-238(-644)	yes
	Depth	85.5(274)	yes	13(41.7)	yes	-284(-910)	yes
SVM 1	Random	29.2(130)	yes	4.44(19.8)	yes	-97(-433)	yes
	Breadth	31.3(79.2)	yes	4.76(12)	yes	-104(-263)	yes
	Depth	40.9(159)	yes	6.22(24.2)	yes	-136(-528)	yes
SVM 2	Random	31.2(130)	yes	4.72(19.8)	yes	-104(-432)	yes
	Breadth	34.6(79)	yes	5.26(12)	yes	-115(-262)	yes
	Depth	42.7(159)	yes	6.5(24.1)	yes	-142(-527)	yes
SVM 3	Random	-74.4(-113)	yes	-11.2(-17.2)	yes	248(377)	yes
	Breadth	-109(-165)	yes	-16.3(-25)	yes	362(547)	yes
	Depth	-57.5(-84.8)	yes	-8.74(-12.9)	yes	191(282)	yes
SVM 4	Random	-82.7(-134)	yes	-12.5(-20.3)	yes	275(444)	yes
	Breadth	-116(-185)	yes	-17.3(-28.1)	yes	385(614)	yes
	Depth	-67.9(-105)	yes	-10.3(-15.9)	yes	226(349)	yes
SVM 5	Random	-80.2(-125)	yes	-12.2(-19)	yes	266(416)	yes
	Breadth	-111(-177)	yes	-16.8(-26.8)	yes	370(586)	yes
	Depth	-64.7(-96.7)	yes	-9.84(-14.7)	yes	215(321)	yes
SVM 6	Random	-82.2(-128)	yes	-12.5(-19.5)	yes	274(427)	yes
	Breadth	-114(-180)	yes	-16.9(-27.3)	yes	379(597)	yes
	Depth	-66.8(-99.8)	yes	-10.2(-15.1)	yes	222(331)	yes

Table 5.17: TREC Task 541: 3 away to depth 4. Median (and mean) differences in log discounted reward with 0.95 significance (Wilcoxon Sign-Rank).

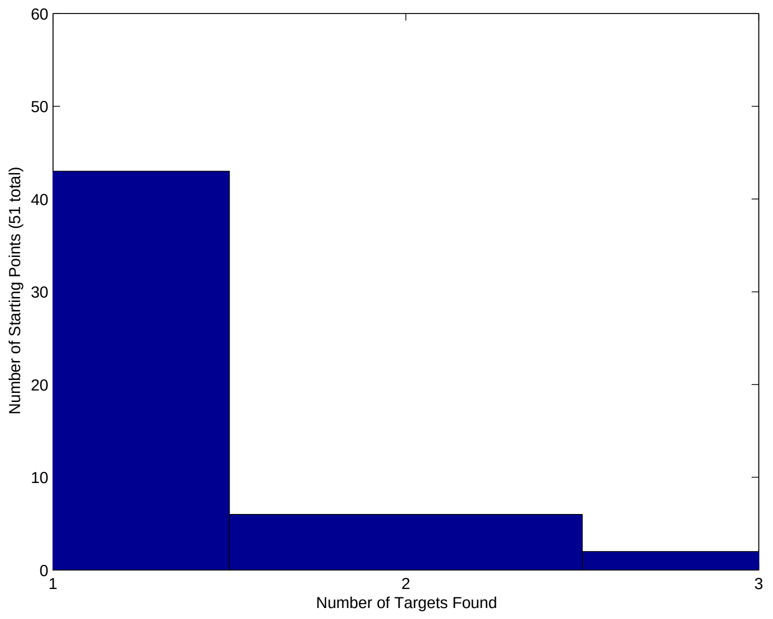


Figure 5.41: TREC Task 541: Starting at pages 3 away (51 starts) to depth 5 searching to exhaustion. Distribution of targets found. This is identical to the distribution in Figure 5.40—no new targets were found. Mean number of targets: 1.2, median: 1. Mean number of pages in search-cone: 3161.8, median: 1944.

depth. As before machines 3-6 all perform poorly, however our specially trained machines (1 and 2) are performing well. Notice that the balance has started to shift towards favoring the breadth-first search among the naive searches.

In our final experiment at height three, we increase our search depth to five. Looking at Figure 5.41, we see that no new targets were found in searching to depth five, despite almost doubling the average search-cone size. These results are especially favorable to the breadth first search as it will find the initial target(s) in a fixed number of actions, not wasting any time on the empty half of the search space. Despite that, our good SVMs continue to outperform all of the naive searches albeit with a small (but significant) margin over the breadth-first search. Notice that our expected ordering among the naive searches has returned with the random search lying between the breadth- and depth-first searches.

Figure 5.42 consolidates the above results for the height-three searches. Notice the distinctive distribution shapes for each of the search strategies. The two “bad” learned spiders really display how badly they performed. It almost looks as if they are trying *not* to find targets. By the way, I did try inverting the preference function

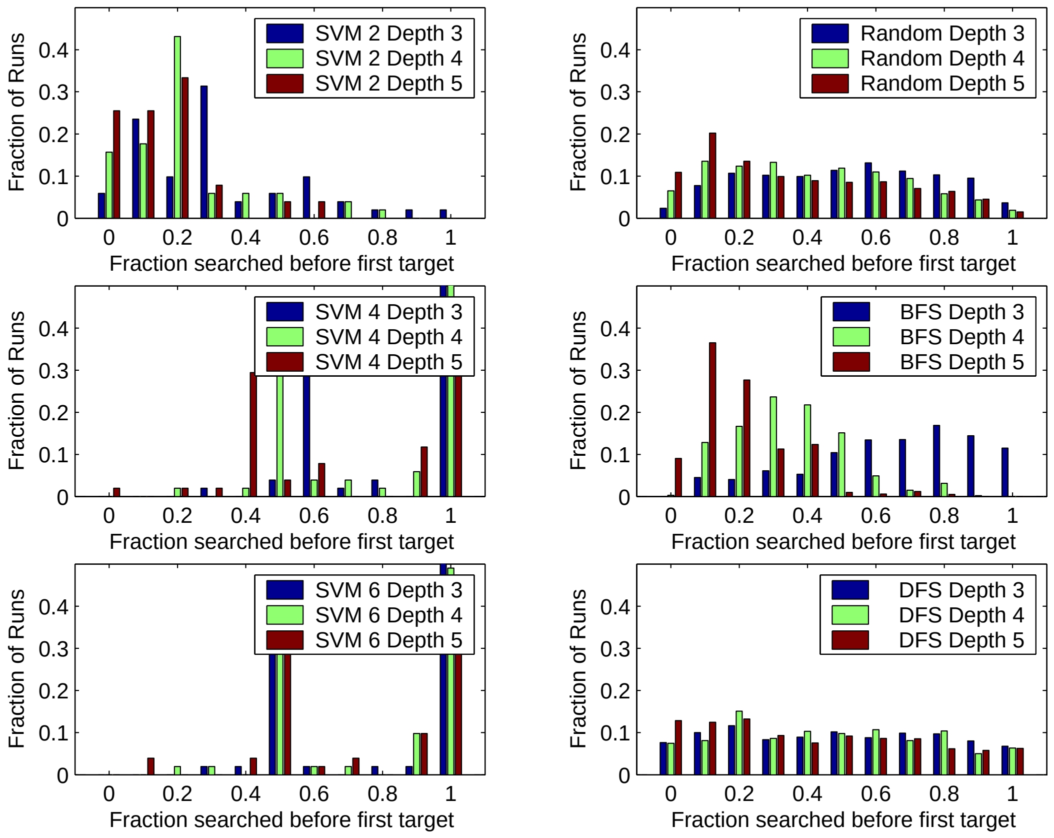


Figure 5.42: TREC Task 541: Starting from pages 3 away (51 starts) from targets and searching to depths 3, 4, and 5. Distribution of page retrievals to find first target divided by total pages in search cone.

		Δ0.05	0.95	Δ0.09	0.95	Δ	0.95
	Machine	Discount	Significance	Discount	Significance	First Target	Significance
Gold Depth	Random	128(385)	yes	19.4(58.5)	yes	-424(-1280)	yes
	Breadth	71.2(194)	yes	10.9(29.5)	yes	-236(-645)	yes
	Depth	149(401)	yes	22.7(61)	yes	-496(-1330)	yes
Gold Discount	Random	128(385)	yes	19.5(58.5)	yes	-424(-1280)	yes
	Breadth	71.2(194)	yes	10.9(29.5)	yes	-236(-645)	yes
	Depth	149(401)	yes	22.7(61)	yes	-496(-1330)	yes
SVM 1	Random	49.4(192)	yes	7.5(29.2)	yes	-164(-637)	yes
	Breadth	14.6(0.964)	yes	2.22(0.158)	yes	-48.5(-3.32)	yes
	Depth	59.6(208)	yes	9.13(31.7)	yes	-198(-692)	yes
SVM 2	Random	57.6(197)	yes	8.76(29.9)	yes	-192(-654)	yes
	Breadth	16(6.13)	yes	2.43(0.942)	yes	-53(-20.5)	yes
	Depth	68.2(213)	yes	10.4(32.4)	yes	-226(-709)	yes
SVM 3	Random	-134(-177)	yes	-20.3(-26.9)	yes	446(589)	yes
	Breadth	-293(-368)	yes	-44.2(-55.9)	yes	972(1220)	yes
	Depth	-108(-161)	yes	-16.3(-24.4)	yes	358(534)	yes
SVM 4	Random	-154(-177)	yes	-23.3(-26.8)	yes	510(587)	yes
	Breadth	-296(-368)	yes	-44.8(-55.8)	yes	982(1220)	yes
	Depth	-113(-160)	yes	-17.2(-24.3)	yes	376(532)	yes
SVM 5	Random	-155(-219)	yes	-23.5(-33.3)	yes	514(728)	yes
	Breadth	-313(-410)	yes	-47.2(-62.3)	yes	1040(1360)	yes
	Depth	-122(-203)	yes	-18.6(-30.8)	yes	406(673)	yes
SVM 6	Random	-168(-244)	yes	-25.3(-37)	yes	558(810)	yes
	Breadth	-328(-435)	yes	-49.5(-66)	yes	1090(1440)	yes
	Depth	-137(-227)	yes	-20.8(-34.5)	yes	456(755)	yes

Table 5.18: TREC Task 541: 3 away to depth 5. Median (and mean) differences in log discounted reward with 0.95 significance (Wilcoxon Sign-Rank).

for one of these to see if it was a negative discriminator and it didn't work, it just made a different set of bad choices.

In our next set of short-range experiments, we increase the starting height by one. Now we start our searches at a height of 4 from targets. In these experiments, we don't bother testing the ineffective spiders (SVM 3-6 in the height-3 trials). In Figure 5.43 from our first experiment at height-4, we see the distribution of targets found vs runs for a depth 4 search from the 100 starting pages. 21 of the 100 starting pages find additional targets. This gives us a slightly richer search space, and also will benefit the depth-first search, as all these additional targets are all at depth-4 where the DFS will focus its efforts.

Examining Table 5.16, we see that indeed the depth-first search does perform well compared to the random and breadth-first searches. However, our good, SVM-trained spiders consistently outperform all of the naive searches. The margin against the depth-first search is small in median, but still significant. The margin is, in expectation, strongly in favor of the SVM based searches.

We conclude our height-4 experiments searching to depth 5. In Figure 5.43 we

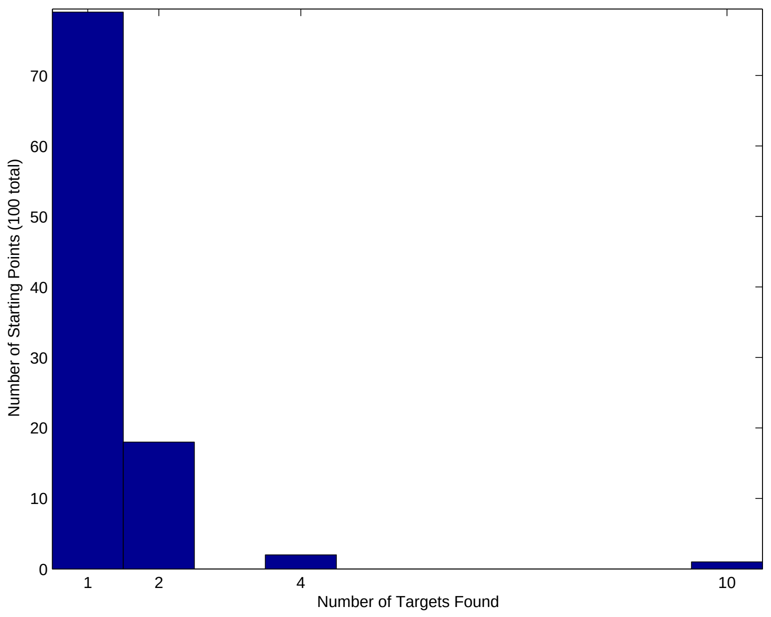


Figure 5.43: TREC Task 541: Starting at pages 4 away (100 starts) to depth 4 searching to exhaustion. Distribution of targets found. Mean number of targets: 1.33, median: 1. Mean number of pages in search-cone: 1201, median: 664.

	Machine	Δ0.05 Discount	0.95 Significance	Δ0.09 Discount	0.95 Significance	Δ First Target	0.95 Significance
Gold Depth	Random	92.1(201)	yes	14.1(30.6)	yes	-306(-667)	yes
	Breadth	132(244)	yes	20.2(37.1)	yes	-440(-810)	yes
	Depth	62.6(199)	yes	9.52(30.3)	yes	-208(-661)	yes
Gold Discount	Random	90(199)	yes	13.7(30.3)	yes	-299(-662)	yes
	Breadth	127(242)	yes	19.4(36.9)	yes	-423(-805)	yes
	Depth	61.3(197)	yes	9.36(30)	yes	-204(-656)	yes
SVM 1	Random	7.22(31.2)	yes	1.12(4.74)	yes	-24(-104)	yes
	Breadth	27.4(74.3)	yes	4.16(11.3)	yes	-91(-247)	yes
	Depth	2.71(29.3)	yes	0.458(4.46)	yes	-9(-97.4)	yes
SVM 2	Random	8.73(37.8)	yes	1.34(5.75)	yes	-29(-125)	yes
	Breadth	33.4(80.9)	yes	5.09(12.3)	yes	-111(-269)	yes
	Depth	4.21(35.9)	yes	0.641(5.47)	yes	-14(-119)	yes

Table 5.19: TREC Task 541: 4 away to depth 4. Median (and mean) differences in log discounted reward with 0.95 significance (Wilcoxon Sign-Rank).

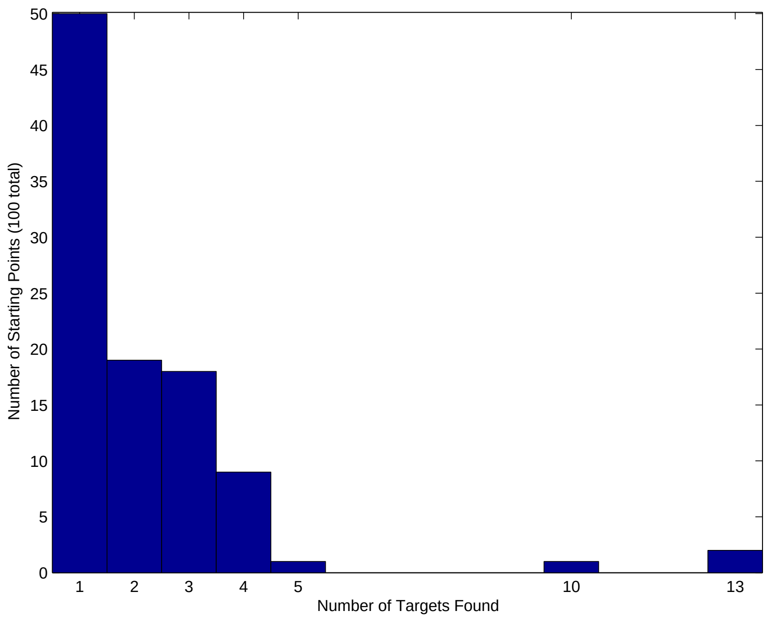


Figure 5.44: TREC Task 541: Starting at pages 4 away (100 starts) to depth 5 searching to exhaustion. Distribution of targets found. Mean number of targets: 2.19, median: 1.5. Mean number of pages in search-cone: 3476, median: 1031.

	Machine	Δ0.05 Discount	0.95 Significance	Δ0.09 Discount	0.95 Significance	Δ First Target	0.95 Significance
Gold Depth	Random	126(316)	yes	19.2(48.2)	yes	-420(-1.05e+03)	yes
	Breadth	132(244)	yes	20.2(37.2)	yes	-440(-810)	yes
	Depth	107(322)	yes	16.3(49)	yes	-354(-1.07e+03)	yes
Gold Discount	Random	126(315)	yes	19.1(47.9)	yes	-417(-1.05e+03)	yes
	Breadth	127(242)	yes	19.4(36.9)	yes	-423(-805)	yes
	Depth	106(320)	yes	16.1(48.8)	yes	-352(-1.06e+03)	yes
SVM 1	Random	10.2(11)	yes	1.56(1.68)	yes	-34(-36.5)	yes
	Breadth	2.11(-61.3)	yes	0.503(-9.3)	yes	-7(204)	yes
	Depth	12.3(16.7)	yes	1.97(2.54)	yes	-41(-55.3)	yes
SVM 2	Random	13.8(23.3)	yes	2.1(3.55)	yes	-46(-77.6)	yes
	Breadth	3.3(-49)	no	0.611(-7.43)	no	-10.5(163)	no
	Depth	12.3(29)	yes	1.94(4.42)	yes	-41(-96.4)	yes

Table 5.20: TREC Task 541: 4 away to depth 5. Median (and mean) differences in log discounted reward with 0.95 significance (Wilcoxon Sign-Rank).

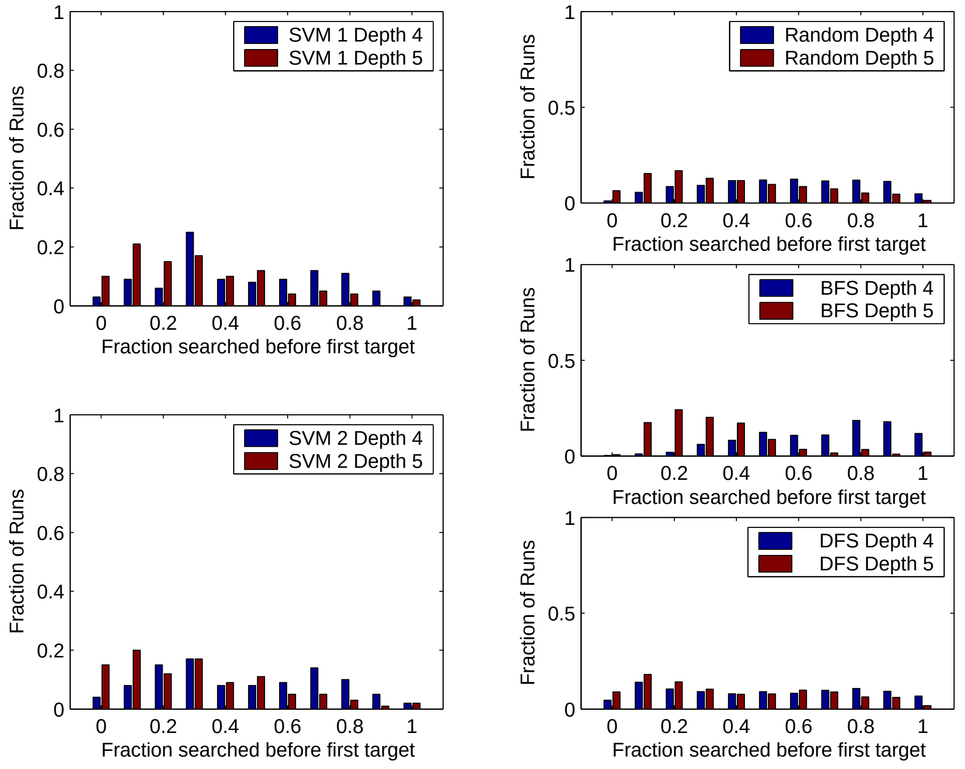


Figure 5.45: TREC Task 541: Starting from pages 4 away (100 starts) from targets and searching to depths 4 and 5. Distribution of page retrievals to find first target divided by total pages in search cone.

see the distribution targets found vs. starting points. A full 50% of the search-cones contain additional targets with the mean number of targets found at just over 2 per search-cone. Table 5.20 summarizes our spidering results. We see that the breadth-first search is now competitive with our SVM based spiders, but still loses in median (without significance).

Figure 5.45 summarizes our results from the height-4 experiments. As usual we get the distinctive distribution shapes for each of the search strategies.

Like in our short-range exploration experiments for Task 544, these results show that it is possible to train a smart spider that can perform better than the naive spiders. They also confirm, that in the presence of knowledge of nearby targets, and lacking a trained spider, the breadth-first search will perform well. If we can bound the search depth to the level at which targets exist, then the depth-first search can

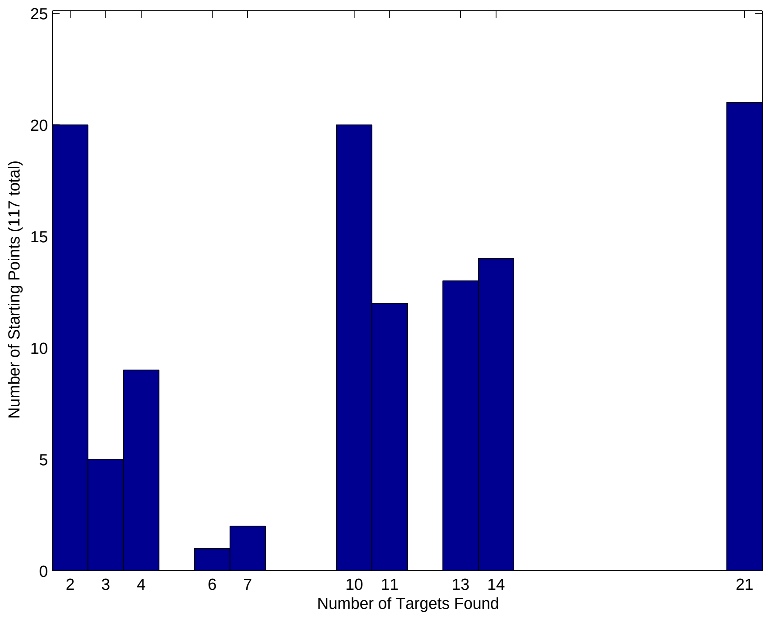


Figure 5.46: TREC Task 541: Starting at targets (117 starts) to depth 4 searching to exhaustion. Distribution of targets found. Mean number of targets: 10.7, median: 11. Mean number of pages in search-cone: 171.5, median: 47.

be effective.

In our next set of experiments we test the exploitation capabilities of our spiders. In other words, we will start our searches from the targets themselves and search out to fixed depths. We select for our targets for our testing set from those that aren't used in training our spiders and from which other targets are findable in a search-cone of depth 4—there is no sense in trying to search an empty cone. In the end we have 117 targets for our testing set. 73 of these targets are weather reports from the NASA Kennedy Space Center site.

In our first exploitation experiment we search to depth-4. The distribution of targets found vs. search cones (start pages) is shown in Figure 5.46. We can see that their are plenty of targets to be found, however keep in mind that all but 44 of the starting pages are in the same domain and are highly structured. All of the Kennedy Space Center pages have targets very nearby which our trained spiders seem to miss. This gives us a huge disadvantage in the median results. Examine our results for the gold standard spiders in Table 5.21. Notice that we are just barely beating the

	Machine	Δ0.05 Discount	0.95 Significance	Δ0.09 Discount	0.95 Significance	Δ First Target	0.95 Significance
Gold Depth	Random	0.348(10.6)	yes	0.272(1.78)	yes	-1(-34.9)	yes
	Breadth	0.316(7.19)	yes	0.162(1.19)	yes	-1(-23.6)	yes
	Depth	0.365(16.5)	yes	0.397(2.69)	yes	-1(-54.5)	yes
Gold Discount	Random	0.348(10.3)	yes	0.272(1.74)	yes	-1(-34)	yes
	Breadth	0.316(6.91)	yes	0.162(1.15)	yes	-1(-22.7)	yes
	Depth	0.358(16.2)	yes	0.397(2.65)	yes	-1(-53.6)	yes
SVM 1	Random	-1.2(6.04)	yes	-0.0402(1.01)	yes	4(-19.9)	yes
	Breadth	-1.2(2.64)	yes	-0.166(0.42)	yes	4(-8.65)	yes
	Depth	-1.24(11.9)	no	0.0209(1.92)	yes	4(-39.6)	no
SVM 2	Random	-1.19(6.23)	yes	-0.0402(1.04)	yes	4(-20.5)	yes
	Breadth	-1.2(2.83)	yes	-0.164(0.446)	yes	4(-9.29)	yes
	Depth	-1.24(12.1)	no	0.0209(1.95)	yes	4(-40.2)	no

Table 5.21: TREC Task 541: From targets to depth-4. Median (and mean) differences in log discounted reward with 0.95 significance (Wilcoxon Sign-Rank).

	Machine	Δ0.05 Discount	0.95 Significance	Δ0.09 Discount	0.95 Significance	Δ First Target	0.95 Significance
Gold Depth	Random	0.537(91.5)	yes	0.393(14.1)	yes	-1(-304)	yes
	Breadth	0.48(99.3)	yes	0.251(15.2)	yes	-1(-329)	yes
	Depth	0.71(70.5)	yes	0.614(11)	yes	-2(-234)	yes
Gold Discount	Random	0.537(91.5)	yes	0.394(14.1)	yes	-1(-303)	yes
	Breadth	0.48(99.2)	yes	0.251(15.2)	yes	-1(-329)	yes
	Depth	0.659(70.5)	yes	0.61(11)	yes	-2(-234)	yes
SVM 1	Random	-1.19(34.6)	yes	-0.00742(5.35)	yes	4(-115)	yes
	Breadth	-1.18(42.4)	yes	-0.152(6.46)	yes	4(-141)	yes
	Depth	-1.25(13.6)	no	0.0358(2.19)	yes	4(-45.1)	no
SVM 2	Random	-1.19(34.8)	yes	-0.00811(5.37)	yes	4(-115)	yes
	Breadth	-1.16(42.5)	yes	-0.149(6.47)	yes	4(-141)	yes
	Depth	-1.25(13.8)	no	0.0376(2.2)	yes	4(-45.5)	no

Table 5.22: TREC Task 541: From targets to depth 5. Median (and mean) differences in log discounted reward with 0.95 significance (Wilcoxon Sign-Rank).

naive spiders in median. Looking at the "Δ First Target" column, the gold spiders are finding the first target in median only one page retrieval earlier than the naive spiders. However, if we look at the mean scores, we see that our gold spiders are performing much better on the other tasks, despite the median. This pattern continues to our SVM based spiders.

Although the SVM spiders are losing in median for the pages-to-first target score and the $\alpha = 0.9$ discounted reward scores in median (not always significantly), they are still winning when considering the mean. An interesting phenomenon, showing just how close the results are, is that when we look at our discounted reward calculated with $\alpha = 0.9$ we actually have a slight reversal when we consider the depth-first search results. Notice also that while the SVM depth-first search loses in median without significance at discount $\alpha = 0.5$ it wins *with* significance at 0.9.

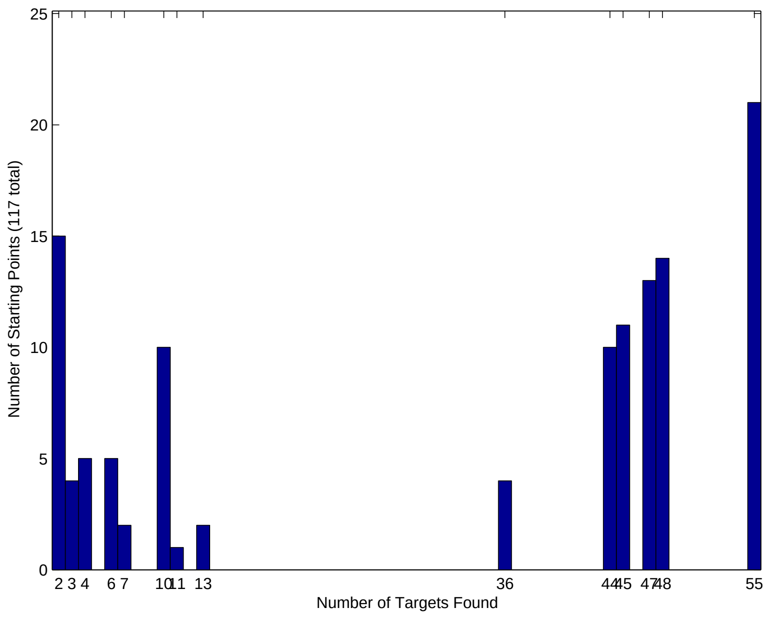


Figure 5.47: TREC Task 541: Starting at targets (117 starts) to depth 5 searching to exhaustion. Distribution of targets found. Mean number of targets: 32.1, median: 45. Mean number of pages in search-cone: 675.8, median: 491.

In our final exploitation experiment against the Weather Instrument task we increase our search depth from the targets to 5. The distribution of targets found vs. starting pages is shown in Figure 5.47. We see that the expected number of targets for an arbitrary run has almost tripled, as has the size of the search-cone. This doesn't result in much of a change in the results however. Unlike with the exploration phase, there are very close targets to be found, and our trained spiders are missing the first close one while the naive searches aren't. Despite this impact on our median, the mean scores have dramatically increased in favor of the trained spiders. The phenomenon of the depth-first search winning with discounted reward of $\alpha = 0.9$ continues from the depth-4 results. Remember that an α value of 0.5 tends to strongly favor early results while a high discount factor like 0.9 favors sustained performance. Figure 5.48 summarizes our exploitation results showing normalized page-retrievals to first target.

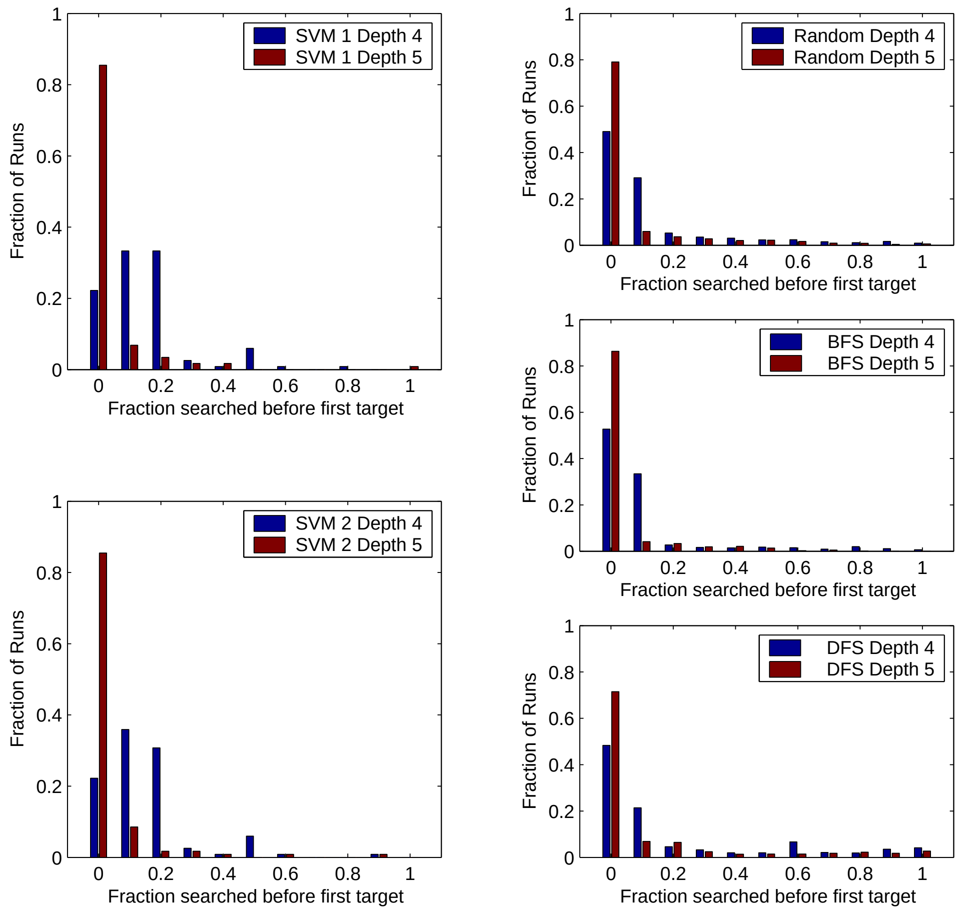


Figure 5.48: TREC Task 541: Starting from targets (117 starts) and searching to depths 4 and 5. Distribution of page retrievals to find first target divided by total pages in search cone.

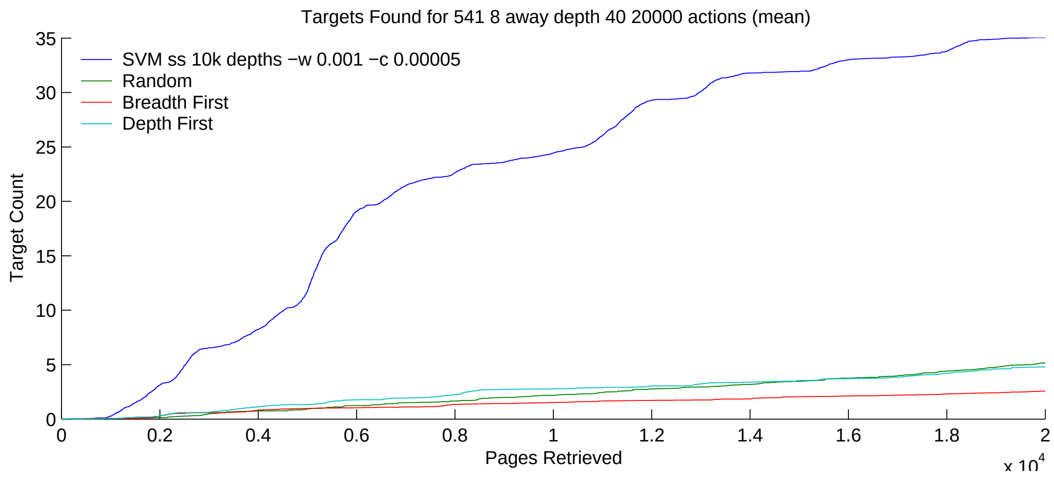


Figure 5.49: TREC Task 541: Starting at 100 pages 8 away to depth 40 for 20,000 page retrievals. Mean performance of spiders.

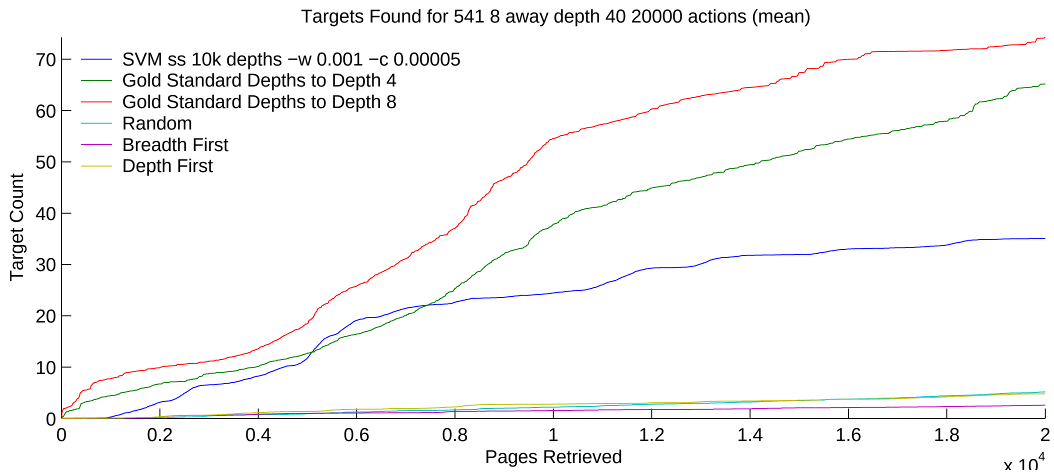


Figure 5.50: TREC Task 541: Starting at 100 pages 8 away to depth 40 for 20,000 page retrievals. Mean performance of spiders comparing the SVM spider and the Gold Standard spiders.

Long-range Evaluation

In the previous discussion we showed that our spider, carefully trained from an over-sampled dataset built from the search-cone from 40 targets was able to perform better than or competitively with the naive spiders, especially when considering the mean results. This isn't all that surprising as the short-range tests are taking place in the same region of the web-graph as the testing dataset and we did select a machine that performed well against the testing set. (There isn't complete identity as of course the child search-cones only overlap the parent search-cone of the testing dataset).

The next question is: Can the learned machine guide a spider navigation from a greater distance than that of the region it was trained on? To test this, we ran an experiment similar to that shown in Figure 5.31. We started from 100 randomly selected pages whose closest targets were 8 hops away on the shortest path. We then ran the learned spider as well as the naive spiders from each of these initial pages for $20,000$ page retrievals. Figure 5.49 shows the results.

While it is clear that our trained spider dominates the naive spiders in the long range search, how much better could we hope to do? Examining Figure 5.50 we see the performance of our SVM spider vs. two Gold Standard spiders. One is only using the information the SVM spider is directly trained to recognize: depths up to a height of four hops above targets. The second is using perfect knowledge of the heights up to a height of eight hops above the targets. We see that while the Gold Standard spiders get off the starting mark much faster than the SVM spider, in the long haul we are doing pretty well, about 50 percent of the best we could hope to do. In fact, at one point (hitting a pocket of closely connected targets) we actually surpass the performance of the Depth-4 Gold Standard spider! Remember the Depth-4 Gold Standard spider has no leverage at heights greater than four hops from the target, merely performing a random search until they finally meander close enough to some target to pick up the scent.

Summary

In the above results we examined the performance of our SVM spiders on a new TREC task with different structural properties in the web-graph as well as lower quality labeled answer pages. These differences made our technique for building spiders for

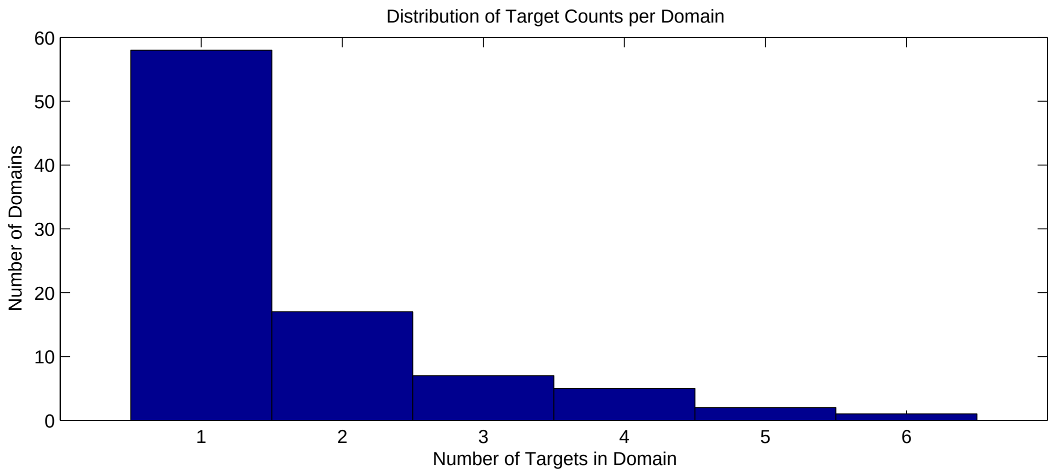


Figure 5.51: Distribution of the number of targets per domain

the 544 task not work for the 541 task. However by using additional seed information, and some fancier sampling techniques, we were still able to learn a machine able to spider successfully in this task. We even got exceptional results in our long-range search, sometimes even surpassing the performance of our gold-standard spiders.

5.2.2 Searching for Frog Habitats

In the previous section we examined our spidering performance on a challenging task, that of searching for pages about weather instruments. In this section we continue our discussion of results with challenging tasks. In particular we present our spidering results for TREC task 519:

- **Description:** Find documents that describe the habitat of frogs.
- **Narrative:** A relevant document will identify the natural habitat of any type of frog. A frog's diet is not relevant.

Examining Figure 5.51 we see that their are no exceptionally large domains such as we had in Task 541 (Figure 5.2.1). The majority of domains only contained one target and no domains had greater than six targets. Contrast this with Task 544 (Figure 5.1) which had several domains containing many targets.

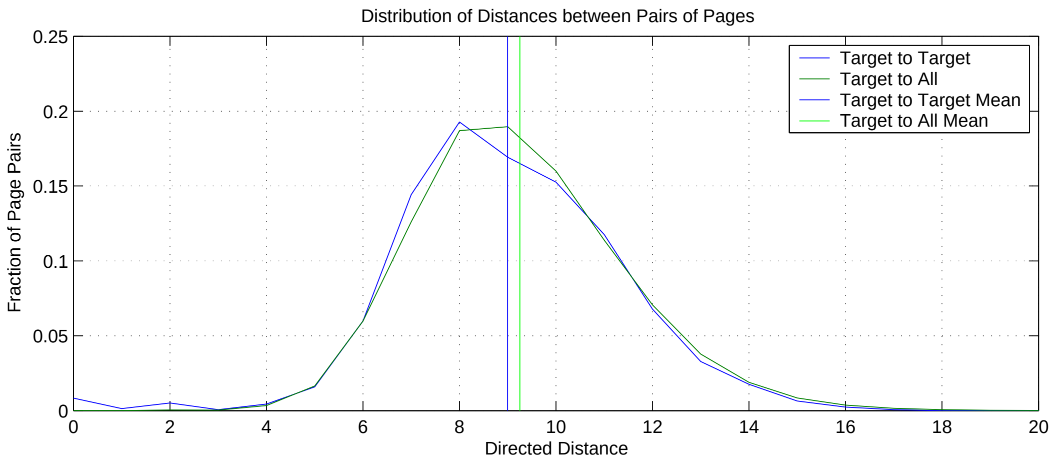


Figure 5.52: Distributions of the Diameter of the Web-graph as observed from the TREC Task 519. The mean target-to-target distance is 9.00 and the mean target-to-all distance is 9.25.

Also, looking at the diameter distribution for this task we see that the target-to-target expected diameter is very close to the target-to-all pages expected diameter. In our previous two tasks, the expected diameter for the target-to-target distribution was about one whole link less than the target-to-all pages diameter.

These two measurements indicate that the 519 tasks are less focused than our earlier tasks. In other words they don't seem to be as strong of a thematically unified cluster (TUC). This will make learning an effective spider more difficult. Also, like task 541, there are very few high quality targets in the target group (7 of 149). The other targets are diffuse.

Building the Machines

As was the case for our experiments with Task 541, building effective SVM based spiders was a challenge for this task. As before, we filtered our dictionaries by using stopping and stemming. We then reduced our final dictionary for the task down to $10,000$ terms picking the words with the highest information gain.

In our first tests we arbitrarily selected 15 seed targets

`http://205.130.176.100/msschool/wet04.html`

```
Machine 1:  1^0^1^seed_15_10000^seed_15.dp_svmlight.svm-z r -w 0.000001 -c 0.0001^seed_15.widx
Machine 2:  1^1^1^seed_15_10000^seed_15.dp_svmlight.svm-z r -w 0.000001 -c 0.0001^seed_15.widx
```

Table 5.23: Support Vector Machine trained machines used for TREC task 519 in the short-range experiments. Both machines use the same SVM machine. Machine 2 uses the augmented delta scores when evaluating links during spidering.

```
http://bcn.boulder.co.us/campuspress/april41996/Presshome.html
http://www.peoriaud.k12.az.us/docs/mediaminder/T.html
http://www.r1.fws.gov/4deaa/chap4.html
http://www.s-t.com/daily/06-96/06-23-96/m01lo179.htm
http://www.snowcrest.net/klewis/feather.htm
http://www.uwsp.edu/stuorg/pointer/issue24/frogs.htm
http://www34.web.binc.net/ncoldlinks.html
http://gto.ncsa.uiuc.edu/pingleto/herps/frogpix.html
http://orpheus.ucsd.edu/wmrs/minutes3postsnep96.html
http://tbone.biol.sc.edu/~dean/carl/Ecology.html
http://www.bonsaiweb.com/ponds/show/show.html
http://www.defenders.org/esa-12.html
http://www.enviroworld.com/September96/091096.html
http://www.gallaudet.edu/~kartweb/earthventure.html
http://www.katy.isd.tenet.edu/se/eslegg.html
http://www.livingplanet.org/global200/nigeria/bio.htm
```

and trawled the parent graph to height-4 finding seven more targets in the process. This was our training set. We then calculated our estimated discounted rewards for this training set. We our depth and discounted reward training values, we trained linear kernel support vector machines using several different parameterizations. From these we picked one machine that seemed to have good training results. We then ran the short-term experiments described below using this machine with and without the augmented delta technique. These machines are shown in Table 5.23.

We also used the augmented delta machine as one of our machines for the 5 away long range tests. Not being happy with our short-range or long-range results, we selected another sampling of 16 target pages and trawled them to height-4 finding 10 more targets in the process. The initial pages were

```
http://www.projo.com/special/crans1.htm
```

```
http://bermuda.wwa.com/Guest/feedback.html
http://www.rain.org/~sals/multi.html
http://www.smh.com.au/daily/content/1996/Nov/11/col8/961111-column8.html
http://www.sunshine.net/www/700/sn0737/
http://www.wwfcanada.org/facts/amphib.html
http://acorn-group.com/contents.htm
http://gto.ncsa.uiuc.edu/pingleto/herps/greentrees.html
http://people.delphi.com/grinnellj/at2.html
http://tbone.biol.sc.edu/~dean/carl/Journal-of-experimental-biology.html
http://www.ci.pacifica.ca.us/NATURAL/SNAKE/snake.html
http://www.defenders.org/pfroga.html
http://www.evergreen.ca/nslgvanp6.html
http://www.great-lakes.net/envt/exotic/loosestf.html
http://www.kortright.org/wildfac.html
http://www.mccutchen.com/env/wet.htm
```

Again we computed our estimated discounted rewards and trained SVM spiders with various parameterizations. One of these machines is tested in our long range results at distance 8 away below.

As in our previous tasks, due to the vast number of pages in the training set at a height of 3 or 4 above the targets, we found the SVM machines had a tendency to simply learn to guess the mean and still score 90 plus percent on our tests. To overcome this we sampled, with replacement (as we did for Task 541), 900 pages from each depth from this previous task and then built SVM machines for these. The best of breed from these machines were tested in the height-5 and height-8 long-range tests. Unlike for Task 541, as we'll see in the discussion below, this wasn't enough to produce universally effective spiders.

Short-range Evaluation

As we did for the 544 and 541 tasks above, we first investigate the short-range search capabilities of our spiders. This consists of two phases: the exploration phase in which we try to find targets; and the exploitation phase where we start from targets, trying to find additional targets.

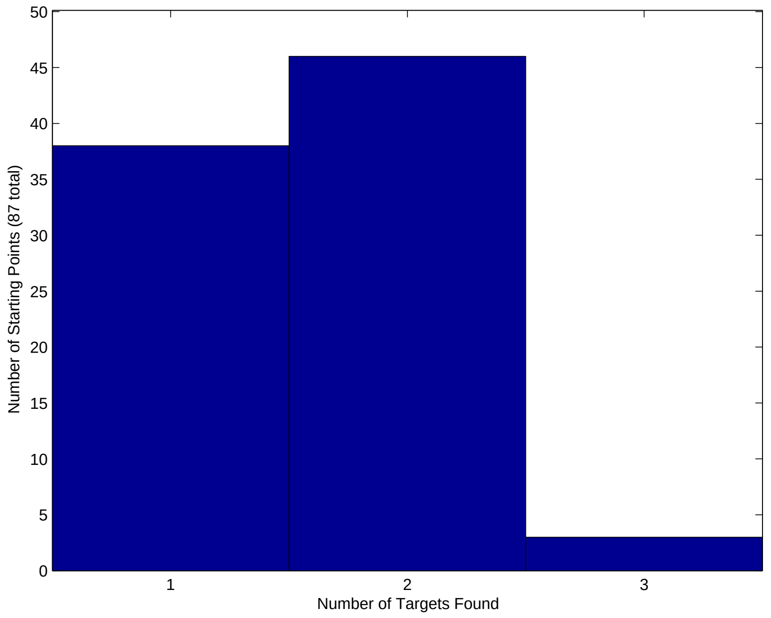


Figure 5.53: TREC Task 519: Starting at pages 3 away (87 starts) to depth 3 searching to exhaustion. Distribution of targets found. Mean number of targets: 1.6, median: 2. Mean number of pages in search-cone: 297, median: 260.

	Machine	Δ0.05 Discount	0.95 Significance	Δ0.09 Discount	0.95 Significance	Δ First Target	0.95 Significance
Gold Depth	Random	43.1(48.8)	yes	6.61(7.51)	yes	-143(-162)	yes
	Breadth	51.9(56.8)	yes	7.97(8.68)	yes	-172(-188)	yes
	Depth	58.3(55.5)	yes	8.92(8.48)	yes	-193(-184)	yes
Gold Discount	Random	42.6(48.7)	yes	6.59(7.48)	yes	-142(-161)	yes
	Breadth	51.9(56.6)	yes	7.96(8.65)	yes	-172(-188)	yes
	Depth	58.3(55.4)	yes	8.92(8.46)	yes	-193(-184)	yes
SVM 1	Random	17.6(16.8)	yes	2.81(2.64)	yes	-58(-55.8)	yes
	Breadth	24.5(24.8)	yes	3.83(3.81)	yes	-81(-82.2)	yes
	Depth	42.6(23.5)	yes	6.51(3.61)	yes	-142(-78)	yes
SVM 2	Random	20.1(18.5)	yes	3.11(2.89)	yes	-66.5(-61.3)	yes
	Breadth	25.7(26.4)	yes	3.94(4.05)	yes	-85.5(-87.8)	yes
	Depth	45.2(25.2)	yes	6.95(3.86)	yes	-150(-83.5)	yes

Table 5.24: TREC Task 519: 3 away to depth 3. Median (and mean) differences in log discounted reward with 0.95 significance (Wilcoxon Sign-Rank).

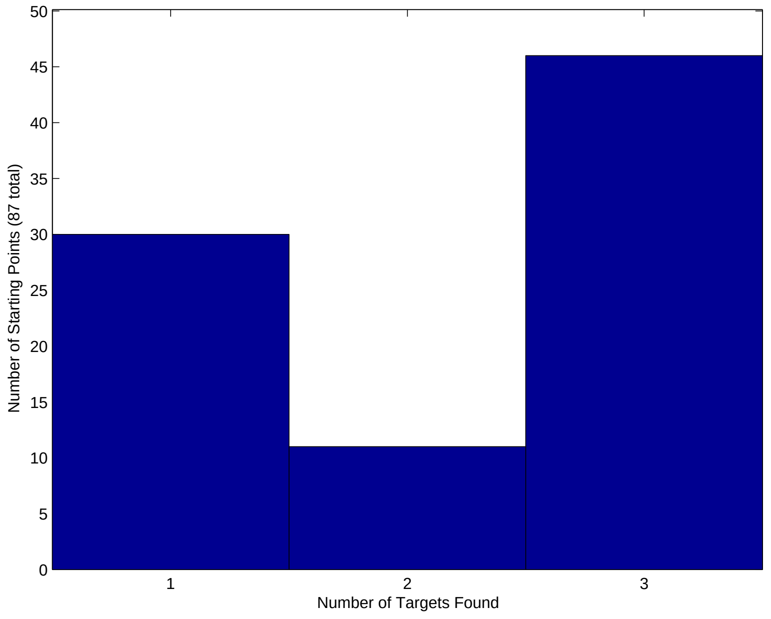


Figure 5.54: TREC Task 519: Starting at pages 3 away (87 starts) to depth 4 searching to exhaustion. Distribution of targets found. Mean number of targets: 2.2, median: 3. Mean number of pages in search-cone: 688, median: 716.

In our first set experiments with the 519 task, we start at a height of three above the targets and spider to a depth of three. Figure 5.53 shows the distribution of targets found vs. the number of starting points. Most of the search-cones (46 of 87) have two targets and only a few search-cones contain three targets; none have more. Like the 541 task and unlike 544 this is very sparse, and indicates that the target class is not as tightly connected as the 544 class (as also indicated by the high target-to-target diameter as shown in Figure 5.52). Another big difference is that the mean number of pages in the search-cones is very small compared with the other tasks, 297 vs. $1,078$ for 541 and 407 for Task 544.

Looking at Table 5.24 we see that despite these differences, our SVM based spiders are doing well at finding the targets. An interesting phenomenon is that the random search is actually outperforming the other naive searches. In virtually all previous experiments it was bracketed between the depth- and breadth-first searches.

Following our usual pattern, we increase the permitted search depth by one. Now examining Figure 5.54 we can see that most of the search-cones that had contained

	Machine	Δ0.05 Discount	0.95 Significance	Δ0.09 Discount	0.95 Significance	Δ First Target	0.95 Significance
Gold Depth	Random	30.1(49.7)	yes	4.7(7.68)	yes	-100(-165)	yes
	Breadth	51.9(56.8)	yes	8(8.7)	yes	-172(-188)	yes
	Depth	78.4(91.3)	yes	12(14)	yes	-260(-303)	yes
Gold Discount	Random	29.7(49.6)	yes	4.6(7.65)	yes	-98.5(-164)	yes
	Breadth	51.8(56.6)	yes	7.96(8.68)	yes	-172(-188)	yes
	Depth	78.4(91.1)	yes	12(13.9)	yes	-260(-302)	yes
SVM 1	Random	7.53(-0.00747)	yes	1.24(0.0775)	yes	-25(0.161)	yes
	Breadth	22.4(7.04)	yes	3.51(1.11)	yes	-74(-23.3)	yes
	Depth	41.7(41.6)	yes	6.35(6.37)	yes	-138(-138)	yes
SVM 2	Random	8.5(3.79)	yes	1.34(0.654)	yes	-28(-12.4)	yes
	Breadth	23.2(10.8)	yes	3.58(1.68)	yes	-77(-35.9)	yes
	Depth	44.9(45.3)	yes	6.82(6.95)	yes	-149(-151)	yes

Table 5.25: TREC Task 519: 3 away to depth 4. Median (and mean) differences in log discounted reward with 0.95 significance (Wilcoxon Sign-Rank).

	Machine	Δ0.05 Discount	0.95 Significance	Δ0.09 Discount	0.95 Significance	Δ First Target	0.95 Significance
Gold Depth	Random	27(69.5)	yes	4.21(10.7)	yes	-89.5(-230)	yes
	Breadth	51.9(56.8)	yes	8(8.71)	yes	-172(-188)	yes
	Depth	94.2(151)	yes	14.3(23)	yes	-313(-500)	yes
Gold Discount	Random	26.9(69.3)	yes	4.21(10.6)	yes	-89(-230)	yes
	Breadth	51.8(56.6)	yes	7.96(8.68)	yes	-172(-188)	yes
	Depth	94.2(150)	yes	14.3(22.9)	yes	-313(-499)	yes
SVM 1	Random	5.34(4.36)	yes	0.923(0.727)	yes	-18(-14.3)	yes
	Breadth	22.4(-8.33)	yes	3.46(-1.23)	yes	-74(27.7)	yes
	Depth	61.5(85.5)	yes	9.36(13)	yes	-204(-284)	yes
SVM 2	Random	4.61(5.15)	yes	0.802(0.854)	yes	-15(-16.9)	yes
	Breadth	21.8(-7.54)	yes	3.4(-1.11)	yes	-72(25.1)	yes
	Depth	58(86.3)	yes	8.89(13.2)	yes	-192(-287)	yes

Table 5.26: TREC Task 519: 3 away to depth 5. Median (and mean) differences in log discounted reward with 0.95 significance (Wilcoxon Sign-Rank).

two targets now contain three targets and our mean search-cone size increases to 688 pages. In Table 5.25, we see that our SVM based spiders continue to perform well, however they are loosing ground to the random search. They are losing to random search when looking at the mean scores, although the difference in medians is still significant.

As we continue to increase our depth to 5, we notice, as shown in Figure 5.55, that no additional targets were discovered in the expanded search-cones, despite doubling the mean number of pages in the search-cones. Performance-wise, our SVM spiders are continuing to outperform, in median, the naive searches (Table 5.26) with significance. Looking at the naive searches, the tide has started to shift in favor of the breadth-first search. Although random is still winning, the breadth-first search is actually starting to perform strongly in the mean. Remembering our experience with Tasks 544 and

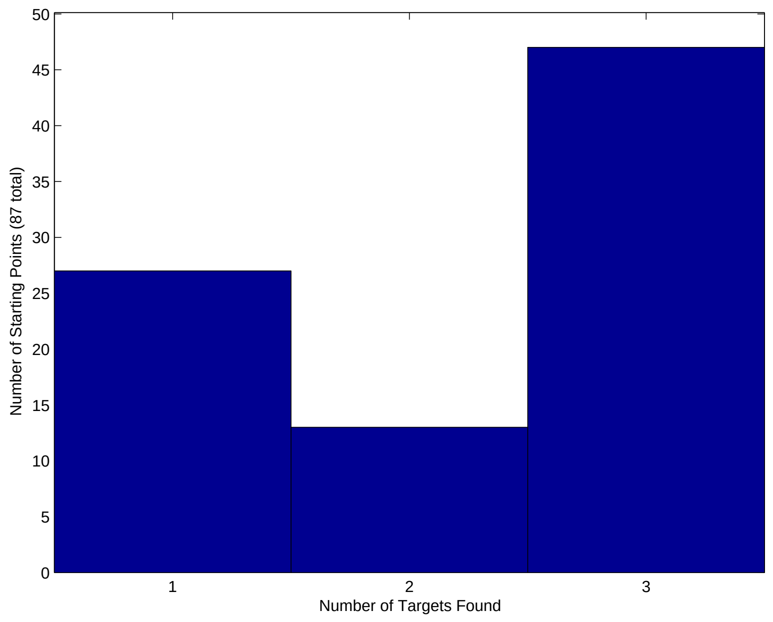


Figure 5.55: TREC Task 519: Starting at pages 3 away (87 starts) to depth 5 searching to exhaustion. Distribution of targets found. Mean number of targets: 2.2, median: 3. Mean number of pages in search-cone: 1267, median: 1351.

541, this is expected behavior.

Figure 5.56 summarize our results for Task 519 starting at height-3. Like for our previous tasks, the naive searches have their usual distinct shapes, with an exaggerated shift of peak as the search depth increases. Also we see typical distributions for our SVM-based spider. The SVM spiders are finding the first target within the first 10% of the search cone almost half the time. Much better than any of the naive searches.

In the next phase of testing the exploration capabilities of our spiders on Task 519 we increase our height to four. We begin with a search-depth of four. Remember that this creates search-cones whose target bearing layer is at the maximum depth and in previous tests tends to favor the depth-first search.

Looking at Figure 5.57 we see the distribution of targets found per search-cone. The distribution very strongly favors finding only one target. This is somewhat similar to the cones for Task 541 (Figure 5.43) but is dramatically different from that of Task 544 (Figure 5.11). We see that our SVM-based spiders are losing to the depth-first

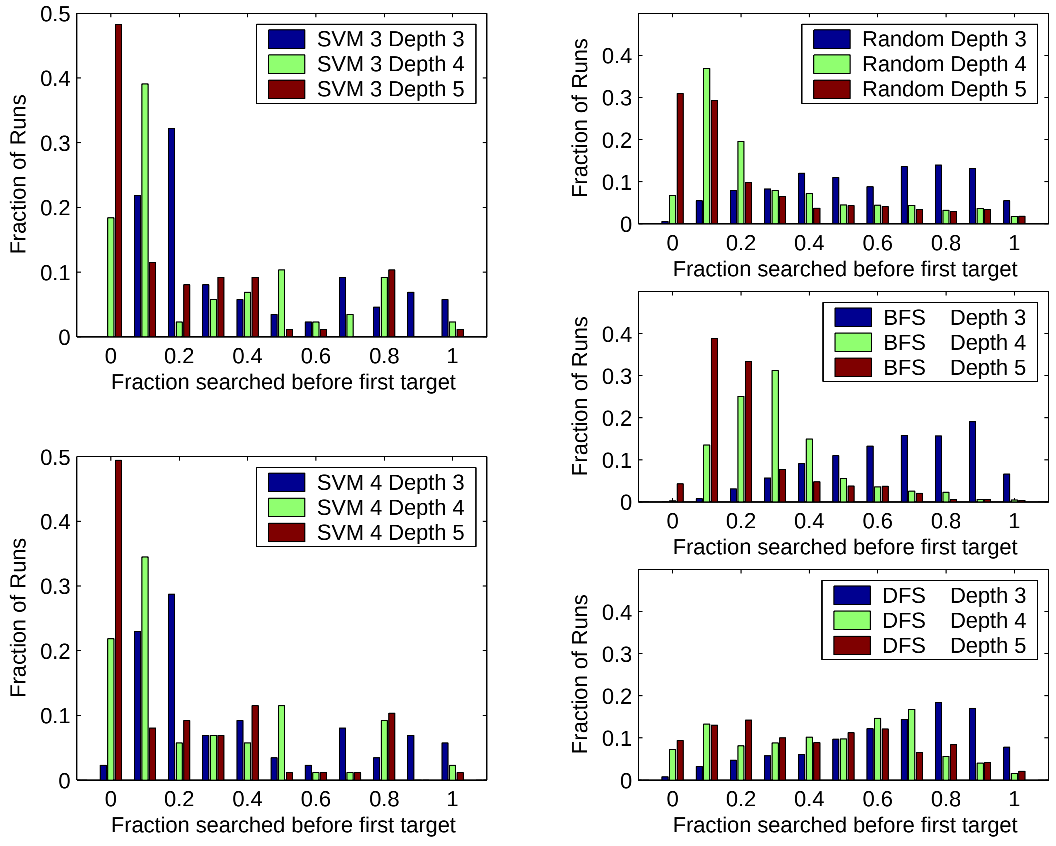


Figure 5.56: TREC Task 519: Starting from pages 3 away (87 starts) from targets and searching to depths 3, 4, and 5. Distribution of page retrievals to find first target divided by total pages in search cone.

	Machine	Δ0.05 Discount	0.95 Significance	Δ0.09 Discount	0.95 Significance	Δ First Target	0.95 Significance
Gold Depth	Random	85.2(211)	yes	12.9(32)	yes	-283(-700)	yes
	Breadth	106(261)	yes	16.1(39.7)	yes	-352(-868)	yes
	Depth	65.8(143)	yes	10(21.7)	yes	-218(-475)	yes
Gold Discount	Random	81.7(188)	yes	12.5(28.6)	yes	-272(-624)	yes
	Breadth	105(238)	yes	15.9(36.2)	yes	-348(-792)	yes
	Depth	63.4(120)	yes	9.63(18.3)	yes	-210(-399)	yes
SVM 1	Random	-7.83(51.5)	yes	-1.19(7.82)	yes	26(-171)	yes
	Breadth	2.71(102)	yes	0.412(15.5)	yes	-9(-338)	yes
	Depth	-15.5(-16.4)	yes	-2.36(-2.49)	yes	51.5(54.5)	yes
SVM 2	Random	-9.18(41.1)	yes	-1.4(6.24)	yes	30.5(-136)	yes
	Breadth	1.2(91.5)	yes	0.183(13.9)	yes	-4(-304)	yes
	Depth	-20.2(-26.8)	yes	-3.07(-4.07)	yes	67(89)	yes

Table 5.27: TREC Task 519: 4 away to depth 4. Median (and mean) differences in log discounted reward with 0.95 significance (Wilcoxon Sign-Rank).

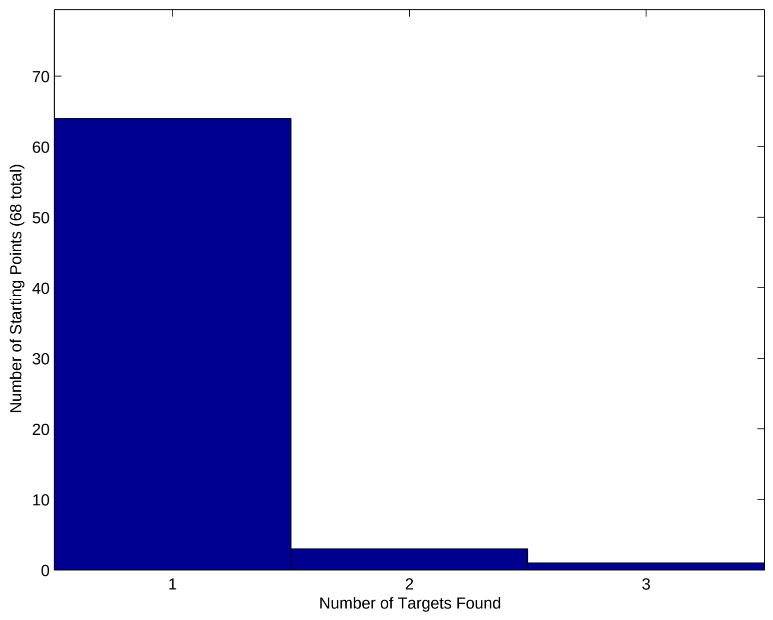


Figure 5.57: TREC Task 519: Starting at pages 4 away (68 starts) to depth 4 searching to exhaustion. Distribution of targets found. Mean number of targets: 1.1, median: 1. Mean number of pages in search-cone: 1239, median: 532.

search and random search in median, but beating the random search in mean.

The odd performance pattern continues to manifest as we increase the search depth to 5. As Figure 5.58 shows, we are finding only a few more targets although the search-cone size has doubled. Looking at the search performance scores, we see that the naive searches are now performing in their expected order for this height and depth: Depth-first beating breadth-first with random bracketed between them. Unfortunately, the SVM based spiders are now losing across the board in the median scores, however the differences in performance are only significant for the breadth-first search. Interestingly, the SVM spiders are significantly outperforming the naive spiders when examining the mean.

Looking at Figure 5.59 we can get a little bit better feel for what is happening in these tests. As usual, the naive spiders have their typical distributions. The distribution for the SVM-based spiders are unusual. They show a peak of performance early for about 20% of the runs with another peak of about 25% having particularly poor performance. This difference in behavior may be due to the poor quality of the

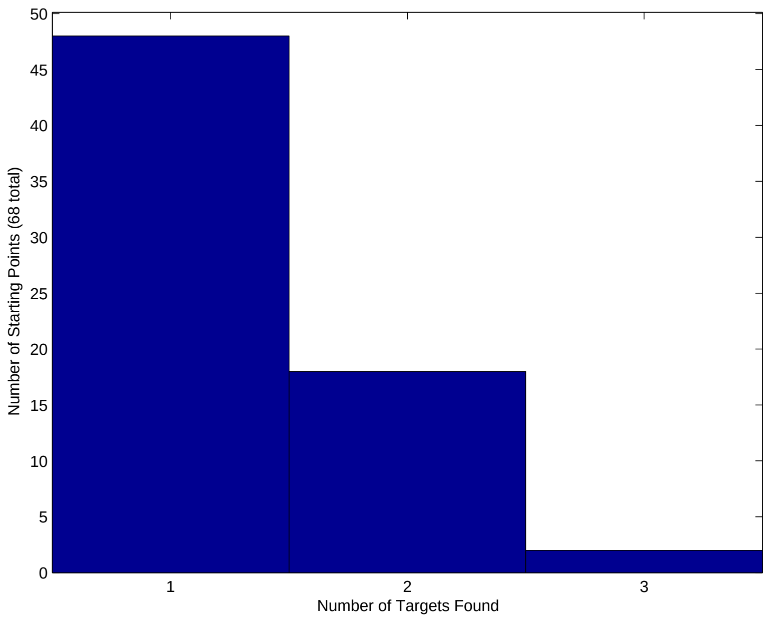


Figure 5.58: TREC Task 519: Starting at pages 4 away (68 starts) to depth 5 searching to exhaustion. Distribution of targets found. Mean number of targets: 1.3, median: 1. Mean number of pages in search-cone: 3620, median: 864.

	Machine	Δ0.05 Discount	0.95 Significance	Δ0.09 Discount	0.95 Significance	Δ First Target	0.95 Significance
Gold Depth	Random	114(403)	yes	17.3(61.2)	yes	-379(-1.34e+03)	yes
	Breadth	106(261)	yes	16.1(39.7)	yes	-352(-868)	yes
	Depth	101(342)	yes	15.3(51.9)	yes	-335(-1.13e+03)	yes
Gold Discount	Random	113(401)	yes	17.2(61)	yes	-375(-1.33e+03)	yes
	Breadth	104(260)	yes	15.9(39.5)	yes	-346(-862)	yes
	Depth	99.5(340)	yes	15.1(51.7)	yes	-330(-1.13e+03)	yes
SVM 1	Random	-1.81(252)	yes	-0.275(38.2)	yes	6(-836)	yes
	Breadth	-22.6(110)	yes	-3.43(16.7)	yes	75(-365)	yes
	Depth	-10.1(190)	no	-1.53(28.9)	no	33.5(-632)	no
SVM 2	Random	-4.82(247)	no	-0.732(37.5)	no	16(-819)	no
	Breadth	-26.8(105)	yes	-4.07(16)	yes	89(-349)	yes
	Depth	-13.4(185)	no	-2.06(28.2)	no	44.5(-616)	no

Table 5.28: TREC Task 519: 4 away to depth 5. Median (and mean) differences in log discounted reward with 0.95 significance (Wilcoxon Sign-Rank).

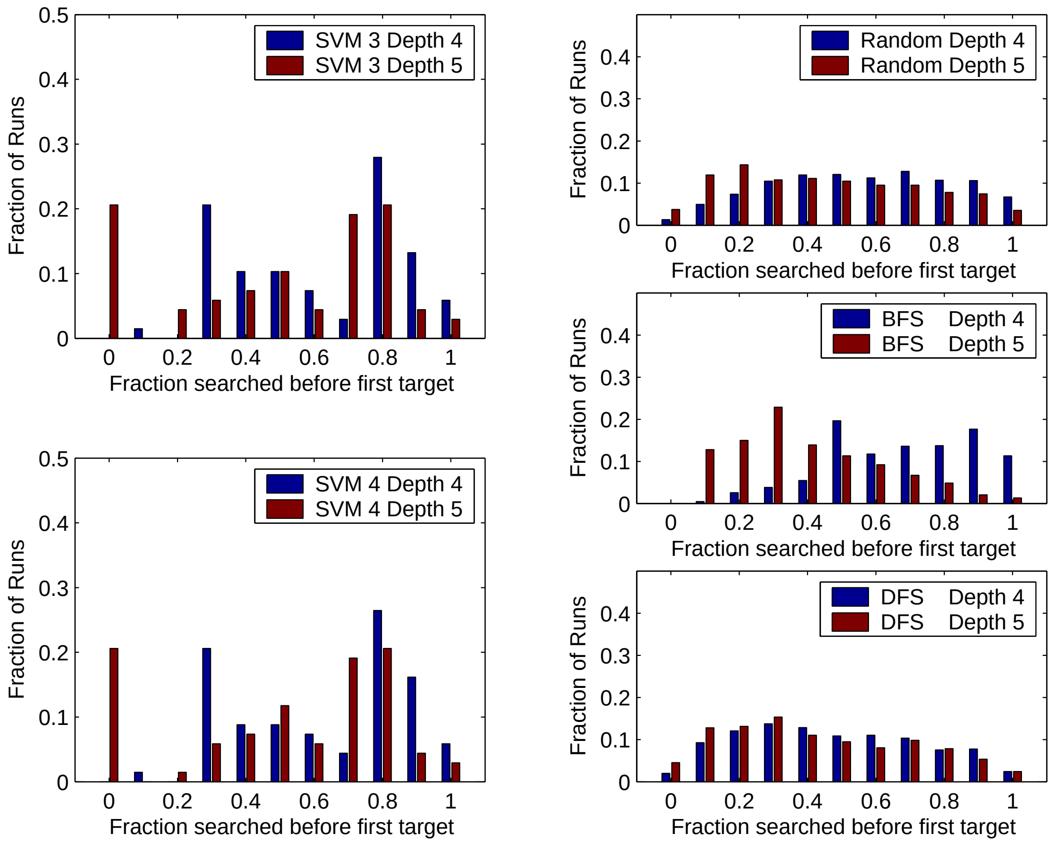


Figure 5.59: TREC Task 519: Starting from pages 4 away (68 starts) from targets and searching to depths 4 and 5. Distribution of page retrievals to find first target divided by total pages in search cone. SVM 3 and 4 here corresponded to SVM 1 and 2 respectively in Table 5.28

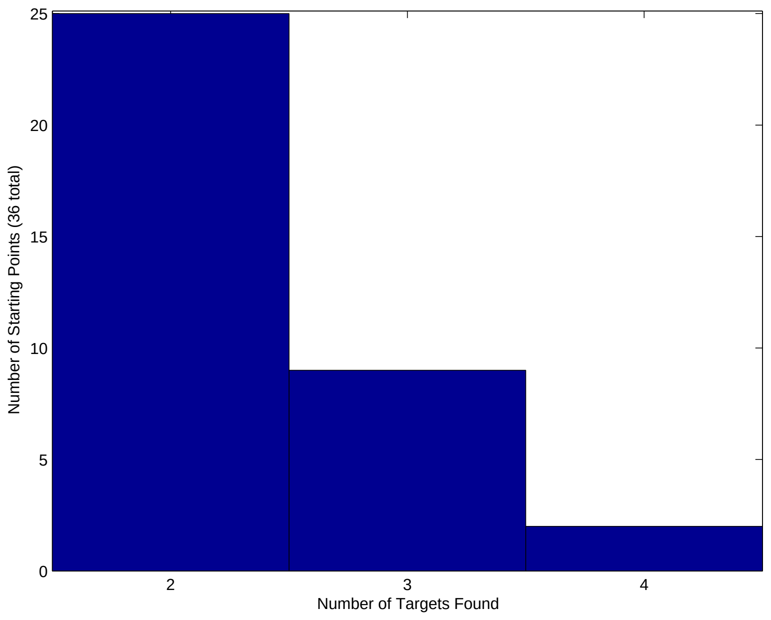


Figure 5.60: TREC Task 519: Starting from targets (36 starts) to depth 4 searching to exhaustion. Distribution of targets found. Mean number of targets: 2.4, median: 2. Mean number of pages in search-cone: 431, median: 309.

targets for this task, that is, since the nature of the targets is varied, the pages the spider was trained on might be quite different from those found in the cones for the 25% throwing off our performance.

In the results above where we test the exploratory capabilities of our spiders for this task, our SVM based spiders were not as potent as we might have hoped. In these results here we test the ability of our spiders to exploit a found signal. In other words, their ability to search from targets for other targets.

Unfortunately, as Figure 5.60 shows, there aren't that many targets to be found. This is dramatically different from the distributions found for our previous tasks (Figures 5.16 and 5.46). In our first test we set the search depth to 4. Our SVM spiders do outperform the naive spiders, but not by a large margin.

As we increase the search depth to 5, only a few more targets are found (Figure 5.61) and examining our search results (Table 5.30 we see that the breadth first search has gained a lot of ground and wins sometimes (with or without significance). Also the difference for one of our SVM spiders with the random search is insignificant.

	Machine	Δ0.05 Discount	0.95 Significance	Δ0.09 Discount	0.95 Significance	Δ First Target	0.95 Significance
Gold Depth	Random	19(43.7)	yes	2.88(6.67)	yes	-63(-145)	yes
	Breadth	18.8(37.6)	yes	2.88(5.74)	yes	-62(-125)	yes
	Depth	21.7(64.8)	yes	3.29(9.87)	yes	-72(-215)	yes
Gold Discount	Random	15.4(37.3)	yes	2.39(5.69)	yes	-51(-124)	yes
	Breadth	13.5(31.2)	yes	2.06(4.76)	yes	-45(-104)	yes
	Depth	17.9(58.3)	yes	2.75(8.89)	yes	-59.5(-194)	yes
SVM 1	Random	1.2(5.04)	yes	0.196(0.781)	yes	-4(-16.7)	yes
	Breadth	1.51(-1.03)	yes	0.229(-0.151)	yes	-5(3.45)	yes
	Depth	2.9(26.1)	yes	0.408(3.98)	yes	-10(-86.7)	yes
SVM 2	Random	0.903(5.73)	yes	0.139(0.883)	yes	-3(-19)	yes
	Breadth	1.51(-0.348)	yes	0.229(-0.0484)	yes	-5(1.17)	yes
	Depth	2.9(26.8)	yes	0.412(4.08)	yes	-10(-89)	yes

Table 5.29: TREC Task 519: From targets to depth 4. Median (and mean) differences in log discounted reward with 0.95 significance (Wilcoxon Sign-Rank).

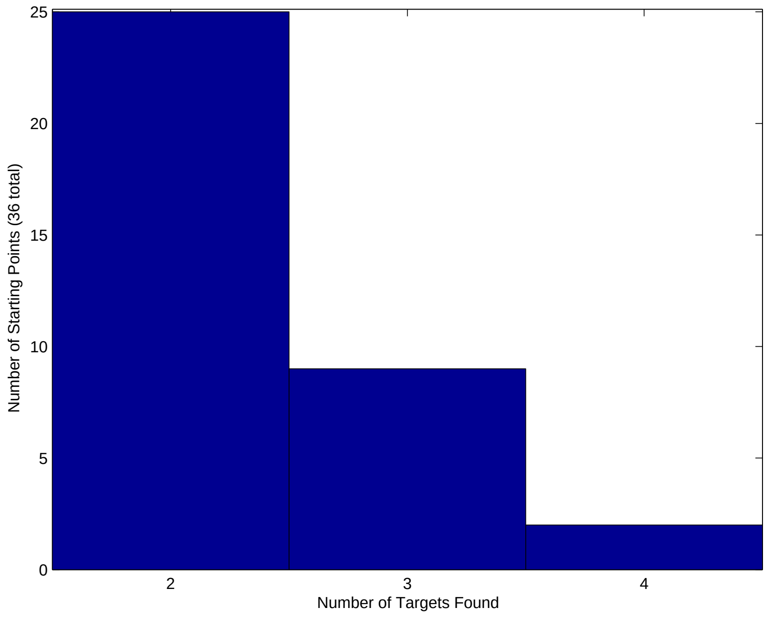


Figure 5.61: TREC Task 519: Starting from targets (36 starts) to depth 5 searching to exhaustion. Distribution of targets found. Mean number of targets: 2.7, median: 2. Mean number of pages in search-cone: 665, median: 439.

	Machine	Δ0.05 Discount	0.95 Significance	Δ0.09 Discount	0.95 Significance	Δ First Target	0.95 Significance
Gold Depth	Random	21.2(63.6)	yes	3.24(9.71)	yes	-70(-211)	yes
	Breadth	18.9(37.6)	yes	3(5.76)	yes	-62(-125)	yes
	Depth	28.6(105)	yes	4.47(16)	yes	-95(-350)	yes
Gold Discount	Random	18.1(57.6)	yes	2.84(8.79)	yes	-60(-191)	yes
	Breadth	13.7(31.6)	yes	2.19(4.85)	yes	-45.5(-105)	yes
	Depth	24.5(99.3)	yes	3.8(15.1)	yes	-81.5(-330)	yes
SVM 1	Random	1.31(12.2)	yes	0.204(1.86)	yes	-4.5(-40.4)	yes
	Breadth	0.301(-13.8)	no	0.0458(-2.09)	no	-1(45.8)	no
	Depth	3.61(53.9)	yes	0.549(8.18)	yes	-12(-179)	yes
SVM 2	Random	0(5.72)	no	0(0.874)	no	0(-18.9)	no
	Breadth	-0.301(-20.2)	yes	-0.0462(-3.07)	yes	1(67.3)	yes
	Depth	2.41(47.4)	yes	0.326(7.2)	yes	-8(-157)	yes

Table 5.30: TREC Task 519: From targets to depth 5. Median (and mean) differences in log discounted reward with 0.95 significance (Wilcoxon Sign-Rank).

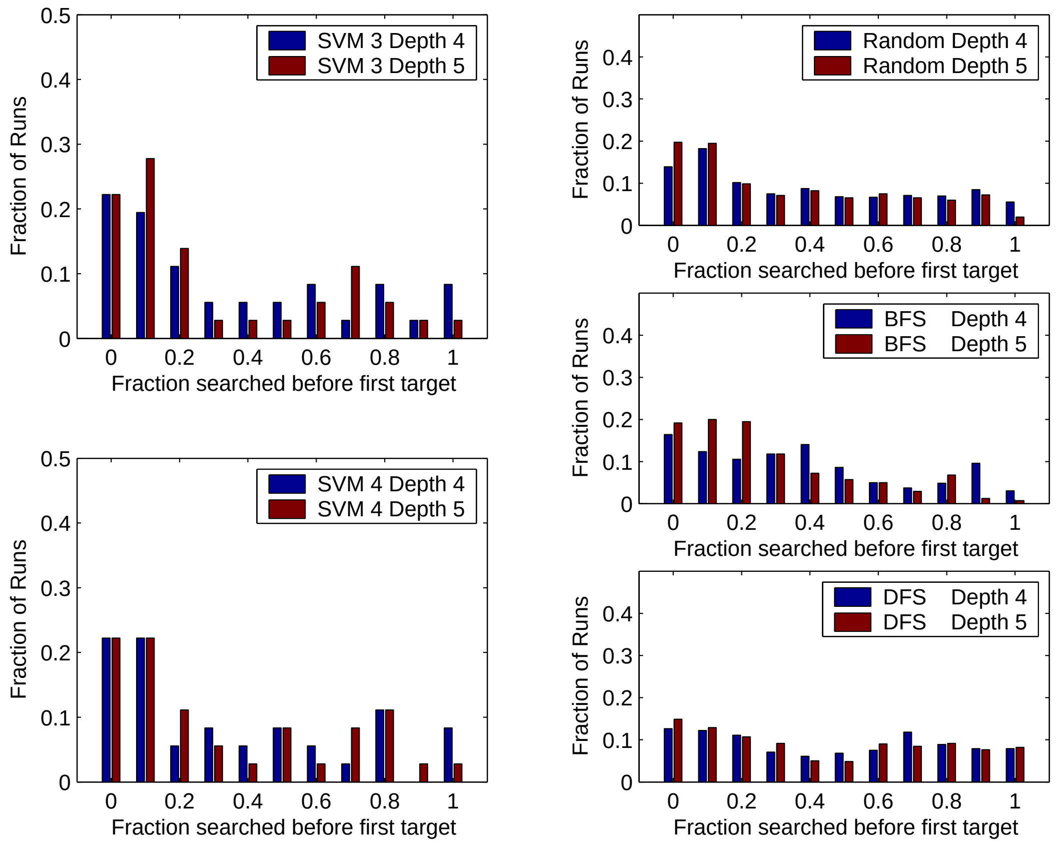


Figure 5.62: TREC Task 519: Starting from targets (36 starts) and searching to depths 4 and 5. Distribution of page retrievals to find first target divided by total pages in search cone.

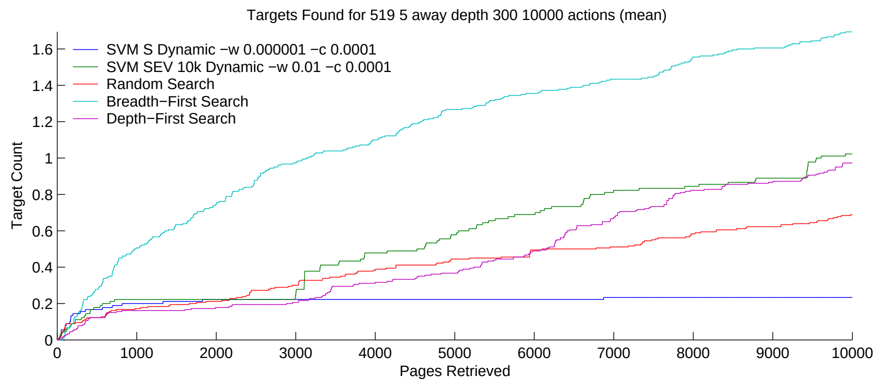
Let's take a look at our summary distribution and see what is happening. Looking at Figure 5.62 we see distributions for the naive searches that are fairly typical, however if we go back and compare the SVM distributions with the ones for Task 541 (Figure 5.48) and Task 544 (Figure 5.24) we see that they are not getting the large peak at the left side. Looking at the curves, we definitely can see a strong performance for about 40% of the search-cones (Runs), but this is balanced by the very weak performance for about 10% of the search-cones.

Long-range Evaluation

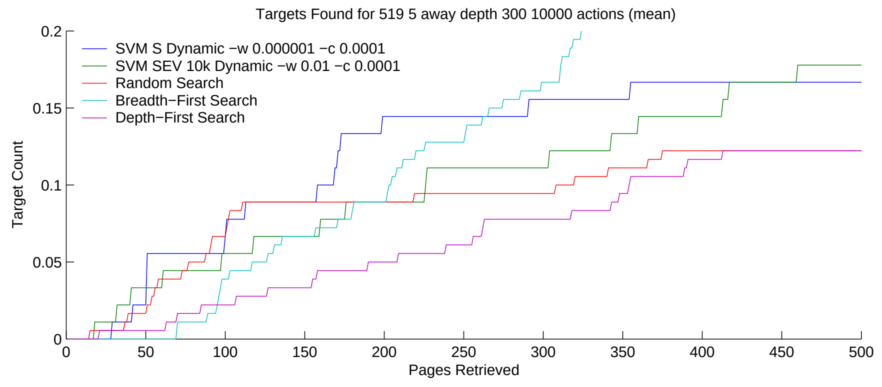
In the previous section we surveyed our test on short-range spider performance against the 519 Task. In some cases we got good results, in others neutral results, and in a few we got negative results. In this section we examine the long-range performance of our spiders. Unfortunately, for historic reasons, we do not have long-range results at each depth for both of the spiders used in the short-range experiments. We do have results for the augmented-delta spider used.

In our first long-range experiment, we start from 90 pages at height-5 above the testing targets. We run our experiments for $10,000$ page retrievals with a depth-limit of 300 (effectively infinite). Figure 5.63 shows the mean number of targets found across the spiders used in this experiment. These SVM machines selected for these spiders are representational. We actually tried many such machines on this task. These results are representative of the best performers. As can be seen from the results, the breadth-first search is outperforming our best SVM-based spiders. In Figure 5.63(c), we see that there is quite a bit of room for improvement before we catch up with the gold standard spiders. Unusually, there is quite a large gap between the spider with perfect depth information for the full depth and that limited to only perfect knowledge to a max of height-4. This indicates that there is a wide distance between targets, that is, the spider finds a target and then has to search at random because there is no new scent until it finally gets close enough.

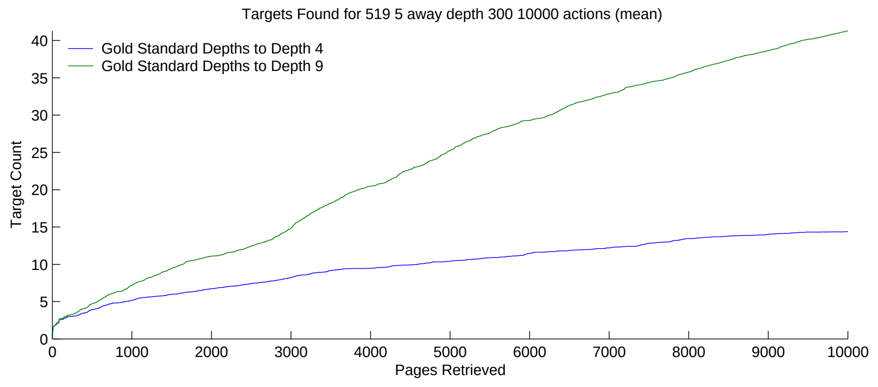
In Figure 5.64 we show a histogram showing the number of runs finding their first target ten different points through the $10,000$ page retrievals. We can see that the breadth-first search and the SVM based spiders are doing the best at finding a target in the first 1000 retrievals. The final set of bars at the far right, shows the number



(a) Performance across 10, 000 actions



(b) Zoomed in to show detail



(c) Gold Standard Spiders

Figure 5.63: TREC Task 519: Starting at 90 pages 5 away to depth 300 for 10,000 page retrievals. Mean performance of spiders.

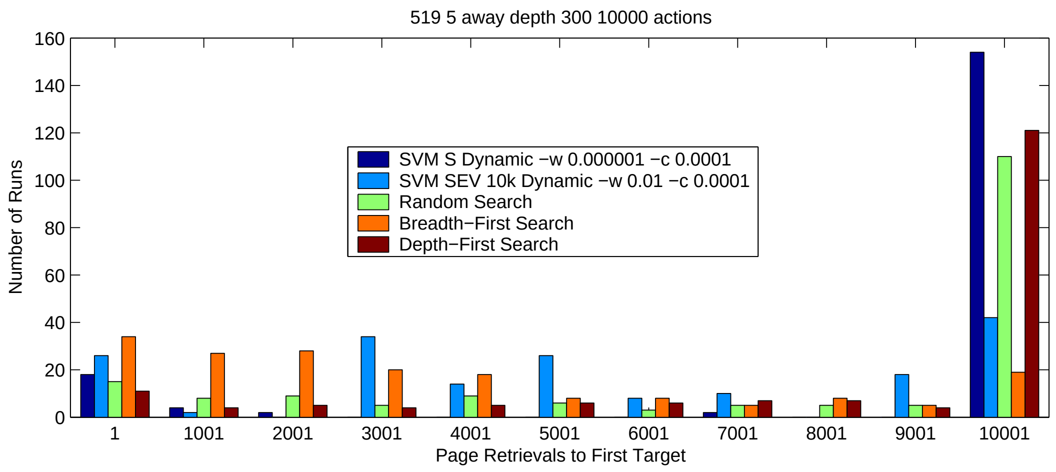


Figure 5.64: TREC Task 519: Starting at 90 pages 5 away to depth 40 for 20,000 page retrievals. Histogram of page retrievals to first target vs. number of runs.

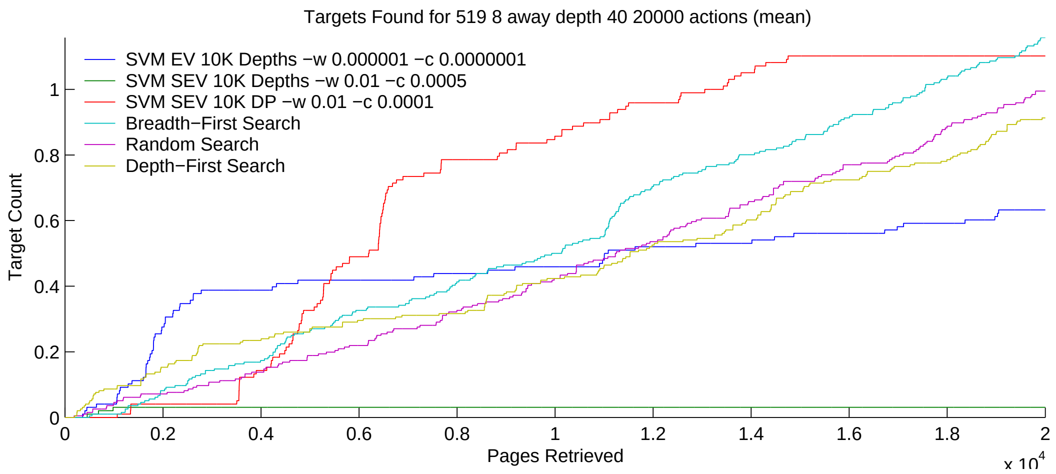


Figure 5.65: TREC Task 519: Starting at 98 pages 8 away to depth 40 for 20,000 page retrievals. Mean performance of spiders.

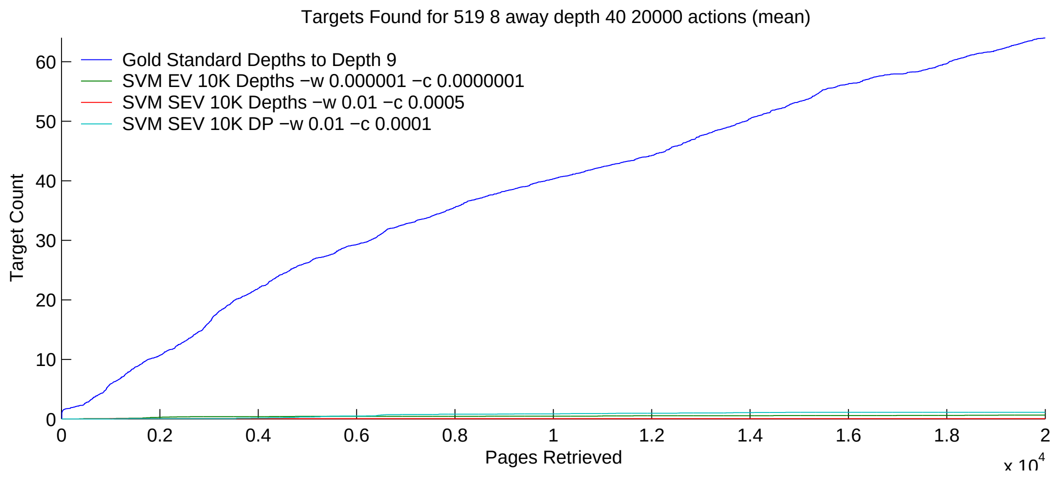


Figure 5.66: TREC Task 519: Starting at 98 pages 8 away to depth 40 for 20,000 page retrievals. Mean performance of the Gold Standard spiders.

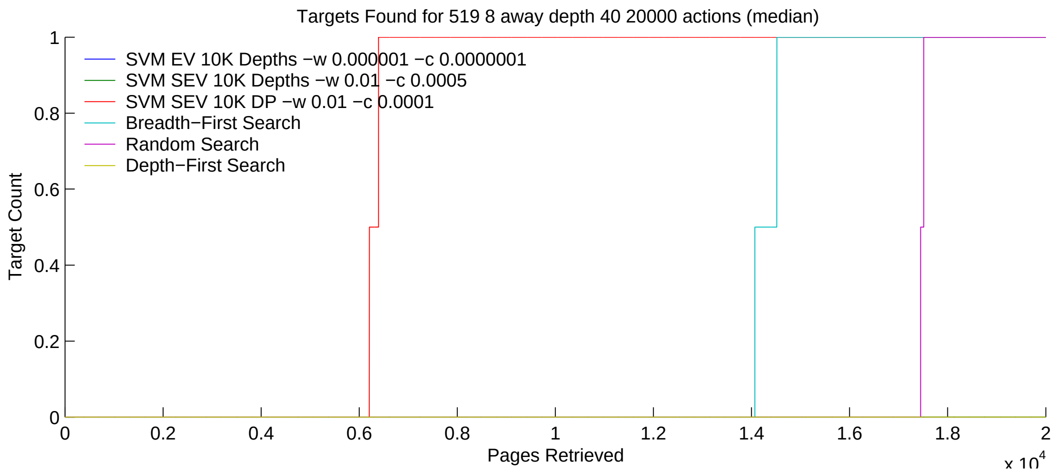


Figure 5.67: TREC Task 519: Starting at 98 pages 8 away to depth 40 for 20,000 page retrievals. Median performance of spiders.

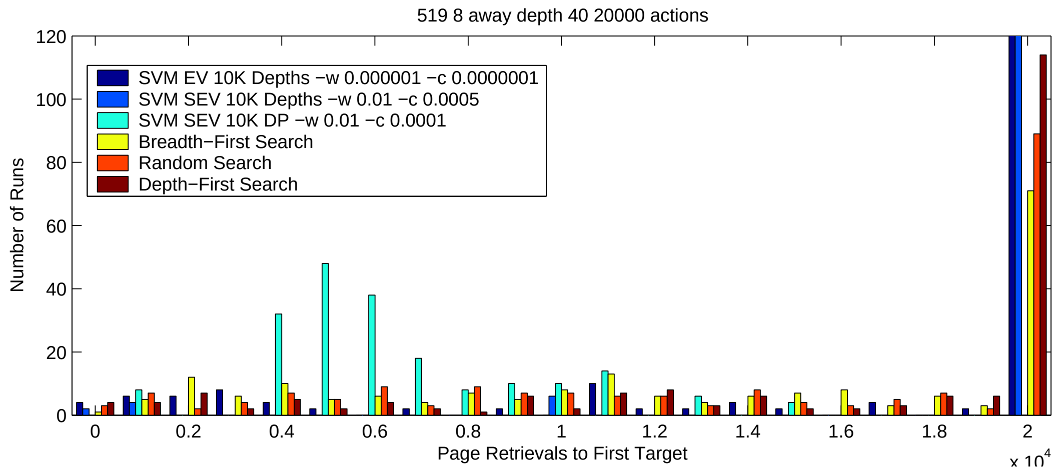


Figure 5.68: TREC Task 519: Starting at 98 pages 8 away to depth 40 for 20,000 page retrievals. Histogram of page retrievals to first target vs. number of runs.

of runs in which no target was found. Note that the number of runs is doubled for the SVM spiders to be comparable with the naive spiders which are run twice with different random seeds.

In our second set of long-range experiments, we pushed the height up to 8 above our testing targets and run for $20,000$ page retrievals to depth-40 with 98 starting pages (2 were rejected as they dead-ended in less than 20,000 retrievals. In the previous two tasks, 544 and 541, we had outstanding results for our SVM-based spiders on this task. That's not the case this time. In Figure 5.65 we can see our results compared with the naive spiders. If we had stopped our experiment early, at only $10,000$ retrievals, one of our SVMs would appear dominant; however, in the long run, the breadth-first search spider catches up. Also interesting is that the growth curves for the naive spiders are almost completely linear and don't exhibit a drop off like we've seen for the other experiments. Figure 5.66 shows the mean performance of the gold standard spider. Interestingly, if we look at the median performance (Figure 5.67) we see that our best SVM-based spider is reaching the 1 median target mark more than twice as fast as the naive spiders.

Figure 5.68 shows a similar histogram as seen in Figure 5.64. Looking to the far right we see our best SVM is finding targets in all cases. With a nice hump in the 4000 to 7000 page range. This tells us that, while we aren't getting great results vs. the naive searches for this task, we are learning a signal that works in some cases.

Summary

In Task 541, Instruments for Forecasting Weather, we were able to, after very careful training, construct a successful spider. In Task 519, we were not so successful. We were able to construct some spiders that were successful for some experiments or that performed well from some vantage points, but not the broad spectrum strengths we had earlier.

5.3 Similarity Measures and Ordering Hyperlinks[1]

Menczer [43] conjectures that there is a strong correlation between the distance between two pages in the Web graph (the length of the shortest directed path from one page to the other) and various similarity measures for comparing two pages derived from the standard vector space model used in information retrieval. In order to explore his conjecture, he performed an experiment which we summarize here. First, he chose 100 topic pages from the Yahoo directory. For each topic page he performed a breadth-first crawl to a depth of 3. This resulted in over $300,000$ pages which were then summarized as weighted term-frequency vectors using standard methods for stemming and weighting. To compare two pages, he adopted the cosine angle measure:

$$\sigma(p_1, p_2) = \frac{\sum_{k \in p_1 \cap p_2} w_{p_1}^k w_{p_2}^k}{\sqrt{\sum_{k \in p_1} (w_{p_1}^k)^2 \sum_{k \in p_2} (w_{p_2}^k)^2}}$$

where p_1 and p_2 are pages, k refers to a term, and $w_{p_i}^k$ is the weight assigned the kth term in the weight vector for the ith page. He then summarized this data using the following averages:

$$\delta(q, d) = \frac{1}{N_d^q} \sum_{i=1}^{d} i(N_i^q - N_{i-1}^q)$$

$$\sigma(q, d) = \frac{1}{N_d^q} \sum_{p \in P_d^q} \sigma(q, p)$$

[1]Joint work with Katrina Ligett and Thomas Dean

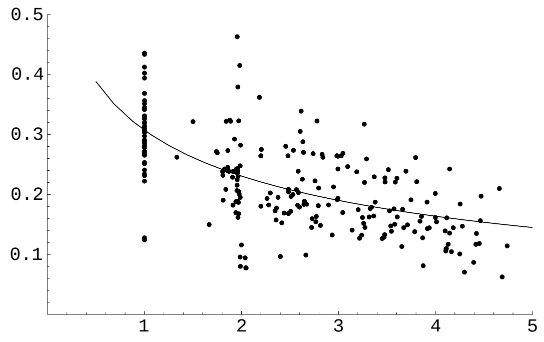


Figure 5.69: Scatter plot of $\sigma(t, d)$ versus $\delta(t, d)$ for TREC classes $t = 501, ..., 550$ and depths $d = 1, 2, 3, 4, 5$.

where N_d^q is the size of the cumulative page set $P_d^q = \{p : \delta_l(q, p) \leq d\}$ and $\delta_l(p_1, p_2)$ is the length of the shortest directed path from p_1 to p_2.

Menczer was able to show a relatively strong (negative) correlation between link distance and content similarity, which might explain what our SVMs were learning in the case of the successful searches. Menczer's data is interesting, but not immediately relevant to the problem of explaining Web searching behavior. Menczer's study used the link distance from target to non-target pages; we are interested in how well the link distance from non-target pages to target pages correlates with similarity measures like cosine angle. We set out to see if the Web pages in our WT10g samples exhibit such a correlation.

For each of the target pages in each of the 50 topic classes, we computed the cosine angle measure against a random sample of pages of up to (link) distance five. Figure 5.69 displays five data points for each topic class: data averaged over the sample of pages at distances of up to one, two, three, four, and five. As might be expected, the similarity between near neighbors demonstrates great variability, even when averaged by topic.

In general, most of the many pages that are five links away from a given page demonstrate very little similarity to the original page. As a result, a strong correlation between link distance and the cosine angle measure for a given topic may serve as an indicator that pages within that topic tend to be tightly clustered. Six topics out of fifty demonstrated no significant correlation or a weak inverse correlation at the 0.05 level of significance. The other topics, including #519, #541 and #544,

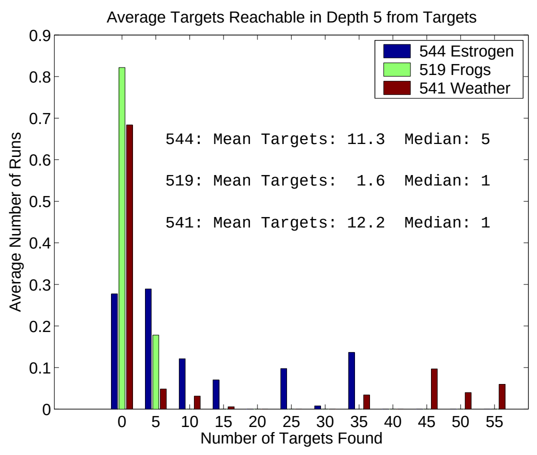


Figure 5.70: Histogram showing the numbers of additional targets reachable within five steps starting from a target.

demonstrated varying levels of positive correlation. We found similar patterns of correlation using information theoretic divergence measures such as Jensen-Shannon divergence [36, 49].

While the sample of target classes is relatively small, this evidence suggests that these correlations may explain some of the performance variance seen in previous experiments: topic #544, which yielded by far the best results of the three topics, demonstrated the fourth-strongest correlation of all fifty TREC topics. In particular, the correlation seen in #544 is more than twice as strong as that seen on either #541 or #519.

In addition, on the 31 topics which had a large (> 30) number of target pages, we selected 13 targets at random and computed the correlation between their semantic similarity to other pages and those other pages' distances to their nearest target. This correlation was significant at the 0.05 level for all 31 topics studied, providing strong evidence for the cosine angle measure as a useful signal for guiding focused crawling.

We also investigated graph theoretic properties of the subgraphs involving target classes. Figure 5.70 illustrates one of our approaches to uncovering such structure. To produce this graph we conducted breadth-first search crawls from each of the targets in target classes #544, #519 and #541 to a depth of five. The graphs are color coded; if you have only a black and white reproduction of this document, each trio of vertical bars is arranged so that the leftmost bar is #544, the middle bar is #519 and the rightmost bar is #541.

Figures 5.70 is a histogram (three histograms actually, one for each of the three target classes) with buckets 0–4, 5–9, etc., indicating how many target pages are reachable from targets. For example, the graph shows that approximately 30% of the #544 targets have between 0–4 targets within five steps, another 30% have between 5–9 targets within five steps, etc. #544 has considerable variation in target clusters; more often than not a spider finding one target will find several more in the area, #519 has very little variability in target clusters; targets tend to be isolated and a spider gets little advantage from finding one target, and #541 is more like #544 than #519.

Based on our studies, it seemed reasonable to suggest that the cosine angle measure might provide a useful signal for guiding focused crawling and that, in particular, it might provide not only guidance in seeking out individual target pages but, indeed, rich clusters of target pages. We need to determine if the correlation between link distance and semantic distance generalizes over targets in the sense that the semantic distance between a particular page and a known, exemplar target provides a reliable estimate of the link distance between that page and its nearest target. If so, then there is an obvious class of algorithms one might use to expedite focused crawling.

5.3.1 The Search Problem

There is no shortage of proposals for combining link and content information [11, 9, 14, 52]. We conjecture that much of the effectiveness of these methods comes from exploiting the correlation between link distance and content similarity discussed above. Moreover, we expect that a good deal of the useful information extracted by these methods can be approximated using easily computed measures of content similarity such as cosine angle and Jensen-Shannon divergence.

Throughout the remainder of this section, we focus on the model of specialized search described in [60] in which search for pages in a specified target class is guided by a set of representative target pages, T. We assume an exploratory phase in which we examine additional pages in the vicinity of the representative targets in order to collect information about the link distance from pages found in the vicinity of targets to the targets themselves.

In our earlier experiments, the exploratory phase was used to gather a set of pairs of the form (p, z) (a supervised training example) where p is a page and z is the length of the shortest path to a target (we also experimented with additional measures based on a discounted reward model). These pairs were used to train support vector machines to estimate the distance from pages to the nearest targets. The learned machines were then used during searching to rank order the set of pages in the *fringe*, i.e., those pages that have been discovered during focused crawling, but not yet explored.

In this paper, we will use the semantic distance to known targets as an estimate of the link distance from a page to its nearest target. For each known target $t \in T$, define $f_t(u) = d(u, t)$ where d is the Jensen-Shannon divergence, the cosine angle, or some other measure that tends to correlate well with link distance. The set $\{f_t | t \in T\}$ represents different sources of information regarding the distance to targets.

When employing the spider we don't have actual text of the pages in the fringe as we haven't retrieved the pages yet. Instead we use an approximation of that page. For each link (u, v) to a page v in the fringe, we approximate v from the text of page u and the anchor text associated with the link to v in the text of u. Specifically, we use the term-frequency vector $\vec{u} + M\vec{a}_{uv}$ where $\vec{u}$ is the term-frequency vector for the text of page u, $\vec{a}_{uv}$ is the term-frequency vector of the anchor text in page u for the hyperlink to page v and M is a multiplier used to "amplify" the influence of the anchor text. In our experiments, we have found that values of M between 10 and 20 work well. In the following development we ignore this runtime approximation.

We could pick a particular function f_t to rank pages in the fringe, or alternatively we could combine the functions to form a composite ranking function. The primary advantage of using a single target is simplicity, but we prefer to use all of the information available and so we learn a strategy for combining the functions f_t for each target t into a single search heuristic to guide the spider's best first search.

5.3.2 Learning to Order Hyperlinks

Cohen, Schapire and Singer [12]) described an algorithm for ranking which is related to Littlestone and Warmuth's "weighted majority algorithm" [37] and more directly to Freund and Schapire's "Hedge" algorithm [22]. They provide a method for learning a linear combination of experts that yields a *preference function* over a set of items and a greedy approximation algorithm for finding a total ordering of those items that agrees best with the learned preference function.

In the parlance of Cohen et al., the functions $\{f_t | t \in T\}$ are called *ordering functions*. An ordering function f induces a *preference function*, $R_f : X \times X \to [0,1]$:

$$R_f(u,v) = \begin{cases} 1 & \text{if } f(u) > f(v) \\ 0 & \text{if } f(u) < f(v) \\ \frac{1}{2} & \text{otherwise} \end{cases}$$

where $R_f(u,v) = 1$ is interpreted as meaning u is preferred over (or ranked higher than) v and $R_f(u,v) = \frac{1}{2}$ means that u and v are unranked relative to one another.

The Cohen et al. learning algorithm produces a composite preference function:

$$\text{PREF}(u,v) = \sum_{t \in T} w_t R_{f_t}(u,v)$$

by assigning a set of weights $\{w_t\}$ shown to minimize an appropriate measure of loss [12].

The algorithm described by Cohen et al. is an iterative on-line algorithm. On each iteration, the algorithm is given a set of instances to rank, and it produces a total ordering of the instances as a way of soliciting feedback. The instances in our case will be a set of Web pages. Feedback is in the form of a set of pairs $\{(u,v)\}$ which in our case correspond to pairs of Web pages with respect to their distance from targets. For our purposes all we require from the exploratory phase is a set of pairs $\{(u,v)\}$ expressing assertions of the form u is closer to a target than v.

Since the goal in the Cohen at al. algorithm is to learn a set of weights used to produce an ordering, we do not care how the composite preference function scores pairs of pages at the same depth. For example, if pages u and v are both three hops away from a target, we do not care whether the algorithm prefers v over u or vice versa. Accordingly, we do not train the weights with pairs of pages at the same depth. All of the other pairs of pages in the training graph are used.

There is one difference between our learning environment and that proposed by Cohen et al.: in their environment, the original ordering functions are unknown and hidden by the corresponding preference functions. Since, in our model, we are using

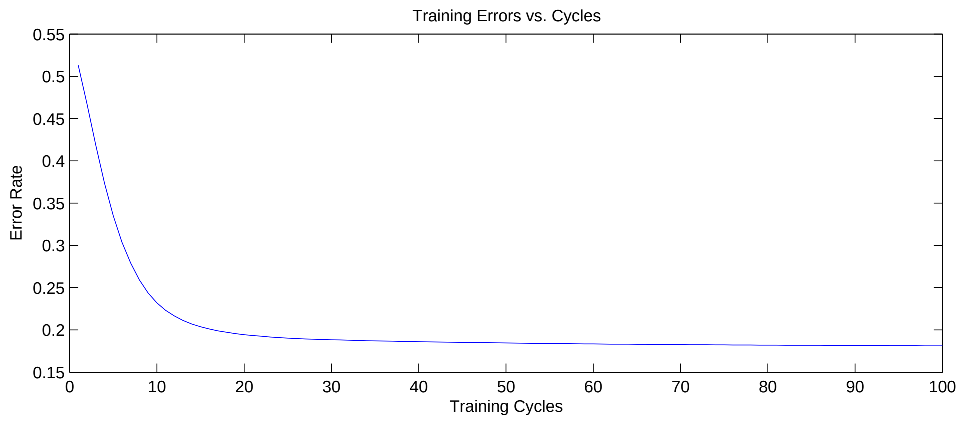


Figure 5.71: Total classification error vs. number of training cycles for a thirteen-target ensemble from the TREC 544 target set trained to a distance of four from the targets.

a distance metric (cosine-angle) which provides a total order, there is no advantage in masking this order with a preference function and then forming a linear combination of the preference functions. Instead we learn a linear combination of the ordering functions producing a composite ordering function. Our training schedule and loss functions are the same as those used by Cohen et al.

5.3.3 Experiments

In comparing the SVM-trained spiders with the techniques used in this paper, the training set of targets is used to learn the weights for the composite ordering functions described in the previous section. Before we compare performance of spiders with the different methods for guiding search it will be useful to examine the performance of the trained composite functions for ranking pages. We will scrutinize the TREC target class #544 in some detail. For comparison purposes, we'll consider composite functions trained from the same thirteen targets used to train the support vector machines discussed in Section 5.1 above.

We are interested in how often the composite function mis-orders pairs of pages. Figure 5.71 shows the total error (probability of mis-ordering) for a composite function trained to a distance of four from the thirteen targets and tested on pages drawn randomly from those pages four or less from the remaining targets. Like the SVM

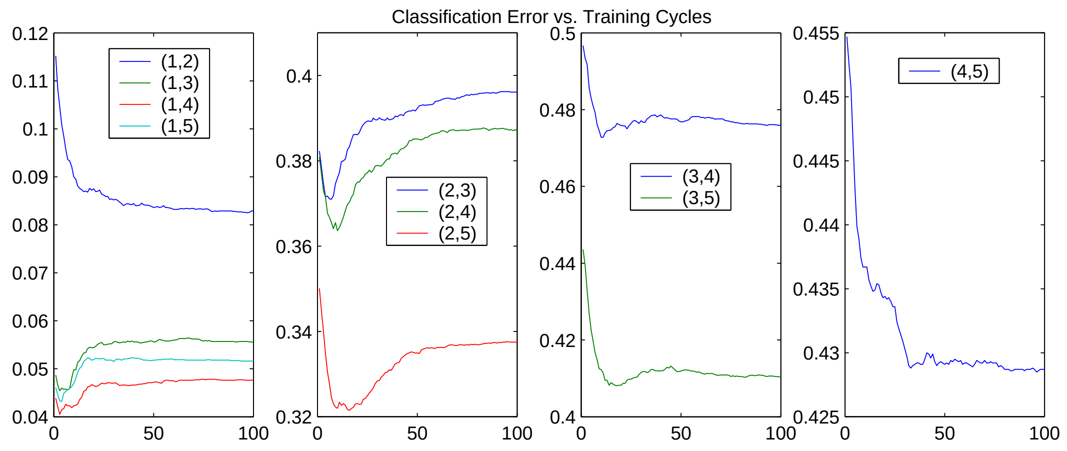


Figure 5.72: Classification error vs. training cycles for the same ensemble tested against all of the pages up to 4 hops away from targets in the 544 TREC task.

spiders we use training data trawled to a distance of 4 hops to targets. By training to order pages using only cosine-angle against term vectors for the target pages, we hope to learn a machine able to generalize outside of the training set. This graph tells only a small part of the story however. It could be, for example, that the signal fails close to the target. To examine learning in more detail, we will focus on performance between pairs of pages at specific distances from their nearest target.

In Figure 5.72 we see a set of curves representing sample training cycles versus error rate for a composite spider. Each line shows the error rate for ordering pairs of pages at the specified distance from targets. For example, "(1,2)" gives the probability for mis-ordering a pair of pages chosen at random where one is chosen from the set of targets and one is chosen from those pages which link to targets. Similarly "(5,6)" gives the probability of mis-ordering a pair consisting of one page four hops away from its nearest target and a second five hops away from its nearest target. Note that while the overall performance improves as was shown in Figure 5.71, the composite function actually gets somewhat worse in ordering pages close to targets.

In Figure 5.73 we show the classification error vs. training cycles using a trawl from those same thirteen targets but testing to a distance of seven. Note that additional targets were found in the extended trawl which were not trained against. Examining this graphic we see that we are still getting strong classification rates and a learning response even deep outside the training region. The major exception is at the "(7,8")

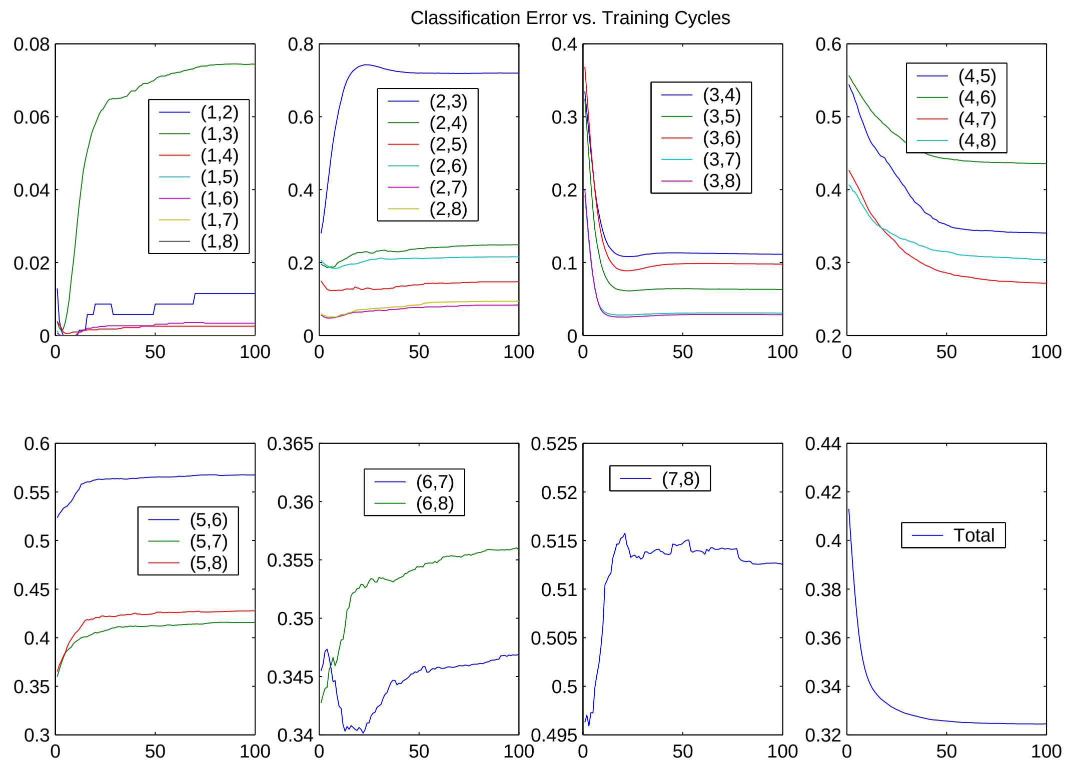


Figure 5.73: Classification error vs. training cycles for the same ensemble showing error rates to a distance of seven.

level where we actually have a negative training response. Notice the set of curves for pairs (third from left, top row) for pages two hops from targets ("(3,x)"). As the distance between the pages increases the ability to distinguish the pages increases. Remember that this machine only has knowledge of the target term-frequencies!

In Figure 5.72 we examine how well the machine generalizes against a trawl from all of the TREC 544 targets. Notice that in all cases with probability greater than 0.5 we pick the proper ordering for any two pages picked at random from the trawl.

Also note that in both Figure 5.73 and Figure 5.72 the training response is strongest in the first 15 cycles and then either tapers off or gets slightly worse. In the results discussed below we stop the training after 15 rounds. Just as training error drops rapidly, search performance increases quickly with training cycles (Figure 5.74). We can see that performance varies little between the 10 and 15 round runs.

In [60] we explored the use of support vector machines for learning regressions

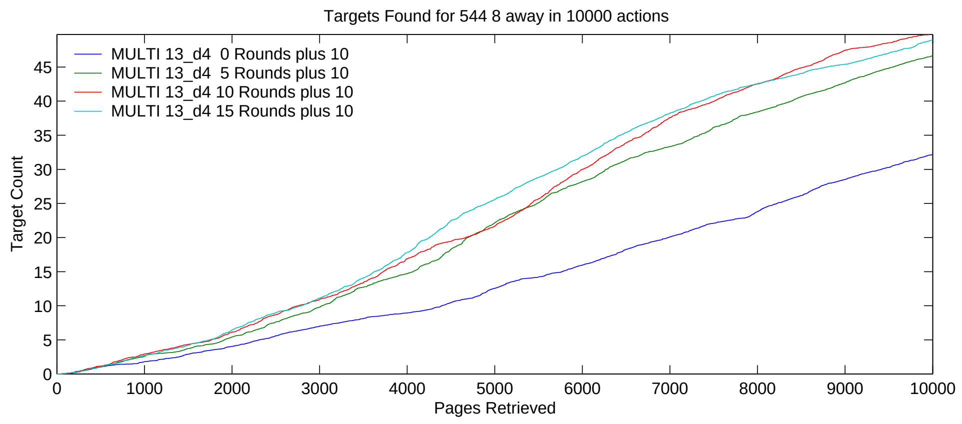


Figure 5.74: Search performance for spiders ranging from no training through fifteen rounds of training.

on navigational distance. With the algorithm described above, we are not trying to directly learn to predict navigational distance. Instead we are simply trying to learn to decide which node of any pair is closer to a target. How do these results compare to the performance of the support vector machine spiders? Figure 5.75 shows the performance of our cosine-angle spider compared with a naive breadth-first search as well as with one of our strongest SVM learned spiders. Note that the top two curves give the maximum performance we can reasonably hope to achieve from a spider trained with depth information; the top curve is with perfect knowledge to eight hops away from targets and the second with only knowledge to four hops (as used in training).

We mentioned earlier that for the 31 TREC target classes that had a large (> 30) number of target pages the correlation between link distance and semantic distance is significant indicating a potentially useful signal for guiding focused crawling. The existence of a signal doesn't guarantee however that we'll able to efficiently exploit that signal to find additional targets. Indeed in our additional experiments there are cases, e.g., #541, in which the signal (correlation) is relatively strong and we have trouble matching the performance of our SVM-guided spiders. There are also cases, e.g., #519, in which the signal, while relatively weak, affords a spider guided by a composite-ordering-function some success in finding targets where our SVM-guided spiders were not able to find any targets. For the WT10g TREC target classes we

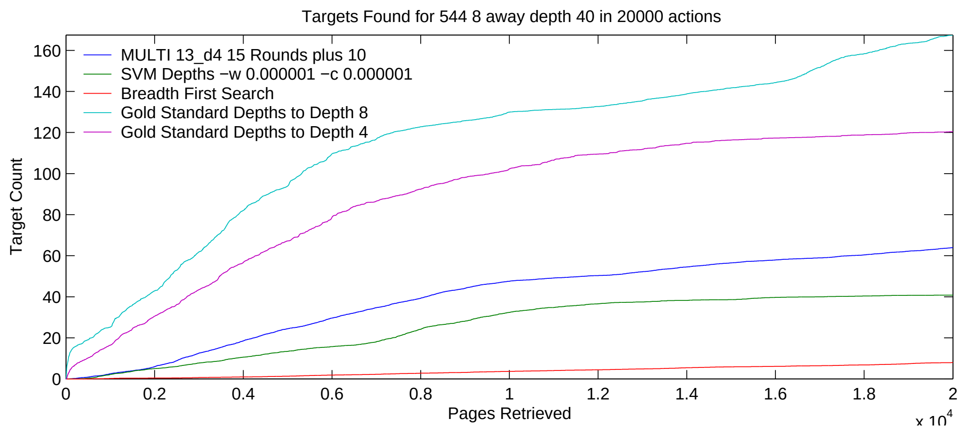


Figure 5.75: Mean performance from 101 targets for different spiders ranging from perfect knowledge of distance-to-target to breadth-first search.

have good signal strength data and we're currently working on a stochastic generative graph model with parameters based on our observed signal-strength distributions to provide predictions regarding spider performance.

5.3.4 Summary

This work was initiated to explain the performance of our earlier work on focused crawling. We found that the strength of the correlation between link distance and semantic distance with respect to particular target classes accounted for much of the observed variation in the performance of our SVM-guided algorithms. This explanation suggested a new class of ordering functions conceptually simpler than our earlier SVM-based techniques. Using simple linear combinations of preference functions provided significantly better performance than we expected, and suggests a host of related techniques for performing focused crawling.

One advantage of the preference-function approach discussed above is that it allows for a straightforward on-line training regimen to update the ordering preferences during search. As new targets are found, new ordering functions can be rolled into the initial set $\{f_t | t \in T\}$ and new feedback pairs can be generated from pages in the vicinity of the new targets. At any given point in time, a spider will have a set of expectations regarding the ordering of pages in the fringe; after following one or more links, some of those expectations may be reversed. We are considering various strategies for assigning confidence measures to such reversals and then using the resulting ordered pairs to provide qualified feedback of a form that can easily be incorporated into the Cohen et al. algorithm.

5.4 Summary

In this chapter we explored our results from several experiments in training spiders. The experimental environment was the WT10g environment using three different tasks drawn from the NIST TREC 2001 competition.

The searches tested support vector machine trained spiders against the naive searches (depth-first, breadth-first, and random search) in a variety of situations. We tested the searches with short-range experiments starting from only three hops away from targets ranging up to five hops with constrained search depths. An interesting pattern emerged showcasing the strengths of the various naive search strategies at different distances from targets and with different limitations on search depth.

At greater distances from the targets we showed that the learned spiders were able to vastly out-perform the naive searches. In these experiments the spiders were searching less than ten percent of the space of several hundred thousand pages to find more than 25 percent of the targets. Our long-range results showed, to the best of our knowledge for the first time, that spidering can be effective for searching from rare targets, starting even from large (with respect to the diameter) distances on the web. These results also suggested areas for further study and experimentation which will be discussed in the next chapter.

Chapter 6

Conclusion and Future Work

In the previous chapter we presented the results from extensive experimentation. We developed spiders trained to search for targets drawn from several tasks in the TREC competition on the WT10g web corpus. The results from these experiments confirms our hypothesis stated in the introduction. We asked a fundamental question about searching on the Web:

> **Question 2.** *Given training data in the form of a (relatively small) sample/subgraph of the web containing a subset of the target pages, is it possible to conduct a directed search and find additional target pages faster (with fewer page retrievals) than by performing a blind or uninformed random or systematic search, such as depth-first or breadth-first search?*

Assuming that the answer to that question was "yes," we asked the next question:

> **Question 3.** *If the answer to Question 2 is in the affirmative, can we efficiently process the training data to generate an efficient search heuristic achieving the desired performance?*

Our hypothesis was that the answer to both questions is "yes" for an interesting set of tasks. In Chapter 3 we presented early results showing that spidering on the live web could be effective. Because of artifacts from working on the live web, we switched our focus to working with the WT10g corpus. In Chapter 5 we presented our experiments with this corpus testing the hypothesis.

We showed that for TREC Task #544 we were quite effective at learning powerful spiders using both SVM regression with linear kernels and with our adaption of Cohen et al's algorithm for learning to order hyperlinks. Both of these algorithms produced spiders that worked better (and often much better) then the naive (depth-first, breadth-first, and random) spiders. Also they were efficient to train. On the other end of the spectrum, with TREC Task #519, with very careful training we were able to develop spiders that were somewhat effective.

Examining the structure of the web-graphs we could see many differences across the tasks. For example, as the tasks got more difficult to classify, the average target-to-target diameter increased while the target-to-all pages diameter stayed close to the across-the-board average. Also, the the quality of the targets degraded in the more difficult task. The difficulty of the task could be ameliorated somewhat by increasing the amount of data available to the offline learning algorithms.

6.1 Future Work

As indicated above, there remains a significant amount of work to understand the types of tasks in which spidering works. We need to more fully automate and perform the experiments discussed in Chapter 5 on more of the TREC tasks on the WT10g corpus. Then these results can be correlated against measures of the TREC tasks.

In additional to these further explorations of the TREC tasks on the WT10g corpus, we need further refine the techniques for web-spidering. There are several avenues for research on the spidering techniques. These include deriving better features to use in our vector space model and improved search heuristics.

The traditional paradigm used to frame the problem of web-spidering is that of best-first search on a directed graph. This paradigm is useful as it is very close to what is "really" happening and it simplifies the problem of next-action selection to that of maintaining and selecting from a priority queue of pages to retrieve or links to follow. The difficulty with this paradigm is that it is imposes an non-Markovian model preventing application of much of the learning and planning communities' efforts. Below we briefly discuss some of these avenues of future research.

6.1.1 Vector-Space Representations

In our experiments above we used a very simple vector-space representation of each web page. Each page was represented by a vector of frequency counts of the words observed on the page. This proved to be an adequate and fruitful representation, however there are many additional features and alternate presentations which might be used.

Possibilities we considered using were various standard normalizations such as the term frequency, inverse document frequency (TF-IDF) normalization. We didn't try these various normalizations as initial results using straight frequency counts seemed to work well enough to demonstrate the thesis. It would be worthwhile to repeat the experiments using a few different normalizations. One reasoning being that scaling the values into the 0 to 1 range is one technique for tuning the learning for support vector machines.

Potential additional features might draw from a more detailed parsing of the HTML in the pages, with separate features depending on the source of the terms, e.g., headers, links, tables. More interestingly, we might use more (web) graphical features such as out-degree, or terms from explored child/parent pages. We might also use estimated features calculated during the search such as PageRank or HITS scores. An issue with these types of features is that as a search progressed, the vector representation would change. This isn't a problem for the theory of the spidering, but would present a *small matter of coding* having to periodically re-score the actions on the fringe.

6.1.2 Search Heuristics

We proposed above using some graphical features as part of our vector space representation. In the experiments presented in this thesis we only captured graphical properties of pages in the value we tried to train the regressions to learn. That is, we used estimations of discounted reward or distance to nearest target (both properties of pages embedded in the web-graph) as values to try to estimate to use for the priority queues to guide the spiders. These were our search-heuristics.

It turned out in our experiments that the discounted reward was not any more effective than simply using the distance to nearest target measure. It was also very expensive to calculate for the training datasets. We only performed our spidering experiments using a 0.5 discount factor. It would be interesting to try other factors, especially ones closer to unity to see if that differentiates the spidering performance from simply using the distance to target.

However, since the discounted reward is so expensive to calculate we should try other approaches. One idea we have considered is using a max-flow approach where we put a source at the page being scored and sinks at each of targets and measure the flow from the source to sink given a limited capacity on each edge. This might be normalized by the number of edges in the flow. At one extreme this is identical to the discounted reward model with a discount-factor of 1.

6.1.3 Generative Models

One interesting possibility is evaluating spidering stratagems against generative web-models. Interesting generative models for the web have been proposed. Examples include Thomas Hoffman, et al. PHITS style models and Panduragon, Upfal style in-degree/PageRank models.

Working with generative models is interesting in a few ways. First we can evaluate, perhaps formally, the ability of a spidering stratagem in the context of a generative model. In other words, given a generative model, how well can we expect the spider to perform. If the generative model mimics the real web, we would expect our analysis to extend to the real world. This brings us to our second: given that a spidering stratagem appears to work on the real web, does it work on a generative model? If it does then it helps certify the model, otherwise it provides hints on improving the model. Finally the generative model can be used to give ideas on designing effective spidering stratagems. For example the generative model may be able to be used to predict the probability that a particular type of pages exists at the end of an unexplored link.

6.1.4 Thematically Unified Clusters for Spidering

In our experiments with TREC tasks on the WT10g corpus, we found evidence of an inverse correlation between the difference in target-to-target diameter versus target-to-all pages diameter and the difficulty in learning to spider. When the target-to-target diameter is high it indicates that the target set may not really be forming a thematically unified cluster (TUC).

It would useful to analyze the web sub-graphs containing each of the TREC target sets to examine how the distributions of structural properties for the sub-graph match that of the WT10g corpus and web as a whole. Next we would want to correlate these distributions against the difficulty in learning effective spiders.

In addition to experiments with WT10g corpus, it would be profitable to return to the task of exploring the live web. We might gather a set of candidate TUCs designed to explore their self similarity properties (with PageRank added) as well as to gather information on how these TUCs interconnect. We could leverage TUCs as follows:

1. Select a leaf node in Google's Directory, for example, "Society > Philosophy > Philosophers > Nietzsche, Friedrich".

2. Identify a keyword set identifying the leaf nodes topic.

3. Select the top n rated pages.

4. Spider the backlink pages to a depth d.

5. Check for new targets based on keyword set and move them to depth 1. Perform a shortest path analysis.

6. Separate TUCs:

   (a) entire crawl is a candidate TUC

   (b) the pages on the same contour from the target set is a candidate TUC

   (c) sample the children of pages in each TUC to generate new TUCs

These subgraphs would be tested for various properties such as power law distributions on PageRank, in-degree, out-degree, etc. as well as small world properties such as low diameter (in the strongly connected component) and high connectivity.

With our spidering experiments on the WT10g corpus, we built large training sets and tried to learn a regression on the navigational distance. The difficulty with this approach is that as the distance-to-target increases, the variation between pages increases and our ability to learn a signal decreases. Also the exponential increase in pages as the distance increases swamps out the few pages closer to targets. A better approach might be to partition the training graph into a set of clusters.

Understanding the properties of these subgraphs is important to efficient spidering. If these rings of nodes pointing towards a set of targets do indeed have the properties of TUCs and exhibit small world properties, then we can leverage this information to help us with the spidering problem. If we can identify the properties of particular TUCs then we can categorize newly discovered pages into their most probable TUCs and help guide our spider towards new target pages.

We can then use the clusters as a series of way-points to pass through to get to the target set. This may reduce the challenge of the spidering task. Spidering would become the process of navigation through a TUC to get to the next one, followed by the task of selecting the next TUC to try and find to move closer to targets. This may simplify the search problem as well as simplify the training problem. This is akin to Dill et al's [18] proposal that a *navigational backbone* exists between TUCs.

6.1.5 Markovian Models

As we discussed in the Introduction (Chapter 1, web search is an inherently non-Markovian process. The agent is never in the same state twice and can never earn a particular reward more than once. This is unfortunate as the myriad tool sets from the partially-observable Markov decision process and re-enforcement learning communities could provide powerful tools for attacking the spidering problem. In fact, as discussed earlier, Rennie and McCallum's [51] work on spidering using estimates of discounted reward with naive Bayes was a motivator for our approach.

There are several fundamental problems with treating web-spidering as a Markovian process:

1. Each action, i.e., following a link/retrieving a web page, is unique

2. Each reward, i.e., finding a target page, can only be visited once

3. No state, i.e., the web-graph with visited pages, unvisited pages, and discovered edges, can ever be reached again
4. The full state can only be partially observed

The problem of partial observability can be addressed, providing we solve the previous issues, through the use of partially observable Markov decision processes (POMDPs).

In the approach to spidering presented and explored in this thesis we manage a priority queue of all the known actions and try, at each time-step, to select the best action to attain our goal of discovering target pages. Ideally each action we take either earns a reward (discovers a target page) or increases the probability we will achieve rewards in future actions. Using our depth-heuristic this corresponds to trying to expanding our fringe towards the nearest target page.

It is this idea of the ever changing fringe, combined with the lack of repeatability of actions and achievement of rewards, that binds us to a non-Markovian view of spidering and makes spidering fundamentally different from some other exploration/planning problems. If we twist our view of the problem space to eliminate the concept of fringe we may be able to move to a much simpler Markovian model. How might we go about doing this?

The fundamental idea is that the Web is extremely large (billions of pages), and the number of actions we allow our spiders is very small (tens of thousands of pages). That is, from our spiders' vantage, the web is effectively infinite.

Now keeping this idea in mind, consider our web spider. At any particular point in time, there is some minimal distance, measured in directed edge traversals, between the fringe and the undiscovered target set. Lets imagine that we call this distance the state of our spider. Now if the state of our spider is this distance, what would the action be? In the fringe model, our action is following some high-priority link, but that doesn't work in this new model.

Let's hold-off for a moment on thinking of what the action is. Let's instead think of what the result of an action is in this model. There are three basic possibilities: the action (whatever it is) might reduce our distance; it might increase the distance; or it might leave us at the same distance. If we think back to our fringe-based spider, the action of retrieving a page would result in expanding our fringe. That expansion

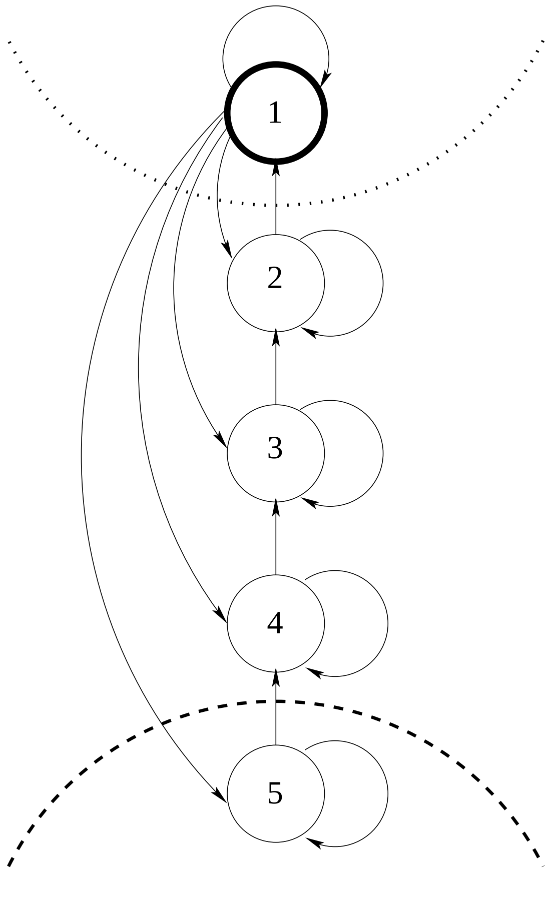


Figure 6.1: State machine abstraction of the web-spidering process. Each state encapsulates all of the "true" states in which the fringe is a particular distance from some nearest target.

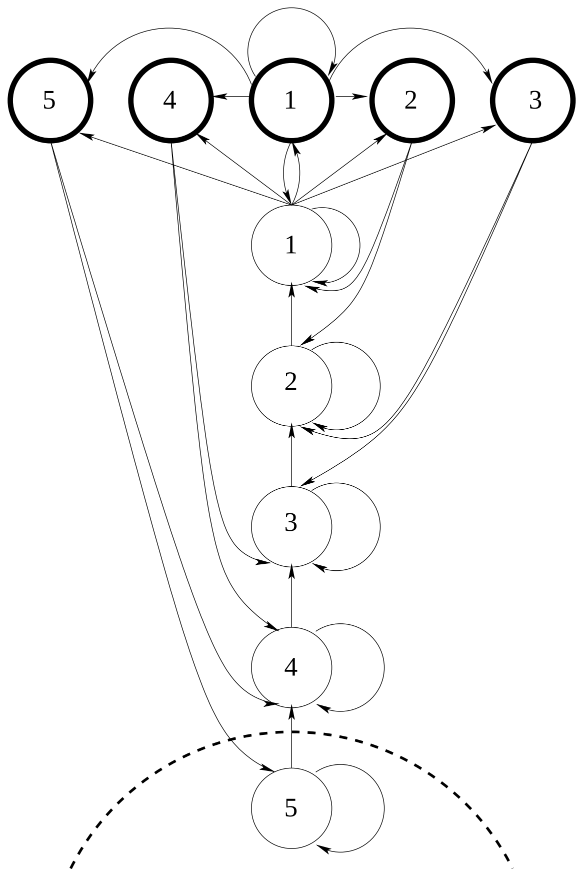


Figure 6.2: State machine abstraction of the web-spidering process with the reward state expanded.

would, at worse, bring us no closer to a target page, unless that retrieved page turned out to be a target. If it was a target then our distance to the nearest target could of course change.

Abstracting this back to our high level model, in all states, other than the one where we are at distance one from targets, the result of an action is to stay in place, or to reduce distance. If we are one link form a target, the next action could find that target and then we could be, potentially, in any state. If we aggregate all distances greater than some fixed value, we get a model like that in Figure 6.1. In this model we collapse all of the fringe states with nearest-target-distance of five or greater down to one state framed off with the dashed circle. The actual value to choose from is arbitrary but it is reasonable for it to be relatively small as the expected distance between the fringe and target will be low (driven by the low diameter of small world graphs).

In the discussion so far we have avoided discussing the abstraction of actions. If our action is the choice of which particular page to retrieve next, we will be back to the non-Markovian problem. Instead we need some abstraction of the choice of pages. Maybe instead of choosing a particular page, our action is to select the strategy of choosing the next page. For example we might choose to use the learning-to-order strategy or random choice strategy or a particular SVM machine strategy.

Now we can position the problem as learning problem where we need to learn the transition probabilities for each action at each distance. Since we are working in the approximate world of an infinite web-graph it is a reasonable assumption to pretend that there are also an infinite number of targets, which means it is reasonable to assume we can reach the reward state any number of times.

What exactly is the reward state? It is that state in which our action results in discovering a target. This is the bold circle in Figure 6.1. The actual structure of the model would be richer in the sense that one might need to disambiguate the target states to capture the different fringe-to-target distances that might occur after discovering a target (See Figure 6.2. The figure only shows the transitions between states, however each of these transitions occurs with some probability depending on the particular action chosen.

Also, note that we don't ever know what state we are in until and unless we reach a reward state. Once we discover a reward we know that at the previous step we were one away from discovering a reward.

With this model we can now treat our problem as a more classic learning problem in which our agent needs to learn the transition probabilities to the various states for each of the actions. Figure 6.3 captures this model in the form of a partially observable Markov decision process in which only the discovery of a target and some summary statistics are observable. Everything else is hidden or abstracted away.

This model has the potential to open the web-spidering problem to the full power of the tools developed by the reenforcement and Markov decision process communities. Some of these tools include efficient techniques for learning and representing policies as well as techniques for proving bounds on performance.

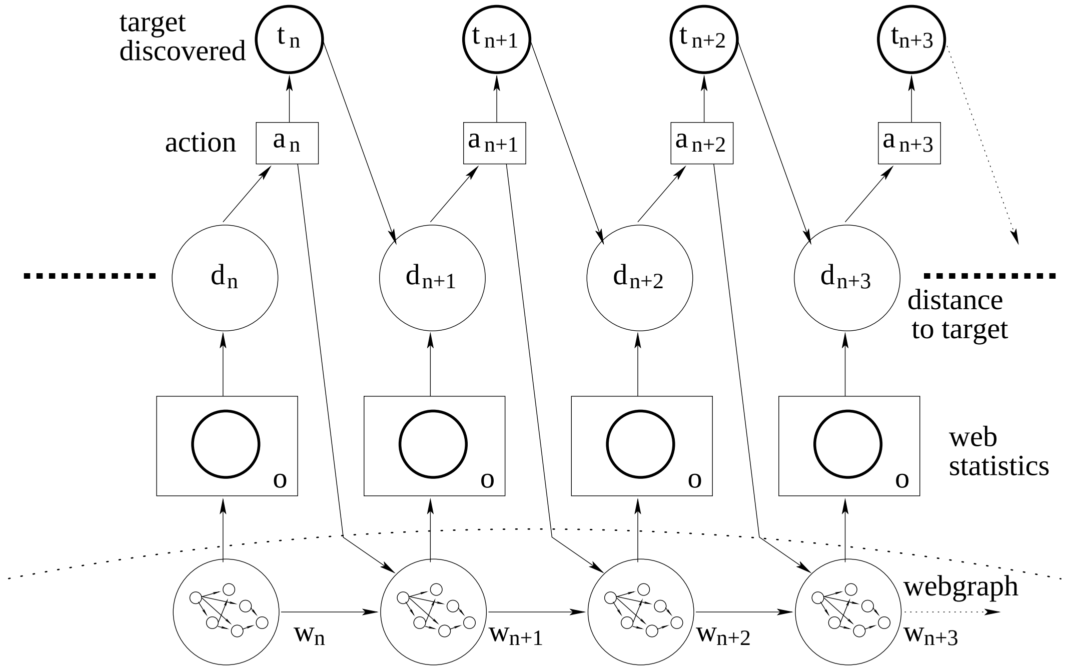


Figure 6.3: Partially Observable Markov Decision Process Abstraction. The actual state of the web-graph is abstracted away and isn't observable to the model.

6.1.6 Search Strategy Characterization

In Chapter 5, when analyzing the performance of different spidering strategies, we presented a serious of figures showing the distribution of page retrievals to first target, normalized by the size of the search space. See Figure 5.42 for an example.

These distributions consistently had different shapes for different search strategies, largely invariant to the task being explored. Also the shape difference in shape between the naive and trained spiders is dramatic. These differences suggest an opportunity for some theoretical exploration of the situation.

If we can develop a mathematical model explaining and predicting how the various distributional shapes emerge from spidering on the small world web-graph, we can use the model to predict performance of spiders. For example, we may be able to parameterize our model on the probability of picking an advantageous next page to explore from the fringe. Then given any learned regression, we could predict how well it will perform on a typical piece of the web-graph.

6.2 Summary

In this thesis, we have presented several major contributions to the understanding of the problem of search on the world-wide-web. We formalized the web-spidering hypothesis. We built an extensible infrastructure for future web research that has already seen application by other researchers. We explored the relationship between the naive search strategies of depth-first, breadth-first, and random searches and search strategies based on performing regressions against navigational discount and discounted reward. We successfully performed web-spidering tasks starting from large (for the web-spidering community) distances from targets. All of this was done in the context of a standard NIST approved web corpus allowing for repeatable experiments using data available to researchers worldwide.

Bibliography

[1] Lada A. Adamic. The Small World Web. In S. Abiteboul and A. M. Vercoustre, editors, *Proceedings of the 3rd European Conference for Research and Advanced Technology for Digital Libraries, ECDL*. Springer-Verlag, 1999.

[2] Reka Albert, Hawoong Jeong, and Albert-Laszlo Barabasi. The Diameter of the World-Wide Web. *Nature*, 401:130–131, 1999.

[3] Peter Bailey, Nick Craswell, and David Hawking. Engineering a multi-purpose test collection for Web retrieval experiments. *Information Processing and Management*, to appear.

[4] Albert-Laszlo Barabasi and Reka Albert. Emergence of Scaling in Random Networks. *Science*, 286:509–512, 1999.

[5] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet Allocation. In *Advances in Neural Information Processing Systems 14*. MIT Press, 2001.

[6] J. Boyan, D. Freitag, and T. Joachims. A Machine Learning Architecture for Optimizing Web Search Engines. In *Proceedings of the AAAI Workshop on Internet-Based Information Systems*, 1996.

[7] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[8] Andrei Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet Wiener. Graph Structure in the Web. In *Proceedings of the 9th International World Wide Web Conference*, pages 309–320, Amsterdam, The Netherlands, May 2000. Elsevier Science.

[9] S. Chakrabarti, K. Punera, and M. Subramanyam. Accelerated focused crawling through online relevance feedback. In *Proceedings of the 11th World Wide Web Conference*. ACM, 2002.

[10] Soumen Chakrabarti, Byron E. Dom, S. Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, Andrew Tomkins, David Gibson, and Jon M. Kleinberg. Mining the Web's Link Structure. *Computer*, 32(8):60–67, 1999.

[11] Junghoo Cho, Hector García-Molina, and Lawrence Page. Efficient Crawling Through URL Ordering. *Computer Networks and ISDN Systems*, 30(1–7):161–172, 1998. `http://citeseer.nj.nec.com/30730.html`.

[12] W.W. Cohen, R.E. Schapire, and Y. Singer. Learning to order things. *Journal of Artificial Intelligence Research*, 10:243–270, 1999.

[13] David Cohn and Huan Chang. Learning to Probabilistically Identify Authoritative Documents. In *Proceedings of the 17th International Conference on Machine Learning*, pages 167–174. Morgan Kaufmann, San Francisco, CA, 2000.

[14] David Cohn and Thomas Hofmann. The missing link - a probabilistic model of document content and hypertext connectivity. In *Advances in Neural Information Processing Systems 13*. MIT Press, 2001.

[15] Netscape Communication Corporation. Open Directory Project. `http://dmoz.org/`.

[16] CSIRO. TREC web corpus : WT10g. `http://www.ted.cmis.csiro.au/TRECWeb/wt10g.html`, 2001.

[17] Michelangelo Diligenti, Frans M. Coetzee, Steve Lawrence, C. Lee Giles, and Marco Gori. Focused Crawling using Context Graphs. In *26th International Conference on Very Large Databases*, pages 527–534, Cairo, Egypt, 10–14 September 2000.

[18] Stephen Dill, S. Ravi Kumar, Kevin S. McCurley, Sridhar Rajagopalan, D. Sivakumar, and Andrew Tomkins. Self-similarity in the Web. In *The VLDB Journal*, pages 69–78, 2001.

[19] Gary William Flake, Steve Lawrence, and C. Lee Giles. Efficient Identification of Web Communities. In *Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 150–160, Boston, MA, August 20–23 2000.

[20] Gary William Flake, Steve Lawrence, C. Lee Giles, and Frans Coetzee. Self-Organization of the Web and Identification of Communities. *IEEE Computer*, 35(3):66–71, 2002.

[21] Dayne Freitag and Andrew Kachites McCallum. Information Extraction with HMMs and Shrinkage. In *Proceedings of the AAAI-99 Workshop on Machine Learning for Information Extraction*, 1999.

[22] Y. Freund and R.E. Schapire. An efficient boosting algorithm for combining preferences. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[23] M.E. Lesk Gerard Salton. Computer evaluation of indexing and text processing. *Journal of the ACM*, 10(4):440–457, 1968.

[24] David Gibson, Jon M. Kleinberg, and Prabhakar Raghavan. Inferring Web Communities from Link Topology. In *Proceedings of the 9th ACM Conference on Hypertext and Hypermedia*, pages 225–234, Pittsburgh, Pennsylvania, June 1998.

[25] Taber H. Haveliwala. Efficient Computation of PageRank. Technical report, Stanford Digital Library Technologies Project, 1999.

[26] Taber H. Haveliwala. Topic-Sensistive PageRank. In *Proceedings of the 11th International World Wide Web Conference*, Honolulu, Hawaii, May 2002.

[27] Jun Hirai, Sriram Raghavan, Hector García-Molina, and Andreas Paepcke. WebBase: A repository of web pages. In *Proceedings of the Ninth World-Wide Web Conference*, 1999.

[28] Thomas Hofmann. Probabilistic Latent Semantic Analysis. In *Proceedings of the Fifteenth Annual Conference on Uncertainty in Artificial Intelligence*, Stockholm, 1999.

[29] Thomas Hofmann. Probabilistic Latent Semantic Indexing. In *Proceedings of the 22nd Annual ACM Conference on Research and Development in Information Retrieval*, pages 50–57, Berkeley, California, August 1999.

[30] T. Joachims. Making large-scale support vector machine learning practical. In A. Smola B. Schölkopf, C. Burges, editor, *Advances in Kernel Methods: Support Vector Learning*. MIT Press, Cambridge, MA, 1999.

[31] Leslie Pack Kaelbling, Michael L. Littman, and Andrew P. Moore. Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[32] Terence Kelley and Jeffrey Mogul. Aliasing on the World Wide Web: Prevalence and Performance Implications. In *Proceedings of the 11th International World Wide Web Conference*, Honolulu, Hawaii, May 2002.

[33] Jon M. Kleinberg. Authoritative Sources in a Hyperlinked Environment. *Journal of the ACM*, 46(5):604–632, 1999.

[34] Jon M. Kleinberg. The Small-World Phenomenon: An Algorithmic Perspective. In *Proceedings of the 32nd ACM Symposium on Theory of Computing*, 2000.

[35] Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew Tomkins. Trawling the Web for emerging cyber-communities. *Computer Networks*, 31(11–16):1481–1493, 1999.

[36] Jianhua Lin. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, 1991.

[37] N. Littlestone and M. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):221–261, 1994.

[38] Andrew Kachites McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. `http://www.cs.cmu.edu/~mccallum/bow`, 2002.

[39] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. A Machine Learning Approach to Building Domain-Specific Search Engines. In

The Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99), 1999.

[40] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Building domain-specific search engines with machine learning techniques. In *Proceedings of the AAAI-99 Spring Symposium on Intelligent Agents in Cyberspace*, 1999.

[41] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the Construction of Internet Portals with Machine Learning. *Information Retrieval*, 3(2):127–163, 2000.

[42] Filippo Menczer. ARACHNID: Adaptive Retrieval Agents Choosing Heuristic Neighborhoods for Information Discovery. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 227–235, 1997.

[43] Filippo Menczer. Links tell us about lexical and semantic Web content. Technical Report Computer Science Abstract CS.IR/0108004, University of Iowa, Department of Management Science, Iowa City, IA, August 2001.

[44] Stanley Milgram. The Small World Problem (1967). In S. Milgram, J. Sabini, and M. Silver, editors, *The Individual in a Social World: Essays and Experiments, 2nd Edition*. McGraw Hill, New York, NY, 1992.

[45] Marc Najork and Janet L. Wiener. Breadth-First Crawling Yields High-Quality Pages. In *Proceedings of the 10th International World Wide Web Conference*, pages 114–118, Hong Kong, 1–5 May 2001. Elsevier Science.

[46] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford Digital Library Technologies Project, 1998.

[47] Gopal Pandurangan, Prabhakara Raghavan, and Eli Upfal. Using PageRank to Characterize Web Structure. In *8th Annual International Computing and Combinatorics Conference (COCOON)*, 2002.

[48] Gautam Pant, Padmini Srinivasan, and Filippo Menczer. Exploration versus Exploitation in Topic Driven Crawlers. In *Proceedings of the Second International Workshop on Web Dynamics*, Honolulu, Hawaii, May 2002.

[49] C. Radhakrishna Rao. Diversity: Its measurement, decomposition, apportionment and analysis. *Sankhya: The Indian Journal of Statistics*, 44(A):1–22, 1982.

[50] Jason Rennie and Andrew Kachites McCallum. Efficient Web Spidering with Reinforcment Learning. Technical report, Just Research, 1999.

[51] Jason Rennie and Andrew Kachites McCallum. Using Reinforcement Learning to Spider the Web Efficiently. In *Proceedings of the 16th International Conference on Machine Learning*, pages 335–343. Morgan Kaufmann, San Francisco, CA, 1999.

[52] Mathew Richardson and Pedro Domingos. The Intelligent Surfer: Probabilistic Combination of Link and Content Information in PageRank. In *Advances in Neural Information Processing Systems 14*. MIT Press, 2002. `http://citeseer.nj.nec.com/article/richardson02intelligent.html`.

[53] Gerard Salton. *The SMART Retrieval System – Experiments in Automatic Document Processing*. Prentice Hall Inc., Englewood Cliffs, NJ, 1971.

[54] Jeremy Siek, Lie-Quan Lee, and Andrew Lumsdaine. Boost graph library. `http://www.boost.org/`, 2004.

[55] Ian Soboroff. Does WT10g look like the web? In *Proceedings of the 25th International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 423–424, 2002.

[56] Padmini Srinivasan, Gautam Pant, and Filippo Menczer. Target seeking crawlers and their topical performance. In *The 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Tampere, Finland, August 2002.

[57] The openMosix Project. openMosix. `http://openmosix.sourceforge.net`.

[58] Ellen Voorhees. Private Correspondence, TREC 2001 Web Ad Hoc qrels, Sep 2002.

[59] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, June 1998.

[60] Joel Young and Tom Dean. Exploiting locality in searching the Web. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, 2003. `http://www.cs.brown.edu/~tld/postscript/YoungandDeanUAI-03.pdf`.