Observation of changing information sources

A Thesis
Submitted to the Faculty
in partial fulfillment of the requirements for the
degree of

Doctor of Philosophy

by

Brian E. Brewington

Thayer School of Engineering
Dartmouth College
Hanover, New Hampshire

JUNE 2000

Examining Committee:

George V. Cybenko (chairman)

Robert S. Gray

Clayton M. Okino

Clifford S. Stein

Neal E. Young

Dean of Graduate Studies

(Signature of Author)

Thayer School of Engineering
Dartmouth College

"Observation of changing information sources "

Brian E. Brewington

Doctor of Philosophy

Prof. George V. Cybenko (Chair)
Prof. Robert S. Gray
Prof. Clayton M. Okino
Prof. Clifford S. Stein
Prof. Neal E. Young

ABSTRACT

Many modern information management tasks consist of an observer that must maintain current knowledge of a collection of changing information. The goal of this observer is to maintain acceptably accurate state estimates given limited observation resources, such as bandwidth, time, and storage. Good examples of such "observation problems" are found in any situation where bandwidth is limited and old observations become less useful over time. Two such examples are maintaining a search engine's index of the World Wide Web (WWW) and automated monitoring of multiple sensors. This thesis addresses the general observation problem by (1) devising a formal framework of what it means to be "up-to-date", (2) gathering empirical data about the web that allows us to apply this framework to an important setting, and (3) presenting algorithms for scheduling revisits to optimize formal performance measures. One year's worth of web page observations are analyzed to show how quickly and in what ways web pages change. The observations are used to estimate the distribution of web page change rates. Using this data as a model of the web we present and benchmark an algorithm for optimizing the observation of a set of web documents such that fewer changes go unnoticed. Our experiments on real search engines show that between 40 and 50% of results returned have been altered at least once since the search engine last observed them. Our algorithm can theoretically reduce these figures to between 36 and 44%, assuming that existing search engines currently use simple round-robin re-indexing strategies. These methods benefit the "overworked" observer much more than the one with ample bandwidth, and are general enough to be of use in a wide variety of monitoring tasks.

Acknowledgments

First and foremost, I thank God for His generous providence, for placing me in circumstances where I could explore exciting topics amongst intelligent people and great friends. God has granted me my health and has helped me to enjoy every step along this journey–there have been so many blessings over the last five years at Dartmouth.

First amongst the people I must thank is my wife, Flora, without whose patient love and support this thesis would have long since succumbed to my occasional bouts with discouragement. The prayers of my Bible study group and the encouragement of my parents and my brother have also helped me through the rough spots. I am indebted as well to my grandfather Ed Aldredge, who lives on in my memory as a role model and also as my inspiration in studying engineering.

I must express my appreciation for my advisor Professor George Cybenko, who exhibits a rare combination of exceptional brilliance and an uncanny knack for instilling the belief that your research does make an important and interesting contribution. The core ideas in this thesis are the product of our long but enjoyable hours of discussion and mutual inspiration. Thanks also go to the other members of my committee (Professors Robert Gray, Clayton Okino, Cliff Stein, and Neal Young) for putting in the time to review the work and guide it towards completion. A special thanks to Bob for somehow managing to be on my committee and yet remain a good friend, as well as to Professor Young for taking the additional time to travel from Boston up to Dartmouth for the defense. Thanks as well to Lee Giles for invaluable feedback in the early stages of the thesis.

The many good friends I've made over the years in the Thayer School "Lab for Total Global Domination" deserve thanks for their support, camaraderie, and feedback. In particular, thanks to Katsuhiro Moizumi for encouraging me and helping me along in the early steps; to Bob Gray and Aditya Bhasin for their many hours of work on the Informant; to Dan Bilar, Dan Burroughs, Shankar Sundaram, and Guofei Jiang for being my sounding boards for the latest wild idea; to Anush "King" Kumar and Jeanne Tania Sucharitaves for their help in coding some of the web page analysis scripts; to Michael Corr for valuable feedback and expert editing. Our work aside, there is no question that we had more fun in that lab than in the entire rest of the school combined. I cherish our peculiar combination of hard work and good humor. And thanks to Chris Schoeneman for creating `bzflag`, an enjoyable network tank game that has preserved my sanity on more than one occasion.

Last, I wish to express my appreciation for the financial support for this research, partially supported by AFOSR grant F49620-97-1-0382, DARPA grant F30602-98-2-0107 and NSF grant CCR-9813744. Naturally any opinions, findings, and conclusions are those of the author and do not necessarily reflect the views of the above agencies.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Just weeks prior to the Soviet invasion of Czechoslovakia, Corona satellite imagery of the area showed military buildup, but no signs of imminent attack. By the time another round of imagery was available, it was too late to react; the invasion had already taken place [Moo99]. In a real sense, the information obtained by the satellite weeks earlier was no longer useful. The fact that information has a useful lifetime is well known in the intelligence community.

On the other side of the Iron Curtain, an entirely different problem existed. The East German secret police, the Stasi, was especially vigilant in monitoring its own people–it is estimated that one of every three East Germans was a Stasi operative of some sort. Their “missions” were simple, mostly consisting of monitoring neighbors or other people with whom they had frequent contact. Information was gathered in copious quantities, ranging from diaries to odor samples, all neatly cataloged and stored for future use. The basements of the former Stasi headquarters are filled with reports, observations and gossip about who was and was not engaged in some sort of subversive activity. This is to say nothing of the records that were destroyed as the Stasi fled their headquarters. Overwhelmed by the sheer volume of information they held, the Stasi was probably unable to cull the useful or interesting tidbits from the dross.

No longer just the problem of the information-rich intelligence community, we all face the problem of high volume, dynamic information source organization and upkeep. Information overload, as it has come to be known, is exacerbated by a new challenge: the dynamics of information sources. The problem is the same for everything from a newspaper to a temperature sensor. When monitoring such a source of information, when can it be said that a previous observation has gone bad? How does this information translate into strategies for observation when only limited resources are available?

This is not a new problem, and there are some very well-established solutions. Biological sensory systems are always forming state estimates using limited observation capability in such a way as to balance uncertainty against the importance of things being observed. Moreover, biological sensory systems are extremely efficient, in that they rarely waste effort paying attention to slowly-changing or unimportant systems. Human vision, for example, is specifically tuned to discern movement within static fields: in this sense it is optimized for "change detection". Moreover, odors and noise cease to grab one's attention after they have been sensed for some short time. Again, a sort of "novelty filter" exists to optimize attention for change detection. Consider where visual focus is directed when driving: the road ahead is the source of the most important information, as well as the source which changes most quickly, thus the majority of focus is placed there. Changes in the state of other items, such as the rear-view mirror image, the vehicle speed, the radio reception, and the fuel remaining are deemed either less important or more slowly varying, and are therefore given less frequent visual attention. Buried somewhere deep within our brains is the capability to assess what is worth our attention, and how much attention it deserves.

As information sources become more numerous and more dynamic, the cost of keeping up increases. This thesis investigates the balancing act faced when monitoring a collection of dynamic information sources. We discuss one domain (the World Wide Web) at length, considering how fast its documents change and what it means for a web index to stay current. For simple classes of web monitoring systems, our idea of "up-to-dateness" is combined with data to see how a theoretical web index might perform. This is augmented by experimentation to assess how well modern search engines perform in this regard, as well as by data on the ways in which web pages change. We conclude with algorithms for implementing a practical system for optimizing observation. Though much of the discussion centers on the web, monitoring problems are also of more general interest. Indeed, the central problem of the information age is the management and indexing of large, dynamic volumes of information, whatever those may be.

1.1 Information as a depreciating commodity

Any information has a useful lifetime: knowledge is a depreciating commodity. Buying a car is a good analogy, since the car begins to lose value as soon as it is driven off the lot. In the information domain, there is an expenditure of computational resources, such as bandwidth, storage, and computation cycles, to obtain momentarily current knowledge.

This "purchase" of information has two defining characteristics: first, there is an initial value for correctness, and second, there is a rate of depreciation. The value of having certainty can probably best be seen in the consequences of being wrong. Compare, for instance, knowing whether there is a bus coming down the street you wish to cross, versus knowing what the latest fad collectibles fetch on the open market. The second part of an information purchase is how long the information is expected to last. When do we next expect that we need to look left and right *again* to check for oncoming traffic? Simply put, when the uncertainty about a previous observation becomes intolerable (the meaning of which depends upon context), it is desirable to observe again. The idea of assigning a value to a reduction in uncertainty was formally explored in [CBB+96], providing much of the inspiration for this work. Present worth and rate of deterioration are both equally important in deciding upon a purchase of information.

As an alternative to the commodity viewpoint, we can also see information as a consumable supply, like paper or gasoline. Namely, acquired information is "used up" after some time and must be re-acquired. Inherent in this idea is that there is some expected long-term cost to keeping a current knowledge of any single source. Given a large number of dynamic sources to index, and only limited resources (bandwidth, computation and storage) available to do so, a decision must be made as to what should be observed, and when these observations should be made.

Here is a contrived example to give a feel for the tradeoffs involved. A source of information that changes daily (say, a newspaper) requires approximately daily observation if one wishes to stay current. We defer discussion of precisely what it means to be current until later. Clearly, observing a daily-changing source consumes about 30 times the bandwidth as would be necessary to watch a source that only changes monthly (but only 1/30 of the storage, if the sources are all of equal size).

For an index of sources in which any correct index entry is equally valuable, it would seem that it is best not to try to keep up with the fastest changing sources. Consider these alternatives: we could either (1) observe the single fast-changing source daily, or (2) observe 30 different slowly-changing sources. If all these sources are of equal value, there is little reason to observe the daily source. The strategy of observing the daily source would leave us with an index in which 30 out of 31 entries were out of date, while the alternative would have a much higher percentage correct. Resources are spent to literally buy time, specifically, the time until the next observation is required. The daily source will need to be observed each day–this is how long the purchase of index correctness will last. An observation of the monthly source will not expire for 30 times that time period.

But there is a catch. What if, due to popularity or demand, it is more important to know what is going on in a fast-changing source? Returning briefly to the analogy of indexing information to buying automobiles, a car's value is not just that it is a car, but that it provides transportation to individuals. The same is true for a school bus, or a subway. The higher cost of maintaining (and acquiring) mass transit systems as an alternative to cars is justified by the fact that their utility is multiplied by many users. The right metric for this system should account for how many individuals can be moved reliably from point A to point B, not just whether or not it is a functional piece of machinery. By analogy, monitoring "high-maintenance" information sources can be partly justified by the scale of the benefits.

1.2 Observing the World Wide Web

Since its inception scarcely a decade ago, the World Wide Web has become a popular vehicle (so to speak) for disseminating scientific, commercial and personal information. The web consists of individual pages linked to and from other pages through Hyper Text Markup Language (HTML) constructs. The web is patently decentralized. Web pages are created, maintained and modified at random times by thousands, perhaps millions, of users around the world.

The task of maintaining an index of the web has been taken on by the search engine, playing a role loosely equivalent to that of traditional library catalogs. However, a single edition of a book or magazine does not change once it is published, whereas web pages often do. Therefore, web search engines must occasionally re-visit pages and re-index them to stay current. This is an enormous challenge considering that recent empirical studies by Lawrence and Giles [LG99] have estimated the size of the publicly-indexable web to be at least 800 million pages (and climbing). The size of the web is only one factor in the re-indexing problem; the rate at which pages change is equally important.

Re-indexing the web is only one example of an observation problem; our web analysis is built upon a formalization of the concepts used for measuring the "up-to-dateness" of any index of changing information sources. This includes a discussion of different models of information source dynamics. From this, we move into performance metrics for gauging how current an index of information sources is. Beginning with simple notions of probability of entries being out-of-date, we lead into a novel concept, the idea of (α, β)-currency. This defines our notion of being up-to-date as a probability, α, that a search engine is current, relative to a grace period, β, for a randomly selected web page. Loosely speaking, an index entry for a given object is said to be β-current if the object under observation has not changed between the last time it was indexed and β time units ago. In this context, β is the "grace period" during which changes are allowed to go unobserved. In the context of the web, a search engine for a collection of web pages is then said to be (α, β)-current if a randomly (according to some specified probability distribution) chosen page in the collection has a search engine entry that is β-current with probability at least α. We develop an exponential probabilistic model for the times between individual web page changes, as well as a model for the distribution of the change rates defining those exponential distributions. Lastly, this model is applied in the context of devising methods for practical optimization of real observation systems.

Our observational data is based on statistics gathered from roughly three million web pages specified by over 30,000 users of a web clipping service [Inf95]. We have observed pages at a rate of about 100,000 pages per day, for a period of approximately one year, recording how and when these pages have changed. The data indicate that the time between modifications of a typical web page can be modeled by an exponential distribution, which is parameterized by the rate of changes for the page. Our data further indicate that the reciprocal of that parameter, which is the expected time between changes, is well-modeled by a Weibull distribution across pages.

To get an intuitive feeling for the concept of (α, β)-currency, consider some examples. The numbers are purely speculative. We might say that a daily newspaper is (0.90, 1 day)-current when it is printed, meaning that the newspaper has at least 0.9 probability of containing 1 day current information on topics of interest to its readers (this reader interest is the specified probability distribution). Here "1 day current" means that events that have happened within the last day (which is the grace period) are not expected to be reported and we "forgive" the newspaper for not reporting them. Similarly, hourly television news would be (0.95, 1 hour)-current and so on. The idea is that we are willing to "forgive" an index or source if it is not completely up-to-date with respect to the grace period. The grace period is a domain-specific, tolerable time lag between a change in an information source and when that change should appear in an index. Any index of information sources has a spectrum of possibilities, such that for any grace period β there exists a probability α of being current with respect to that period.

Our empirical analysis of web page changes is combined with existing estimates of the web's size to estimate how many pages a search engine must re-index daily to maintain (α, β)-currency of the entire indexable web, if all web pages in the collection are reindexed with the same time between observations. Using 800 million documents [LG99] as the size of the web, we show that a (0.95, 1 week)-current search engine must download and index at a rate of around 50 megabits/second (using an average page size of approximately 12 kilobytes, as was true of our data, and assuming uniform processing). A (0.95, 1 day)-current search engine, by comparison, must re-index at the rate of 104 megabits/second. These figures are based upon our model of web page change rates, but these results allow estimation of re-indexing rates in order to maintain general (α, β)-currency of a web index using other models as well.

These estimates of re-indexing rates assume that there is a single re-indexing period for all documents. As was made clear in the example of reading periodicals, it may be advantageous for a search engine to pay more attention to items that change quickly, are popular, or both. Given models of object dynamics and popularity, what observation schedule maximizes the probability of being β-current? This question is posed in both discrete and continuous time frameworks subject to different modeling assumptions. In discrete time, the problem is shown to be equivalent to a minimum-cost flow problem, in which objects are mapped to strategies in such a way as to minimize cost. The continuous time problem, in which we assume that objects change according to independent Poisson processes, consists of forming an optimal partition of the space of monitored objects and dividing available bandwidth among the partitions. We will see that the discrete-time framework allows for better modeling precision, while the continuous time formulation is more practical to apply for large monitoring systems. We will also show how a typical search engine could expect between 5 and 10 percent performance improvement if these optimizations are applied.

1.3 Related work

Much of the optimization work hinges upon identifying rates of change, or values, of objects by repeated observation. These observations are then used to formulate schedules based on estimates. The problem of modeling an unknown distribution in the context of possible rewards is an old problem, addressed in classic texts on probability theory ([Fel68], [Fel71]), and dynamic programming ([Ber87], [How60]). Linear programming [Lue84] and combinatorial optimization [Ber98] are often used in a formal solution. Our continuous-time optimization routine implements a simulated annealing algorithm, which is popular in neural networks [Hay94]. We use Nelder-Mead optimization for finding minimum variance estimates to the distribution of change rates for web pages. This method is useful for unconstrained local optimization of scalar functions of a vector argument; [PFTV92] contains an excellent description and code for implementation.

The re-indexing scenario may be thought of as a very large adaptive sampling procedure, or "multi-armed bandit problem" (see [HS98]), in which the cost of making an observation is balanced against the possible benefits obtained by viewing the document. Given the size of the indexable web (320 million pages [LG98] in 1998, and 800 million pages [LG99] in 1999), it is probably not tractable to treat the search engine's problem as a strict multi-armed bandit approach, although there are some useful tie-ins. One important difference between the search engine index problem and the typical multi-armed bandit problem is that the objective is to make the best index possible, rather than just to make good single index entries. So whereas a hypothetical player would happily play over and over on a bandit that always returns winnings, the search engine cannot spend all of its time re-indexing only the pages that change very fast. Smaller problems might be nonetheless be treatable by this approach.

In modeling the dynamics of web documents, we find that there is significant overlap between the problems of designing a current index and a good web cache. As mentioned in [DFKM97], good cache design depends upon knowledge of both resource dynamics and client access patterns. Resource dynamics are investigated in [DB96], and client access patterns are studied in almost all papers on cache consistency (usually under the guise of a hit-ratio). The index maintained by an engine can be viewed as a large summary cache that is required to preempt the requests of its users. The web index need not require strong consistency, as only document summaries are presented.

Like a cache, an index may benefit from knowing the expiration time of a document in order to determine when it should be viewed again. Of particular interest is work into adaptive time-to-live (TTL) settings, such as in the Alex system [Cat92] and the Harvest [BDH+94] system (also [CDN+96]), where a server adjusts the TTL setting to a percentage of the age of a document at the time it is requested. This is a fairly simple model, based on the reasonable assumption that an old document is less likely to change in the near future. Our work is an expansion of this principle. We develop probabilistic models and establish precise relations between the probability density of observed age and the probability density of observed time between changes. In addition, we consider how the age distribution depends upon the the time of creation of an object. Not surprisingly, observed ages in a growing population tend to be shifted towards zero age. We discuss why a growth model is necessary to use age distributions for estimating distributions of change rates in an index.

Relatively little work has focused on combining document models with search engine scheduling; Coffman et. al. [CLW97] studied the search engine scheduling problem under the assumption of Poisson page change processes. Their work showed that if service times follow the same distribution for all documents, then each document should be re-indexed as regularly as possible, which is consistent with the memoryless properties of the underlying exponential distributions. For a particular cost function, they show that each document's optimal sampling frequency is determined by the decay constant for that document's change distribution, giving the time between changes. Our measures of performance are similar in spirit, but introduce a temporal and probabilistic relaxation of what it means to be up-to-date, namely the concept of (α, β)-currency. We feel that this relaxation is useful since it gives a better match to intuitive notions of currency. We further extend their work by making use of resource popularity as well as change rate in our optimization routine. We feel that these are necessary and important modifications. For example, the idea of (α, β) currency allows us to cast [CLW97] as an algorithm for maximizing a measure of currency while keeping $\beta = 0$, but we feel there are many monitoring applications where a non-zero grace period is more appropriate.

How search engines actually prioritize observations tends to be proprietary. It is likely that engines do at least prune their indexes to eliminate redundant or useless content, however this is determined. This does constitute a sort of optimization of the re-indexing process, since less redundancy in a collection translates into more bandwidth available for re-indexing. There are a few cases in which these designs have been discussed in the literature, specifically the Google search engine papers ([BP98], [JCP98]) as well as other crawler designs useful for generating topic-specific indices [CvdBD99] and general-purpose crawlers [HN99]. In [BP98] we find a good exposition of the practical issues facing a large scale search engine, such as a high-level layout of search engine architecture. They also discuss the PageRank metric, which is a rough measure of how often the URL is linked to by higher ranking sites. PageRank (or other similar measures) could provide a good means for partitioning a space of monitored objects according to popularity. They also indicate that it is feasible to crawl and index on the order of 10 million pages per day. This figure suggests where a modern search engine should fall within the specturm of (α, β)-currency. It is important to note that [BP98] lists change-rate based scheduling as an area for future work, as does [JCP98]. In [JCP98] we find a strategy for crawling pages expected to have high PageRank. This is similar to the results of our observation optimization routines, which clearly show how some part of a collection should be given preference in the re-indexing process. Our optimization lends mathematical justification to the preferential crawling idea. A good summary of what is known about other commercial engines is provided in [Nic97], although the paper's data on re-indexing rates may be inaccurate since the work is now a few years old, and the technology involved is evolving rapidly.

As of this writing, the only commercial search engine which actively claimed to do any sort of preferential examination of documents based upon their age was Infoseek, in their Ultraseek intranet search engine [Inf99]. Though the details of their system are proprietary, the system is built upon placing all documents in the collection in one of a number of fixed-rate queues, the rates of which are logarithmically distributed between a minimum and maximum polling rate. Within each queue, documents are examined based on an least-recently-used (LRU; also referred to as "round-robin") protocol. All parameters (number of queues, min/max polling rates) are user-defined. Over time, documents are placed in the queue that seems to best match their change rate. The idea is to evenly distribute wasted observations across the collection. The nature of the model used to determine queue placement is unspecified, but if applied correctly the model can give significantly better index freshness as compared to a uniform, single-queue LRU policy. However, leaving parameter choices to users (rather than tuning them automatically) can easily result in suboptimal performance. We investigate the theoretical performance of a similar system and attempt to optimize its performance. By varying the rate at which each sub-queue operates, as well as the boundaries between queues, some gains in performance can be attained.

Some important factors in gauging the scale of the problem faced by web observers are the size of the web, as estimated by [LG98] and [LG99]. The first paper uses the statistical overlap in search engine indices to estimate web size by looking at the common web pages contained within different commercial search engines. It is based upon a sample of 575 queries made by employees of the NEC Research Institute. By assuming the search engines index independent sections of the web, the overlap between search engine responses allows estimation of the size of the web. The second paper actively sampled random IP addresses for web servers, then applied hueristics to remove servers not part of the so-called "publicly-indexable web". An exhaustive crawl was then done within each of these sites. counted documents, then extrapolated to estimate the total size of the web.

Additional web size estimates are available from [Pit98], which is an extensive literature summary detailing many web characterization efforts prior to 1998, also including mention of various Zipf-type [Zip49] power-law distributions found on the web. More recent work showing power law distributions within in-degree distributions can be found in [BKM$^+$00]. This work might be especially helpful in estimating the distribution of site popularity, which is an important factor to consider in designing good algorithms for keeping up with the more important sites on the web.

The oft-cited figure for web pages changing every 75 days comes from [Kah97]; it was unclear from the article how a document was defined. Namely, if a document changes 10 times, does it contribute 10 samples to the average, or only one? Our estimates are formed such that each URL contributes equally to the page lifetime distribution; we expect that our figures would be much shifted to a lower range if we had one contribution to the distribution for each separate instance of a web page. Kahle's article does have some insight into the web's utility as a tool for providing references in scientific papers.

1.4 Summary of major results

Chapter 2 presents the theory behind assigning a probability of change to monitored objects in a collection, as well as performance metrics for gauging how "up-to-date" an index is. Both discrete and continuous time frameworks are discussed. The discrete time model defines the state of a monitored object, with respect to change detection, as the (integer) number of time units since the last change occurred. In conjunction with a matrix of transition probabilities, we model the probability of a monitored object having changed since the last observation and thereby are able to assess how current an index will be for a given observation strategy. We then make a transition to continuous time, treating an object's changes as a renewal process. The concepts of lifetime and age are extended to continuous time. Index performance metrics derived from these models are introduced for both discrete and continuous time monitoring systems. From simple metrics of the expected number of index entries that are not current, we move to a more formal definition of (α, β)-currency than that given earlier in the introduction.

Following the theoretical development, Chapter 3 details the web page data that has been collected by the Informant[1] over the past year. The Informant is a web clipping service run at Dartmouth in support of over 30,000 users from around the world, downloading around 100,000 web pages per day on their behalf. We give an overview of the kinds of data retained in our observation database, as well as the sampling methods (and associated biases) used to obtain this collection.

[1] http://informant.dartmouth.edu/

Within the data, we look at change modalities, and the distribution of page ages and page lifetimes. By representing a document as a term vector, in which individual components correspond to words in a document, we present data on how the words in a sample of documents evolves in time. This is a step towards normalizing the meaning of "change" for a web page. The latter half of the chapter assumes a simple, memoryless page change model and uses observed age and lifetime data to estimate the distribution of mean change times for the web pages in our collection. This distribution of change rates is then applied to show how current we expect a search engine to be when using a basic round-robin re-indexing policy, given different processing rates. We compare this to experimentally derived values for four commercial search engines.

Chapter 4 presents both discrete and continuous time methods of optimizing re-indexing strategies, not necessarily restricted to the web. The discrete time algorithm poses the problem by assigning objects to observation strategies that have different costs so as to keep a fixed number of observations per time unit. The discrete-time algorithm's complexity (exponential in the number of time units) inspires a simplified approach based on partitioning the space of observed objects according to popularity and change rate. In this scheme the optimization consists of forming a partition of the object space, then allocating rate to each bin in the partition. Through our tests of the second algorithm, we show that observers whose bandwidth is very limited stand to gain much more than those with ample bandwidth. As expected, these performance gains come at the expense of the least popular, fastest changing objects. We also show the benefits that can be expected when the size of a collection of monitored objects is reduced by eliminating redundancy. The thesis concludes with a summary and a number of suggestions for future work. An appendix contains some interesting power-law distributions we ran across in studying our web page data.

Chapter 2

Information monitoring theory

To begin our investigation of information source monitoring, we review the theory behind the discrete-time models introduced in the thesis proposal [Bre98]. The discrete-time approach allows for very general models of how an information source can change. However, in some circumstances, more simplistic models are required, so we show how the discrete-time models carry over into more restrictive continuous time representations. Both the discrete and continuous time models focus on defining the probability of an object change as a function of the time since the last change occurred. In discrete time, this dependence is expressed using a Markov process, while the continuous time version models changing objects as a renewal process. The chapter concludes with ways to measure the performance of an index of changing sources, for both continuous and discrete time settings. This theory provides the foundation for both the empirical studies of the web in Chapter 3 as well as the observation optimization routines presented in Chapter 4.

2.1 Modeling changes in observed sources

The goal of our object change models is to estimate the distribution of time between changes, or lifetime, both for single objects and for entire populations. Our approaches all assume that there is some way to estimate the probability of change over a given timespan for any object under observation. This probability depends upon both what constitutes an object's state, as well as how much of a movement in this state is to be considered a "change". How the state is defined is domain specific, as is the associated definition of a change to that state. When watching stock prices, a

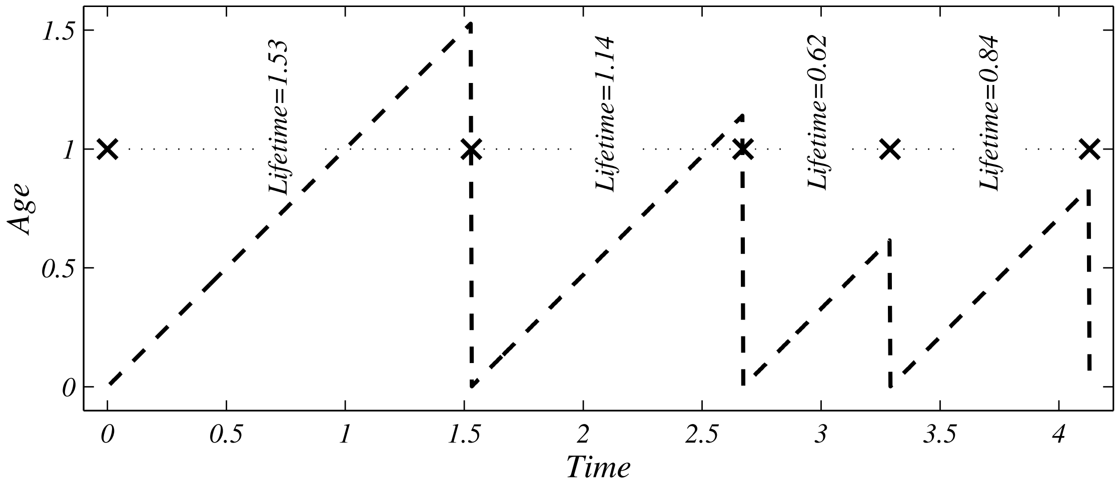


Figure 2.1: **Lifetimes vs. ages.** A single lifetime is represented by the time separating changes, which are denoted by $\times$'s in the graph. For each lifetime, the age (shown as a dashed line) increases linearly from 0 to the lifetime, then resets to 0 as the next lifetime begins.

change might be defined as a change in price of more than 3%, or perhaps a long term average decline. A change on a web page could be defined as anything from changing a single character to some discernible alteration in topic. Still, what is most common to all these applications is the dependence of the probability of change upon the elapsed time since the last change.

A good analogy for the sequence of object changes is a system of replacement parts. Imagine a light fixture into which we place a lightbulb. Whenever that bulb burns out, it is replaced immediately. We speak of the time between lightbulb failures as the "lifetime" of a bulb. At a specific instant, we define the time since the present lifetime began to be the "age" of the bulb. The analogy to an observed object is that the lifetime is the time between changes (where change is arbitrarily but unambiguously defined). The age is the time between a given instant and the most recent change prior to that instant. We diagram these concepts in continuous time in Figure 2.1; there are discrete time versions of these concepts as well.

2.1.1 Time-since-modification as a Markov chain

We begin our modeling efforts by modeling an object's state using only its age: the state of an object is the time since it was last changed. The age is assumed integer in the discrete time case. While other factors will undoubtedly influence an object's modification, we begin by assuming that the probability of a change is completely determined by the age. This could (and will) be treated as a continuous state, but quantizing will allow us to develop a feel for how these types of models perform. Any reasonable discretization of this time will serve our purpose, so long as the probability of a change occurring is relatively constant within that period. Using this definition of state, a probability distribution of the time intervals between changes can be represented as a Markov chain. This is an oft-used means for describing time-dependent recurrent events; see [Fel68] for example. Given a state $s = n$ time units since the last change, there are only two things that can occur. One possibility is that the object will not change, and the state will advance to age $s = n + 1$. Alternatively a change will occur, and the age will reset to state $s = 0$. If we model $N + 1$ states, then the state $s = N + 1$ can be treated as "N or more days since last change." In this way, we can define a matrix of transition probabilities as

$$\mathbf{M} = \begin{bmatrix} p_{reset}(0) & 1 - p_{reset}(0) & 0 & \cdots \\ p_{reset}(1) & 0 & 1 - p_{reset}(1) & \cdots \\ \vdots & \vdots & \ddots & \vdots \\ p_{reset}(N) & 0 & \cdots & 1 - p_{reset}(N) \end{bmatrix} \tag{2.1}$$

Whenever $p_{reset}(t) \neq 1 \ \forall t \in [0, N]$, all states of $\mathbf{M}$ are recurrent. The function $p_{reset}(t)$ can be determined either from the distribution of time intervals between changes, or from the distribution of observed ages for a particular page. By finding either distribution by estimation from samples, and then applying this model for all future time, we assume stationarity of the underlying change dynamics. Both the age and lifetime distributions (whether or not they can be determined from samples) can be used to calculate the conditional probability that a reset will occur in the following time interval, given that no change has occurred for t time steps. When the time between changes (lifetime) is distributed according to a discrete probability mass function $f(t)$, then

$$p_{reset}(t) = \frac{f(t)}{1 - \sum_{k=0}^{t-1} f(k)} \tag{2.2}$$

In modeling $\mathbf{M}$ for an object, we will only be able to define $f(t)$ for values up to some $t = N$. At the upper age limit, the definition for p_{reset} breaks down, since $p_{reset}(N)$ would be unity. This would imply that if the state ever reaches age N, then it will change with probability 1 within the next time step. In reality, though, there will generally be some chance that the state will remain "N or more time intervals since last change," or $s_i = N$. One way to represent this possibility is to assume that $p_{reset}(N) = p_{reset}(N-1)$, or that the reset times are memoryless after a certain time has elapsed.

Sometimes, it is easier to observe ages rather than change intervals (lifetimes). Attempting to observe change intervals directly can cause difficulties with aliasing, as an observation will generally only indicate the most recent change. Web pages, for example, often have a `Last-Modified` date in the `HTTP` header, allowing the age of the document to be sampled at any time. Over many samples, one can build up an estimate of the probability distribution $g(t)$ of observed ages for such documents. Moving from an age distribution $g(t)$ to p_{reset} is straightforward. As with the function $f(t)$ defined for (2.2), we express the age t of an object in some arbitrary units of time. The two probabilities that we wish to determine for age t are the probability of aging to $t+1$ and the probability of being reset to age $t = 0$. That is, some fraction of the objects "dies" (changes) before reaching the next age. This fraction, which is $p_{reset}(t)$, is just the percent difference between successive elements of the age distribution:

$$p_{reset}(t) = \frac{g(t) - g(t+1)}{g(t)} \tag{2.3}$$

Notice that $g(t)$ must be monotonically decreasing.

To get a feeling for how the matrix $\mathbf{M}$ corresponds to different distributions of time-between-changes, we present some simple examples. If an object changes on a purely periodic basis, say every 3rd time unit, then the matrix $\mathbf{M}$ will be

$$
\mathbf{M} = \begin{bmatrix} 0 & 1 & 0 & \cdots \\ 0 & 0 & 1 & \cdots \\ 1 & 0 & 0 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{2.4}
$$

If the system changes according to a Poisson process, then it is memoryless, so the reset probability is independent of state:

$$
\mathbf{M} = \begin{bmatrix} \rho & 1-\rho & 0 & \cdots \\ \rho & 0 & 1-\rho & \cdots \\ \vdots & \vdots & \ddots & \vdots \\ \rho & \cdots & 0 & 1-\rho \end{bmatrix} \tag{2.5}
$$

We emphasize that this particular definition of state is not the only useful one; other models can emphasize different aspects of the problem. If we were interested only in capturing the fact that one or more changes had occurred, a model could be used that is similar to the one used in the classic machine replacement problem [How60]. This model would be similar to $\mathbf{M}$, except that it would not keep track of the age after a page change had occurred. The system would begin in a transient state, "unchanged, with age i", and when a change occurs, the system would enter an absorbing state, "changed, with unknown age". Observation would force a reset to the transient state having the appropriate age.

Formally, we define the ij^{th} element of $\mathbf{M}$ to be the probability that an object of age i will become age j on the following time step. More generally, by raising $\mathbf{M}$ to an integer power k, we can find the probability (for initial age i) that the object is age j after k time steps have elapsed since the last observation:

$$
\text{Prob}\left(s_{t+k} = j | s_t = i\right) = \left[\mathbf{M}^k\right]_{ij} \tag{2.6}
$$

As k grows without bound, the ij^{th} element of $\mathbf{M}^k$ approaches the stationary transition probability π_j, which is the probability that the page will be observed to be age j if we are given no knowledge of the history of the page. This is precisely the age distribution mentioned above, $g(j)$:

$$
\lim_{k\to\infty} \left[\mathbf{M}^k\right]_{ij} = \pi_j = g\left(j\right) \tag{2.7}
$$

2.1.2 Continuous time, discrete age models

Just as the Markov model allows us to represent the probability of occupying any integer age after some number of discrete time steps, it permits an alternative interpretation as the probability distribution of ages within a population of objects having the same starting point. In the same way, we can model the variation in this distribution in continuous time. This evolving distribution can itself be considered the state of a single entity (analogous to quantum mechanics), or it can be interpreted as an age distribution over many individuals within a population. Using either interpretation, this class of models uses discrete ages, but models the variation of this distribution in continuous time. Define the value S_i of a state i describes what portion of the population has a particular discrete age i. When normalized, the state vector $\mathbf{S}$ can be interpreted as a probability distribution. Of course, allowing the values of S_i to be non-integer does not permit an interpretation as "exact number of pages having age i". This is a step towards having a model which is continuous in both the progression of time and the measurement of age. The model can be used to examine how age distributions like (2.7) behave when a population is growing exponentially.

The aging of objects in this system can be described by a finite system of differential equations, with one equation describing the rate of change of every state. Each state variable is analogous to the probability of occupying the corresponding state in (2.1). In the case of continuously-varying state, the rate of change of S_i is the rate of transition out of state $i-1$ into state i, minus the rate of transition out of state i into either the base state, 0, or into the state one age ahead, $i+1$. Now if all objects occupying state $i-1$ are identical, then outbound transition rates from $i-1$ to i are proportional to occupancy of the state, S_{i-1}. This is true for those departing to state $i+1$ and also for those returning to state 0. We collect these ideas together in a single equation, which expresses the rate of change of the occupancy S_i as

$$\frac{dS_i}{dt} = C_{i-1}S_{i-1} - (C_i + R_i)S_i. \tag{2.8}$$

Each C_i gives the rate at which age i entities progress to age $i+1$. The constants R_i describe the rate at which age i entities return to age 0. If a uniform spacing of the ages i is desired, then the sum $C_i + R_i$ should be independent of i. Alternatively, a nonuniform spacing could have a rate sum

$C_i + R_i$ that is altered to match the intended width of the age bin i. The sum $C_i + R_i$ is an indirect measure of the relative rate of the progression of time (fixed) and the progression from state i to possible target states 0 and $i+1$.

For exponential growth of such a population, the base state 0 receives a continuous infusion in proportion to the total occupancy of all states, in addition to the increase due to the start of new lifetimes for all states. Thus we have that

$$\frac{dS_0}{dt} = -C_0 S_0 + K \sum_{i=0}^{N} S_i + \sum_{i=1}^{i=N} R_i S_i. \tag{2.9}$$

Here K is the growth rate of the population (for exponential growth). If the population is neither growing nor shrinking, then $K = 0$. Combining (2.8) and (2.9), the system of rate equations becomes

$$\begin{bmatrix} \frac{dS_0}{dt} \\ \frac{dS_1}{dt} \\ \frac{dS_2}{dt} \\ \vdots \\ \frac{dS_N}{dt} \end{bmatrix} = \begin{bmatrix} -C_0 + K & R_1 + K & R_2 + K & \cdots & R_N + K \\ C_0 & -C_1 - R_1 & 0 & \cdots & 0 \\ 0 & C_1 & -C_2 - R_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \cdots & \vdots \\ 0 & 0 & \cdots & C_{N-1} & -C_N - R_N \end{bmatrix} \begin{bmatrix} S_0 \\ S_1 \\ S_2 \\ \vdots \\ S_N \end{bmatrix} \tag{2.10}$$

We can check the validity of our rate equations by requiring that any choice of K, C_i, R_i satisfy a conservation law; we should have that

$$\frac{d}{dt} \sum_{i=0}^{N} S_i = K \sum_{i=0}^{N} S_i \tag{2.11}$$

This forces zero population growth when $K = 0$. Adding all the equations in the system (2.10) gives the conservation relation (2.11) as an identity, so any set that has the form (2.10) is conservative.

Solutions to (2.10) are not difficult in principle; any equation of the form

$$\mathbf{S}'(t) = \mathbf{A}\mathbf{S}(t) \tag{2.12}$$

is solved by

$$\mathbf{S}(t) = e^{\mathbf{A}t}\mathbf{S}(0) \tag{2.13}$$

where we have used a matrix exponential. Once the constants defining $\mathbf{A}$ are known, a time-varying population vector $\mathbf{S}(t)$ is easily (if slowly) calculated. A continuously-varying discrete age distribution can be found by normalizing $\mathbf{S}(t)$,

$$g(i,t) = \frac{S_i(t)}{\sum_{i=0}^{N} S_i(t)}, \tag{2.14}$$

where the index i denotes the age bin in question. By hypothesis, the population is exponentially growing. If we treat the sum as a single variable σ, we can infer[1] from (2.11) that

$$\frac{d\sigma(t)}{dt} = K\sigma(t) \;\rightarrow\; \sigma(t) = P_0 e^{Kt} \tag{2.15}$$

Therefore we can replace the denominator of (2.14) with a single exponential, and choose P_0 to match the initial population size

$$P_0 = \sum_{i=0}^{N} S_i(0) \tag{2.16}$$

Choosing $P_0 = 1$ is a convenient choice, so that one begins with a state vector $\mathbf{S}(0)$ that is a proper probability distribution.

Calculation of matrix exponentials is very computationally expensive, which makes this type of precise modeling quite impractical for a large number of states. The model still has value in showing the form of age distributions within exponentially-growing populations. Nonetheless, due to the complexity of the matrix exponential model, we will need to employ some simplifying assumptions to move to continuous time, continuous age models.

2.1.3 Continuous time, continuous age models

An entirely different approach to the problem is to model the changes in a single observed object as a renewal process [Pap84]. In this section, we will assume individual lifetimes are independent and identically distributed continuous random variables, and that the lifetime distribution of a particular object does not change over time (the distribution is stationary). Not surprisingly, the lifetime probability density, $f(t)$, is closely related to the age probability density, $g(t)$. The act of observing "the age is t units" is the same as knowing "the lifetime is no smaller than t units."

[1] We note in passing that $\sigma(t)$ could be found as a special (scalar) case of (2.13) by using $\mathbf{A} = K$ and $\mathbf{S}(0) = P_0$.

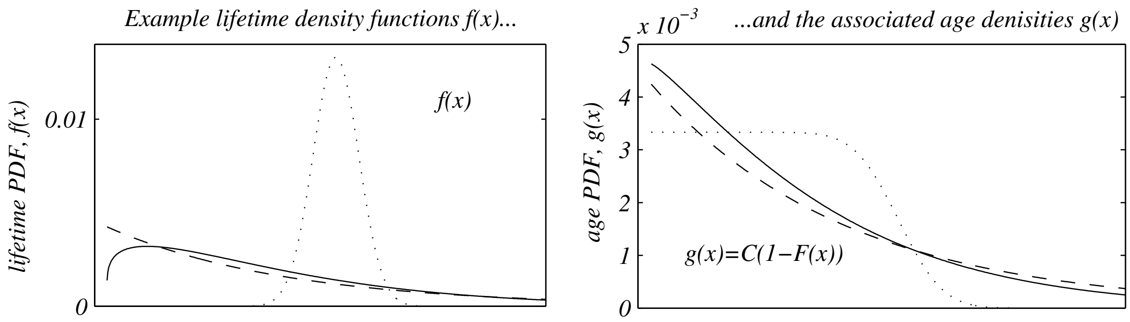


Figure 2.2: **Relationship between lifetime and age distributions.** On the left, we show three hypothetical lifetime distributions; a Gaussian (dotted), Weibull (solid), and an exponential (dashed). On the right, we show the corresponding age distributions. For the Gaussian, the age distribution is a renormalized and shifted complementary error function (erfc). For the exponential, the age and lifetime distributions are identical. The age distribution for the Weibull has a more general shape. Note that periodic lifetimes imply uniform age distributions.

Intuitively, this indicates that the PDF $g(t)$ should be proportional to the probability $1 - F(t)$ of a given lifetime exceeding t units, where $F(t)$ is the CDF corresponding to $f(t)$. To make $g(t)$ a proper probability distribution, the constant of proportionality is chosen so that $g(t)$ is normalized. This intuition proves correct and formal methods [Pap84] show that

$$g(t) = \frac{1 - F(t)}{\int_0^\infty [1 - F(t)]\, dt}. \tag{2.17}$$

This relation is analogous to the combination of (2.2) and (2.3), which describe the discrete lifetime and age distributions. Some examples of this relationship are shown in Figure 2.2.

Establishing the relationship of age to lifetime is useful, since it may be difficult to sample the distribution $f(t)$ directly. Rather, it may be easier to estimate change rates using samples from the age distribution $g(t)$ and then use (2.17) to estimate $F(t)$ and then $f(t)$. Aliasing of $f(t)$ may happen when a page change is observed, since an observer can only conclude that one *or more* changes have occurred since the previous observation. In observing ages, there is no such difficulty. Avoiding the aliasing problem is not magic; we are merely making proper use of the fact that another observer has sampled the modification time on our behalf. As mentioned earlier, observation of a web page age requires the availability of the `Last-Modified` information. If this information is not available then we have no choice but to deal with the aliasing problem by estimating $f(t)$ from observed lifetimes.

Much of what was presented in the earlier sections allowed for the probability of a change to vary with time. While these models are useful when computational resources are plentiful, observation of large collections will force some simplifying assumptions. The simplest possible lifetime model, and a good one to use to reduce the difficulty in modeling, is one in which objects change memorylessly. Intuitively, this means the probability of an object being altered in some short time interval is independent of how much time has elapsed since the last change occurred. This is a common model used in queuing systems and statistical reliability theory [Pap84]. For such objects, $f(t)$ is an exponential distribution with parameter λ. The exponential distribution has many useful qualities that permit some pleasant closed-form solutions. In Chapter 3, we will discuss why this distribution is a good choice for modeling many web pages.

Objects for which $f(t)$ is an exponential distribution also have exponentially distributed ages $g(t)$, since

$$\begin{aligned} 1 - F_c(t) &= 1 - \left(1 - e^{-\lambda t}\right) \\ &= e^{-\lambda t} \end{aligned}$$

implies

$$\begin{aligned} g(t) = \frac{1 - F_c(t)}{\int_0^\infty [1 - F_c(t)]\, dt} &= \frac{e^{-\lambda t}}{\int_0^\infty e^{-\lambda t}} \\ &= \lambda e^{-\lambda t}. \end{aligned} \tag{2.18}$$

This means we can estimate an object's lifetime PDF, assuming an exponential distribution, using only age observations. If these are not available, then we must accept that aliasing will occur in the direct observation of changes.

2.2 Measuring performance in monitoring systems

Using the different types of models developed in the first half of this chapter, we can devise different ways to measure the performance of a monitoring system. These ideas will be applied in Chapter 4 to develop optimization routines for scheduling observations.

2.2.1 Discrete time performance metrics

For the discrete-time model where a document moves between ages according to a Markov process, we can define an objective function that can be evaluated for an index. The objective function in a real system could be much more complex; we will not seek to incorporate all features into our initial work on the performance function. Here are two possible choices:

1. Count the expected number of objects that are incorrectly indexed. An object in the collection is indexed correctly if no changes have occurred for the object that have not been recorded in the index.

2. Add the total time out-of-date for the entire collection (as was developed in [CLW97]). Unobserved object changes add to the cost in proportion to how long the object is expected to have been out-of-date.

We restrict the problem using some simplifying assumptions. First, assume the collection of objects under observation contains d objects (we will treat d as a constant for convenience). Second, assume that objects can be retrieved at a rate of N_o objects per time interval. Let $\mathbf{M}_r$ be a matrix of transition probabilities for the r^{th} object, as was defined earlier in (2.1). The states of this system, corresponding to the rows of $\mathbf{M}_r$, are the possible ages of the r^{th} object in the collection. Any unit of time can be used, so long as it is consistent throughout the collection and the rate N_o is expressed in the same unit. For each object r, we know that it was last observed k_r time units ago to have age i_r. The probability that object r has changed during this k_r-unit interval is

$$\begin{aligned}\text{Prob}\left(change|\left\{i_r, \mathbf{M}_r, k_r\right\}\right) &= \sum_{j\in[0,i_r+k_r-1]} \left[\mathbf{M}_r^{k_r}\right]_{i_r j} \\ &= 1-\left[\mathbf{M}_r^{k_r}\right]_{i_r(i_r+k_r)}\end{aligned} \tag{2.19}$$

We note that this cost is between 0 and 1, corresponding to the cases in which we are absolutely certain that an object is not out-of-date (0) or is out-of-date (1) at the end of a time interval.

Using this result, we can formulate costs such as the two enumerated above. For example, at any given time, the expected total number of objects that are incorrectly indexed (meaning, the indexed version is out-of-date) is just the sum of the probabilities listed in (2.19) over the entire collection, so

$$C_{(1)} = \sum_{r=1}^{d} \left(1-\left[\mathbf{M}_r^{k_r}\right]_{i_r(i_r+k_r)}\right). \tag{2.20}$$

Since all terms are on [0,1], the cost $C_{(1)}$ is nonnegative and can be no greater than the total number of objects, d.

The complement of each term in the summation (2.20) can be used if it is preferable to express performance as the probability of an object being correctly indexed. In this case, we write the probability of a randomly selected object being up-to-date as

$$\alpha = \frac{1}{d}\sum_{r=1}^{d} \left[\mathbf{M}_r^{k_r}\right]_{i_r(i_r+k_r)}. \tag{2.21}$$

In the next section we return to this idea of calculating the probability α of a random object being correctly indexed.

The second metric listed, the expected total time out-of-date for all objects, is calculated using a sum in which the out-of-date probabilities weight the corresponding number of time units out of date. Thus we express the second cost function as

$$C_{(2)} = \sum_{r=1}^{d} Z\left(i_r, k_r\right) \tag{2.22}$$

where the expected number of time units out-of-date for object r is given by

$$Z\left(i_r, k_r\right) = \sum_{j=0}^{k_r-1} \left(k_r - j\right) P\left(j + i_r, i_r\right). \tag{2.23}$$

Here, $P(j, i_r)$ is the probability of an object changing (and thereby going out of date) at exactly age j, given that the page was last observed to have age i_r. This can be found from the conditional probabilities $p_{reset}(t)$. For example, $P(1,0)$ is, trivially, $p_{reset}(0)$. The probability of having gone out of date on the second time unit, $P(2,0)$, requires the object to have both aged past the first time unit, with probability $(1 - p_{reset}(0))$, and changed on the second time unit (conditioned upon not having changed on the first), with probability $p_{reset}(1)$. The probability of the combination is just the product of the individual probabilities, so that

$$P(2,0) = (1 - p_{reset}(0))p_{reset}(1). \tag{2.24}$$

This same logic describes $P(N, i_r)$, since the event of changing on the N^{th} time unit is conditioned upon the events of having reached each age in succession, one integer age at a time. So, in general,

$$P\left(j, i_r\right) = p_{reset}\left(j\right)\left[1 - p_{reset}\left(j-1\right)\right]\ldots\left[1 - p_{reset}\left(i_r\right)\right], \tag{2.25}$$

which can be written using a repeated product

$$P\left(j, i_r\right) = p_{reset}\left(j\right) \prod_{m=i_r}^{j-1} \left[1 - p_{reset}\left(m\right)\right]. \tag{2.26}$$

The terms in the product describe the probabilities of progressing through each of the ages from i_r to $j-1$, given that no change had occurred up to that point. The repeated product is then multiplied by the probability of a change at time interval j. Note that these probabilities can be calculated recursively, since

$$P\left(j+1, i_r\right) = P\left(j, i_r\right) \frac{\left[1 - p_{reset}\left(j\right)\right]}{p_{reset}\left(j\right)} p_{reset}\left(j+1\right). \tag{2.27}$$

Recursion can also be used to simplify the calculation of (2.23). As an example, consider the difference between finding $Z(i_r, 3)$ and $Z(i_r, 4)$, which demonstrates how to advance the cost contributed by a single object if a single time unit elapses:

$$\begin{aligned} Z(i_r,3) &= 3P(i_r,i_r)+2P(i_r+1,i_r)+P(i_r+2,i_r) \\ Z(i_r,4) &= 4P(i_r,i_r)+3P(i_r+1,i_r)+2P(i_r+2,i_r)+P(i_r+3,i_r) \end{aligned} \tag{2.28}$$

If the argument j is advanced by 1, we can simply add to our previous value a partial sum of the sequence $P(j,i_r)$. Using (2.28), the difference in consecutive values would be calculated as:

$$\begin{aligned} Z(i_r,4)-Z(i_r,3) &= P(i_r,i_r)+P(i_r+1,i_r)+P(i_r+2,i_r)+P(i_r+3,i_r) \\ &= S_3(i_r)+P(i_r+3,i_r) \end{aligned} \tag{2.29}$$

where we have defined

$$S_N(i_r)=\sum_{j=0}^{N-1} P(j+i_r,i_r) \tag{2.30}$$

to be the N^{th} partial sum of the sequence of probabilities $P(i_r+j,i_r)$. Obviously, this partial sum can be calculated recursively as well, since $S_{N+1}(i_r)=S_N(i_r)+P(i_r+N,i_r)$. Being a partial sum of mutually exclusive and exhaustive probabilities, $S_N(i_r)$ approaches 1 as $N\to\infty$. The probabilities are exclusive, since an object cannot change on more than a single time interval, and they are exhaustive in that the only possible times for the page to go out-of-date are $t\in[i_r,\infty]$.

Summarizing, we can recursively calculate Z as

$$Z(i_r,N+1)=Z(i_r,N)+S_{N+1}(i_r) \tag{2.31}$$

In the limit, each step simply adds one to the expected number of time units out-of-date, corresponding to the notion that an infinitely old object will have been modified at least once, and therefore grows one more time interval out-of-date for each time step that goes by without an observation.

2.2.2 Continuous-time performance: (α,β)-currency

From the discrete time measures used above, we now turn our attention to measures of performance in continuous time monitoring systems. Specifically, we introduce the intuitive concept of (α,β)-currency, which builds upon the idea of counting the correct objects in an indexing system. When we

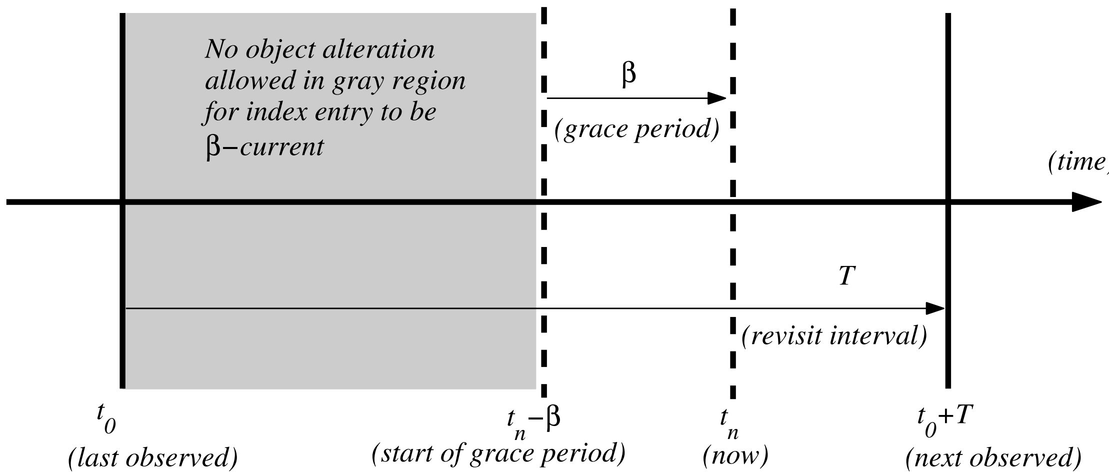


Figure 2.3: **Definition of "β-current".** This diagram shows what is necessary for an index entry to be current with respect to a grace period, β. In order to be β-current, no modification of an object can go unobserved up to β time units before the present.

presented (2.21), we mentioned how it is sometimes desirable to find the probability of a random item being correctly indexed. This section expands upon this idea. In combination with a distribution of change rates for a collection of objects, this new performance measure will allow us to estimate the speed of re-indexing required in order to maintain a given level of currency.

Recall from the introduction that an object's index entry is β-current if the object has not changed between the last time the object was re-indexed and β time before the present. In this sense, we are willing to forgive changes that have occurred within β time units of the present. The grace period, β, relaxes the temporal aspect of what it means to be current. The smaller β is, the more "current" our information about the page is. See Figure 2.3 for a graphical depiction of the concept.

To determine whether or not an index entry for an object is β-current, we need to know the most recent time t_Δ at which it changed. Assume that the object was last observed at time t_o. With this notation, the index entry corresponding to an object is β-current at time t_n if the page did not change between the last observation (at time t_o) and β units before the present, or time $t_n - \beta$

(assuming $t_o \le t_n - \beta$). For $t_o > t_n - \beta$, the entry is by definition β-current because the most recent unobserved change can occur either within the grace period or before we observed the object at t_o, but this includes all past time.

Combining these two cases, the probability that the index entry for an object is β-current at time $t_n > t_o + \beta$ is

$$\Pr(\text{ a fixed object is } \beta\text{-current} \mid t_o, t_n) = 1 - \Pr(t_o \le t_\Delta \le t_n - \beta) \tag{2.32}$$

where these probabilities are understood to be for a fixed, given object within the index. We now compute the probability α that the index entry for a randomly (according to some probability distribution) selected object is β-current. This probability distribution can be thought of as a demand imposed upon the index by its users.

Probability of β-currency for a collection of memoryless sources

The above expression (2.32) for a single object is stated in terms of a conditional probability. Given a prior distribution on the variables t_o and t_n, we can use Bayes' Theorem or the total probability theorem to eliminate them.

In our model, each object has a change rate λ and an associated distribution of re-indexing times T (a periodic re-indexing system will have a single constant T_0). These parameters determine density functions which, together with the grace period β, specify the probability α of being β-current. First, define the probability Pr(an object is β-current $\mid \lambda, T, \beta, t_n$) to be the probability of a single index entry being β-current given λ, T, β, and the time t_n at which the index is examined. Second, define the density $h(\lambda, T)$ to be the joint probability density for (λ, T). We assume that $h(\lambda, T)$ is independent of the time t_n, which is distributed according to a density $x(t_n)$. Using these densities and Bayes' Theorem, the probability α that the system is β-current is

$$\begin{aligned} \alpha &= \Pr(\text{The index is } \beta\text{-current}) \\ &= \iiint \left[\Pr(\text{a single object is } \beta\text{-current} \mid \lambda, T, t_n) x(t_n) dt_n \right] h(\lambda, T) d\lambda dT \end{aligned} \tag{2.33}$$

The integral is restricted to the first octant since no negative times or rates are allowed. In some settings, it is reasonable to assume a dependence between T and λ, since different re-visitation periods may be desirable for sources with different change rates. We investigate this in the chapter devoted to optimizing observation of changing sources. An explicit dependence between the two, where $T = w(\lambda)$, will change the outer integral over the (λ, T)-plane into a line integral. For example, setting a constant revisit time $T = T_0$ selects the line parallel to the λ-axis, and the differential element along this line remains dλ.

We will now evaluate (2.33) for a single, memorylessly-changing object. As before, this object has a change rate λ, and is observed periodically (every T time units). The probability that the next change occurs in the time interval $[t_1, t_2]$, where the last observation or change (whichever occurred most recently) was at time $t_o \leq t_1 < t_2$, is

$$\int_{t_1 - t_o}^{t_2 - t_o} \lambda e^{-\lambda t} dt = e^{-\lambda(t_1 - t_o)} - e^{-\lambda(t_2 - t_o)}. \tag{2.34}$$

When $t_1 = t_o$, this reduces to

$$1 - e^{\lambda(t_2 - t_o)} \tag{2.35}$$

so that the probability that an object change *did not occur* in the interval $[t_o, t_2]$ is the complement,

$$1 - \left(1 - e^{\lambda(t_2 - t_o)}\right) = e^{\lambda(t_2 - t_o)}. \tag{2.36}$$

To evaluate (2.33) we need to specify the function $h(\lambda, T)$ as well as the distribution of times $x(t_n)$ over which we average the β-currency of the index. First, we consider the limits on the inner integral over t_n. Assuming as we have that all the objects change memorylessly, it is sufficient to evaluate the inner integral in (2.33) over a single observation period T, since adding additional periods would only replicate the integral over one period.

For convenience, we choose an interval starting at $t_o = 0$, at which time an observation was last made, and extends until the time T at which the next observation occurs. Using this interval, the probability that the object does not change between $t_o = 0$ and $t = t_n - \beta$, and is therefore β-current, is

$$\Pr(\beta\text{-current } | \lambda, T, t_n) = e^{-\lambda(t_n - \beta)} \text{ for } \beta < t_n < T \tag{2.37}$$

by the above discussion. Further, note that the object is β-current with probability one in the interval $[t_n - \beta, t_n]$. Specifically,

$$\Pr(\beta\text{-current } |\lambda, T, t_n) = 1 \text{ for } 0 < t_n < \beta \tag{2.38}$$

Combining these, the expected probability of a single object being β-current over all values of the observation time t_n, using a uniform density $x(t_n) = 1/T$, is just an average value of the piecewise-defined $\Pr(\beta\text{-current } |\lambda, T, t_n)$ on the interval $t_n \in [0, T]$. This gives

$$\begin{aligned} \Pr(\beta\text{-current } |\lambda, T, \beta) &= \int_0^\beta \frac{dt_n}{T} + \int_\beta^T \frac{1}{T} e^{-\lambda(t_n - \beta)} dt_n \quad (2.39) \\ &= \frac{\beta}{T} + \frac{1 - e^{-\lambda(T-\beta)}}{\lambda T}. \quad (2.40) \end{aligned}$$

In the first integral of (2.39), the probability of being β-current is one when $t_n \in [0, \beta]$, since this would force any change to be within β units of the present. We can clean up (2.40) by expressing β as a fraction ν of T (that is, $\beta = \nu T$) and setting $z = \lambda T$. With these changes, (2.40) becomes a function of the expected number of changes per re-indexing period, z, and the ratio of the grace period to the observation period, ν. When $z > 1$, an object is expected to change once or more prior to T, whereas $z < 1$ suggests fewer than one change expected before T. What fraction of these changes fall within the grace period β is loosely described by the parameter ν; some curves are shown for different choices of ν in Figure 2.4.

We note in passing some properties of the curves in Figure 2.4 that verify our intuition. First, note that the probability of being β-current goes to ν as the expected number of changes per observation period λT approaches infinity. A large number of expected changes per observation period implies an object which is observed much too slowly; the object changes many times between observations. As such, in the high rate limit, ν simply represents the percentage of these changes that occur during the grace period. For the case of a small number of expected changes per observation period, where objects are sampled much faster than they change, the probability of a page being β-current approaches one, regardless of the grace period fraction ν.

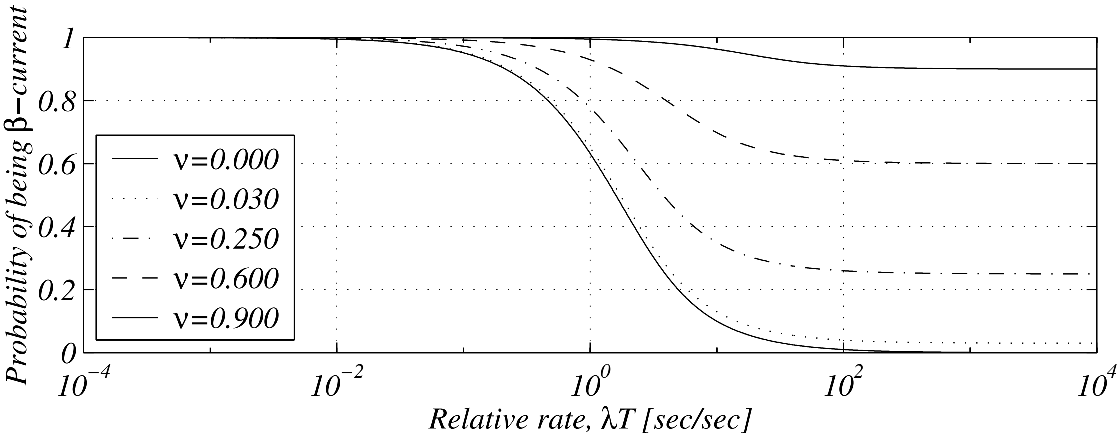


Figure 2.4: **Probability of β-currency vs. expected number of changes per observation interval.** Expected value of $\Pr(\beta\text{-current} \,|(\lambda, T, \nu))$ as a function of expected number of changes per observation period $z = \lambda T$ and grace period percentage $\nu = \beta/T$

2.3 Summary

This chapter has presented some selected theoretical aspects of monitoring changing information sources, to be applied in the context of the web in Chapter 3 and in optimization routines presented in Chapter 4. We began by considering discrete-time, Markov chain models in which the age, or time since the last change, was used as the state of an object. This led to a hybrid model that allowed each state's occupancy to vary continuously rather than discretely. This second model was a stepping stone towards the final model, which treated both time and age as continuously-varying. For both the discrete and continuous time cases, we constructed performance metrics that allow us to gauge how up-to-date an index of changing sources is. This progressed from the expected number of objects that are out of date (in the discrete case), to the concept of (α, β)-currency (in the continuous case). The latter metric was then applied to find the probability of being β-current for objects that change in a memoryless fashion. We concluded with a calculation of the expected probability α for a collection of such objects, according to an imposed demand distribution. In Chapter 3, we will present data obtained by monitoring HTML documents on the World Wide Web, an excellent example of just a collection of monitored objects. Chapter 4 will synthesize the theory from this Chapter with the data to be presented in Chapter 3.

Chapter 3

A case study: Observing the WWW

A good example of a large observation system, and one with wide general exposure, is the WWW search engine. These systems have undertaken the enormous task of attempting to observe, index, and catalog the material on the web for the benefit of their users, done in such a way as to satisfy user queries with accurate and up-to-date references.

The accessibility and scale of the web makes it an excellent testbed for a study of monitoring systems in general. Search engines provide baseline guides for index performance, and web documents are easy to download and inspect for changes. These factors provide an environment that is readily testable and convenient to analyze. The lessons learned during this analysis apply to any large observation system. In this chapter we investigate the ways in which web pages change, as well as how often these changes take place. Using these estimates, it is possible to use the theory from the previous chapter to gauge how current search engine indices can be kept when using simple re-indexing strategies.

3.1 Web page data collection and analysis

We have collected a large sample of web page data to study web documents' inherent rate of change. This is a crucial factor in the search engine's re-indexing problem—observations must be scheduled often enough to keep track of changes. For our data collection, we use observations made by a

	Field	Description
1.	url	Complete web page URL
2.	title	First 80 characters of page title
3.	last_modified	Last-modified date in seconds since 1/1/1970 00:00:00 GMT
4.	last_observed	Observation time in seconds since 1/1/1970 00:00:00 GMT
5.	content_length	Content length (in bytes) mod 2^{32}
6.	num_tokens	Number of structural "tokens" mod 2^{32}
7.	num_images	Number of <IMG> tags mod 2^8
8.	num_links	Number of <A ...> tags mod 2^{16}
9.	num_tables	Number of <TABLE> tags mod 2^8
10.	num_forms	Number of <FORM> tags mod 2^8
11.	num_ads	Number of 468×60 images (common advertisement size) mod 2^8
12.	num_lists	Number of <OL> and <UL> tags mod 2^8
13.	text_hash	the time16-bit hash of combined plain text strings
14.	image_hash	16-bit hash of image SRC references
15.	link_hash	16-bit hash of hyperlink HREF references

Table 3.1: **Web page summary data retained from each observation** This table gives the categories of data collected on each observation, for all web page observations made between March 1999 and March 2000. The table is organized by database field along with a short description of each field's contents. Short hash functions were used for efficiency of computation and storage.

web clipping service called "The Informant" (mentioned in the Introduction) that downloads and processes on the order of 100,000 HTML web pages daily. It performs two types of tasks: it monitors specific URLs on its users' behalf, and it runs standing user queries against one of four search engines[1] at specified intervals. Any of three events trigger a notification of a user by email: (1) a monitored URL changes, (2) new results appear in the top results returned by a search engine in response to a standing query, or (3) any of the current top search results shows a change.

In March 1999, we began archiving HTML page summary information for all page downloads. As of this writing, we have downloaded and processed over 350 gigabytes of HTML data. The categories of summary data retained from the year's observations (from the Informant's data between late February 1999 and March 2000) are listed in Table 3.1. The archived information includes the last-modified time stamp (if given), the time of observation (using the remote server's time stamp if possible), and stylistic information (number of images, tables, links and similar data). In addition,

[1] AltaVista, Excite, Infoseek, and Lycos

beginning in February 2000, we began keeping a term vector[2] representation of about 1/8 of our observations, along with the links observed in each document and the text enclosed within those links. For the observations for which only the summary information was recorded, the database could cover a long time period without occupying too much space. The year's worth of raw numerical data for the 150,000-page sample used to generate our plots occupies about 130 megabytes. Additionally, we have kept the URL's and titles (the first 80 characters) for all pages observed. The indices into this data occupy about as much space as the data itself, so the total size is around 10 gigabytes, which is an acceptable size considering the time span and the number of pages tracked (around 3,000,000). We use an Informix Universal Server (v9.1) database on a Sun Ultra 30 and have around 28 million distinct observations of roughly three million unique URLs (not including search engine results pages).

3.1.1 Sampling issues

We should note that the Informant selects and monitors web pages in a very specific way, so conclusions from the data must be interpreted only after knowing our sampling methods. Since the Informant makes repeated observations primarily of those pages ranked high by search engines, there is a bias against those pages that are not relevant to our users' standing queries. Our sample is also biased towards the URLs that users have deemed worth monitoring. While neither of these is crippling, they do slant our results towards those pages that our users wish to monitor. We do not claim that this bias is a popularity bias, since our users' queries are not necessarily the same as those which are of general interest.

Another important consideration is the sample rate. Standing queries are run at most once every three days for any single user, and some users' queries are run once every seven days or more. Therefore, the only way a page is observed more than once every three days is if it is needed by a different user on each of those days. Many monitored sites exhibit a partial overlap between users, resulting in observations made at irregular intervals. A number of popular sites (news sites,

[2] A "term vector" is a vector in which components correspond to words, and the value of each component is the number of times the corresponding word appeared in the document.

shareware distributors, proficient "keyword spammers"[3]) fall into this category. Moreover, to keep our service from annoying providers of popular content, we cache pages (and delete the cache prior to gathering each day's results), so that no more than one observation is made of a single page per day. In addition, since we run our queries periodically and only at night, sample times for any given page are correlated. For an illustration of this, see Figure 3.2, which shows a scatter plot of observation time and modification times.

Aliasing is also a concern. For extremely fast-changing pages, it is quite possible that many changes will occur between observations, making direct observation of all such changes impossible. When `Last-Modified` information (that is, a formal filesystem timestamp in the HTTP response header) is given, we can work around this by estimating change rates from observed ages rather than lifetimes. This will be discussed in greater detail in later sections.

While `Last-Modified` information is available for around 65% of our observations, the absence of such information does seem to indicate a more volatile resource. Specifically, for our data, not having this timestamp makes an observation of any given resource about twice as likely to show a modification. This makes some intuitive sense, since the purpose of providing a timestamp is to help web caches–timestamps are generally not provided for pages that are either dynamically generated or uncacheable for some other reason. Therefore, estimates of change rates based solely on pages that provide a timestamp are lower bounds (slowest estimate). Timestamps also show, indirectly, that most webpages are modified during the span of US working hours (between approximately 8 AM and 8 PM, Eastern time). This is shown in Figure 3.1. This is where any assumption of stationarity in change probability will break down; modifications are less likely during the low times on this plot.

Another look at how the `Last-Modified` times are distributed is provided in Figure 3.2. This is a scatter plot of a sample of Informant downloads, showing one year's worth of observation timestamp samples. The most obvious features in the plot are the broad vertical white areas, which correspond to the times when data was not being gathered, due to server downtime or other difficulties. In the

[3]This is the practice of filling documents with popular search terms, enclosed in comments or some other unviewable format, so that search engines will return the page as a result even when its viewable content is not relevant.

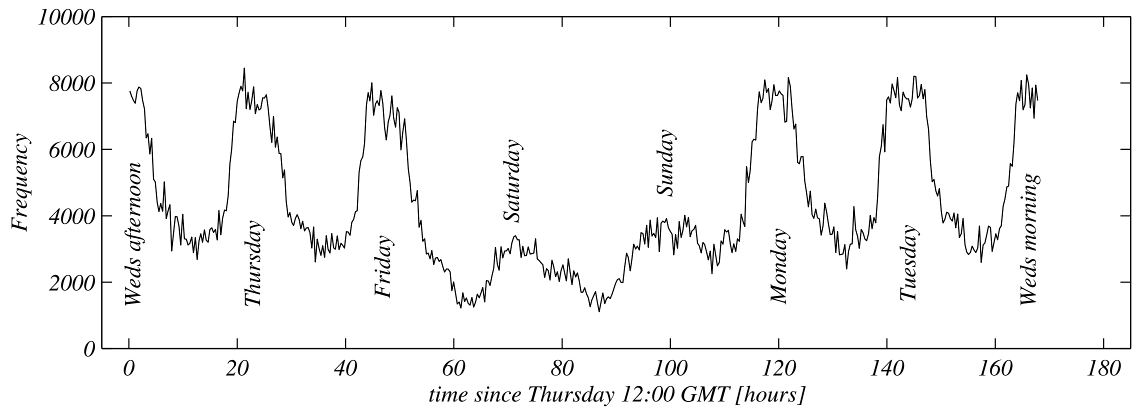


Figure 3.1: **Histogram: Last-modified times (GMT), mod 24×7 hours.** Peaks in modification frequency are clearly visible during US working hours, and diminish on weekends. Assumptions of stationarity in page alteration probability will break down at this scale. The day labels correspond to the workday in the United States; hence "Weds afternoon" is aligned with midnight Thursday GMT.

darker regions, a close look reveals that they are actually composed of dark bands separated by thin white bands, since our observations were all made at night, between about 9:00PM Eastern time and sometime the next morning (generally between 4 and 7AM). Third, notice the thinning that occurs within a horizontal stripe around late December every year, when even web administrators go on vacation and do not alter documents as frequently.

A quick assessment of the data also suggests that the age distribution is getting wider; it seems that probability mass is accumulating in the tail. This is shown in Figure 3.3, as a dependence of the age distribution on time. In the figure, light color represents higher probability density, and dark is lower probability. The lighter area towards the top right corner of the plot corresponds to a slight increase in the probability mass in this region. The changes to the age distribution over the year's worth of observations are fairly small; we will consider it to be constant in later sections and average the distribution over the entire year.

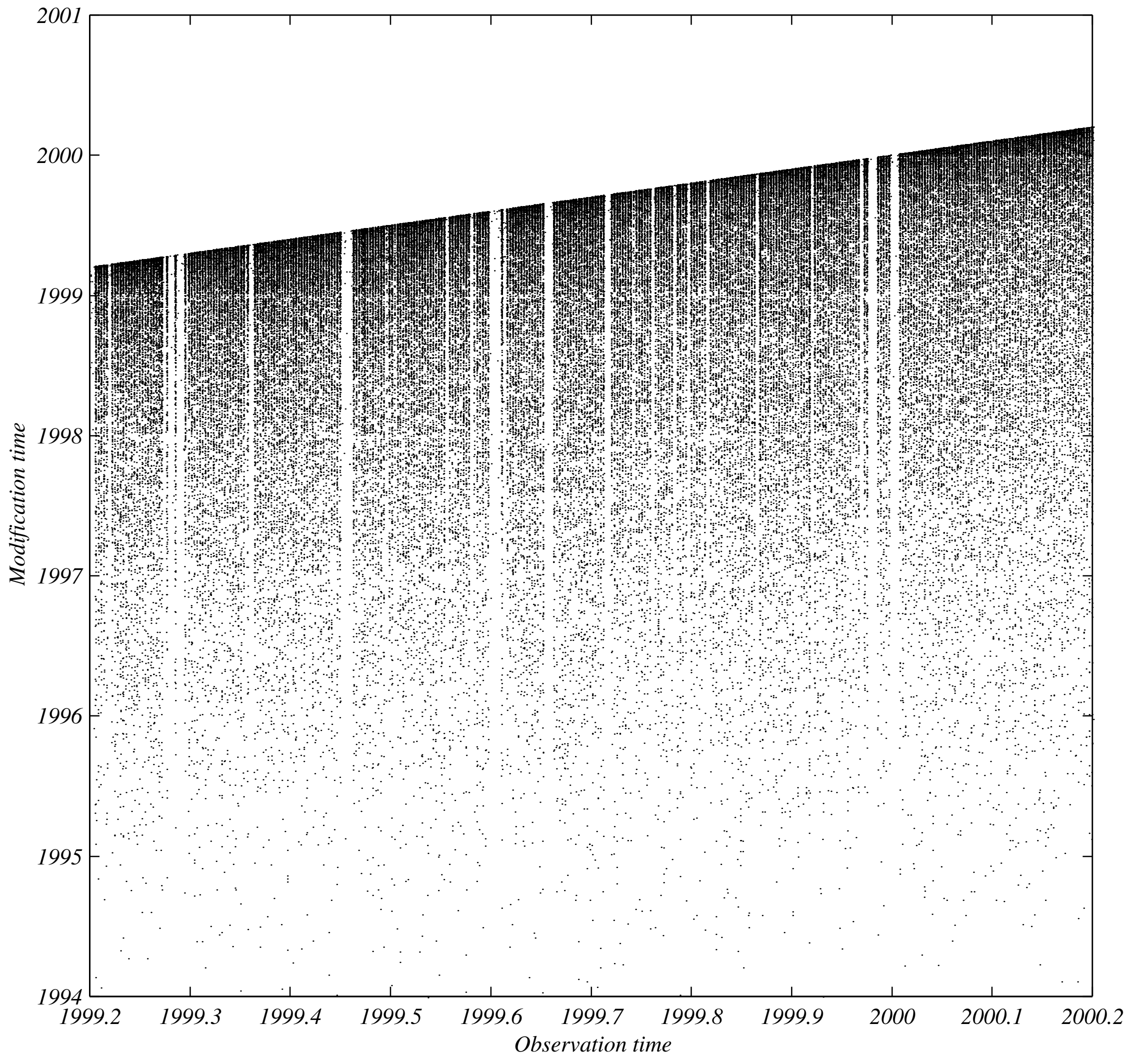


Figure 3.2: **Last-modified time vs observation time** This scatter plot shows what time periods are covered in the Informant dataset. Each dot is an observation; this is just a small fraction of all of our data. The horizontal axis shows the observation time, and the vertical axis shows `Last-Modified` times observed. Broad, vertical white stripes are times during which the Informant was not gathering this data at all, due to server downtime or other glitches. The many narrow white stripes separating the observation times correspond to the daytime; downloads were only run at night. This plot seems to show that the distribution of `Last-Modified` times is getting wider as time progresses; new times are appearing at the front of the distribution faster than the tail is losing density (see Figure 3.3 for more). Also, as a point of trivia, notice the horizontal lower density bands near each year boundary, showing that there are fewer modifications to web pages in the vicinity of the Christmas and New Year's holidays. The units on both axes are decimal years.

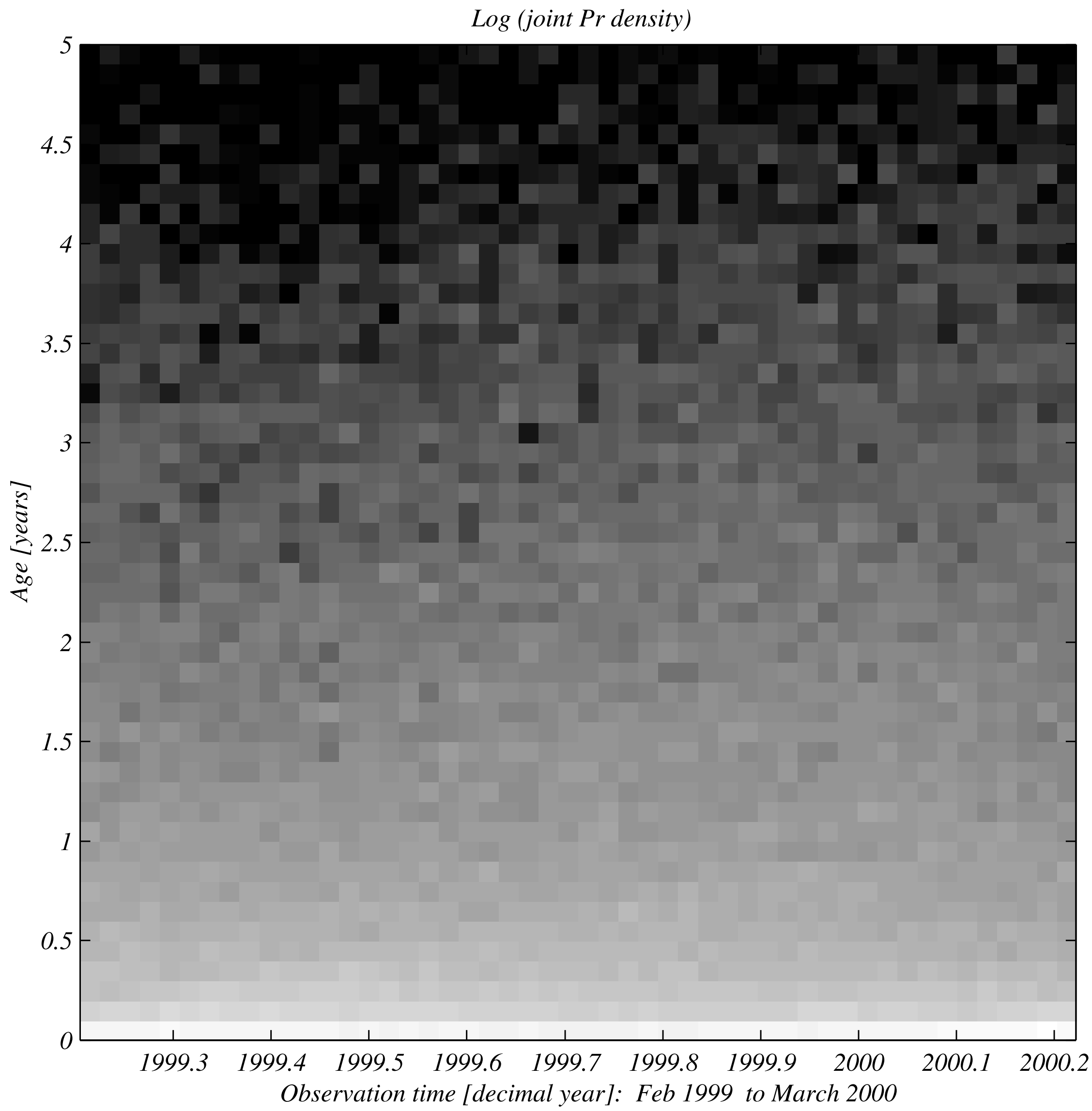


Figure 3.3: **Log-probability age density vs. observation time** This plot shows a time-dependent age distribution that confirms that the tail of the web's age distribution is lengthening. The vertical axis shows document age (in years) and the horizontal shows observation time (as a decimal fraction of the year). The image is colored according to log-probability density; light color is higher probability density and dark is smaller. The upward trend of probability mass in the tail is fairly small in terms of the total portion of the population represented. On a linear color scale this trend would be almost imperceptible.

3.1.2 Characterizing web documents and changes

With our web page data, we can see what kinds of things characterize web documents as well as document changes. For example, a very broad definition of change is the one in which any of the items listed in Table 3.1 (except URL or observation time) is altered for a document from one observation to the next. Given a randomly selected page change (not the same as a random change for a random page), what percentage will include an alteration to each of the above attributes? This is important when one is considering how a restricted definition of change will alter the number of changes observed and thus affect the measurement of the change rate. We show a chart depicting this comparison in Figure 3.4.

The single best indicator of change in our data kept for each observation is the content length, closely followed by the hash functions of link and image references, and the hash of the plain text. The plain text is the user-viewable text. Obviously, this is not the best way to catch all changes; a hash of the entire document or a byte-by-byte `diff`[4] would catch all changes. Calculating the `diff` would require storing the previous instance of the page for comparison. The plain text hash, by contrast, is not altered by extra whitespace or changed markup. A good indicator of general style change is a count of the number of "tokens" or stylistic elements, which changed in 56% of all observed changes. When the number of tokens does not change, it is likely that the page will look very similar to the previous instance. Also note that about 39% of all changes included a different number of links, as compared with a change to the link hash function in about 72% of changes.

As an example of how one might use this data, consider the example of a "prospector" robot that goes about looking for links to new pages so as to expand a search engine's coverage. Such a robot would be well served to revisit pages that frequently added new links, regardless of how else they did or did not change. By defining a change in a page to be a change to either the number of links, or the text contained in the `HREF` fields on that page, one could isolate pages that tend to provide new links. Moreover, the dynamics for link addition might be quite different than those for other types of change on a web page. Going one step further, the utility of these links could be measured in terms of their popularity once included in the index. In this way the space of documents could be divided

[4]This is a UNIX command for performing such comparisons.

according to how often pages provided new links, and how valuable those links eventually prove themselves to be, using a dynamic programming-type approach to page retrieval [Ber87]. Though we will not attempt to do so in this thesis, this idea could be used as a formal basis for heuristic methods for crawling, such as those outlined in [JCP98] and [CvdBD99].

3.1.3 Web page style and age

Not surprisingly, there is a correlation between the style of a webpage and its age. For example, in Figure 3.5, we show how the distribution of content-lengths and number of images depends upon age. Each plot shows two distributions, one using data from pages last modified between 6/94 and 6/95, the other using pages between 6/98 and 6/99, to show how more recently modified pages are frequently longer and have more images. Both distributions in the figure argue for the importance of space-saving technology (such as compression techniques in the HTTP-1.1 standard, cascading style sheets (CSS), and use of Extensible Markup Language (XML) where appropriate). Similar trends, sometimes much more pronounced, are seen in the usage of second-generation tags, such as the `<TABLE>` and `<FORM>` tags. While it might be feasible to use stylistic cues to estimate ages for pages that do not provide a timestamp, a far better solution is for content providers to include one along with an estimated expiration time. The HTTP standard has provisions for providers to include such information in the header as part of `Cache-control` fields. This potentially has many benefits, including better cache performance [Not99] and fewer wasted observations by search engines (if honesty in expiration estimation is enforced).

A popular question regarding our data is, "What about dynamically-generated pages?" We can determine an upper bound on what percentage of pages are dynamic by looking at how many pages change on every repeat observation. Following [DFKM97], we have plotted a cumulative distribution function of "change ratios" in Figure 3.6. A change ratio is defined by the number of changes observed, divided by the number of repeat accesses made. Obviously, this statistic depends heavily upon the sample rate, but it does give a feeling for the distribution of change rates. We have plotted change ratios corresponding to pages that were observed six times or more. A unit ratio indicates a resource that always changes faster than the sample rate, meaning it may be totally

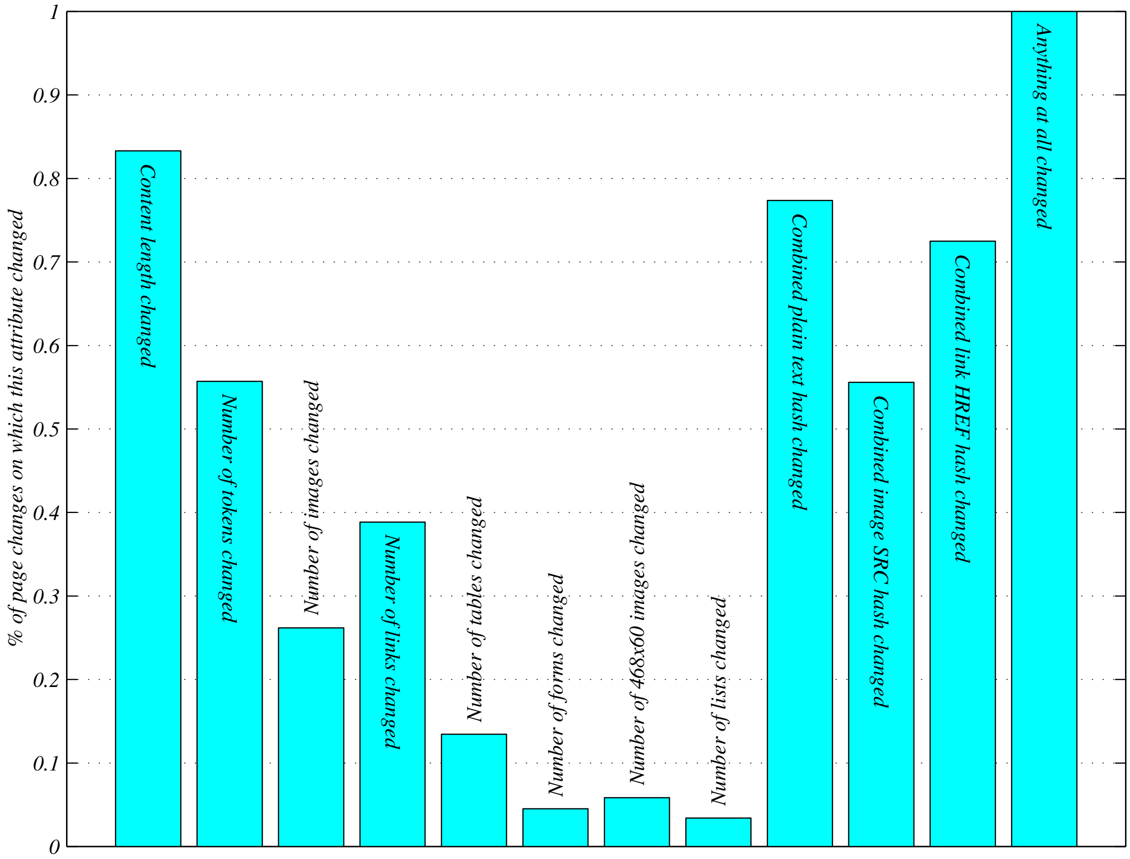


Figure 3.4: **Percentage of all page changes on which given attributes changed** For web page changes in a list of observations of 150,000 web pages, this figure shows what percentage of web page changes included a change in each attribute measured. For example, on the far left, we show that the content length changed for about 83% of all changes observed. Note also that "plain text" is just the user-viewable text (i.e., markup and whitespace are ignored).

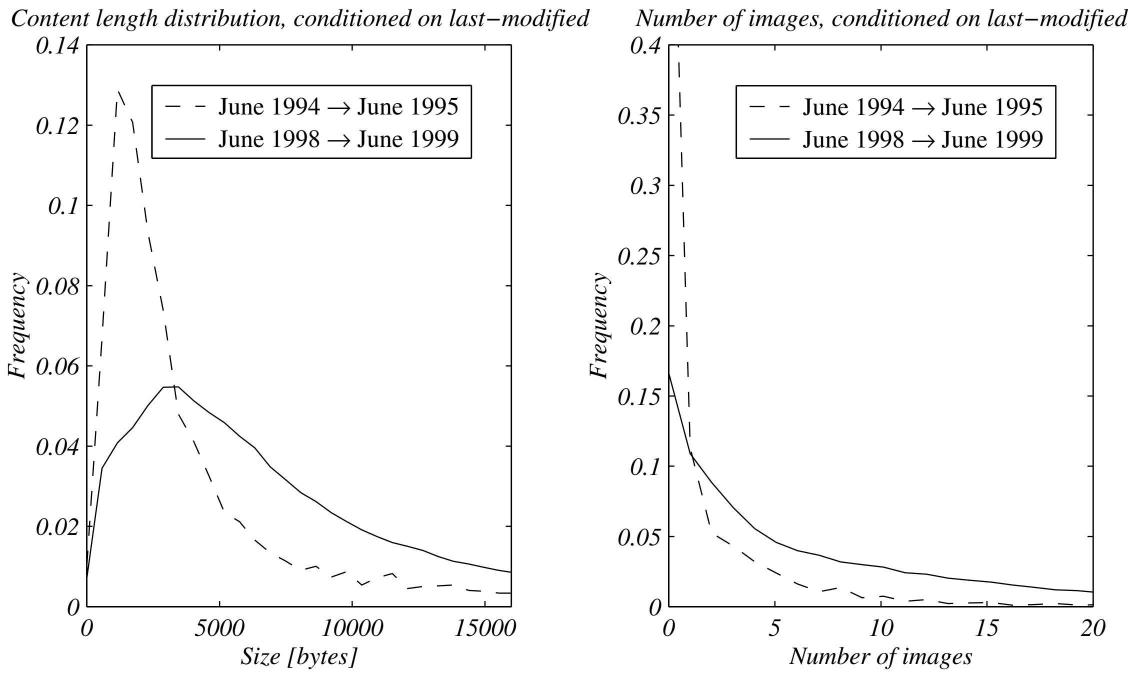


Figure 3.5: **Stylistic clues to webpage age.** On the left, we show two distributions of content-length, or the number of bytes in a webpage. One is for pages dated between 6/94 and 6/95, the other is for pages last modified between 6/98 and 6/99. Widespread use of space-intensive scripting languages and stylistic elements (`<FONT>` tags, precise table and image sizing, and so forth) has driven the content length upwards. It will be interesting to see if CSS (Cascading Style Sheets) reduces this. On the right, a similar trend is seen in the number of images, often used in more recently modified pages to make a more visually appealing presentation. Much of this reflects the shift from an academic-centric web to a commercial-centric one.

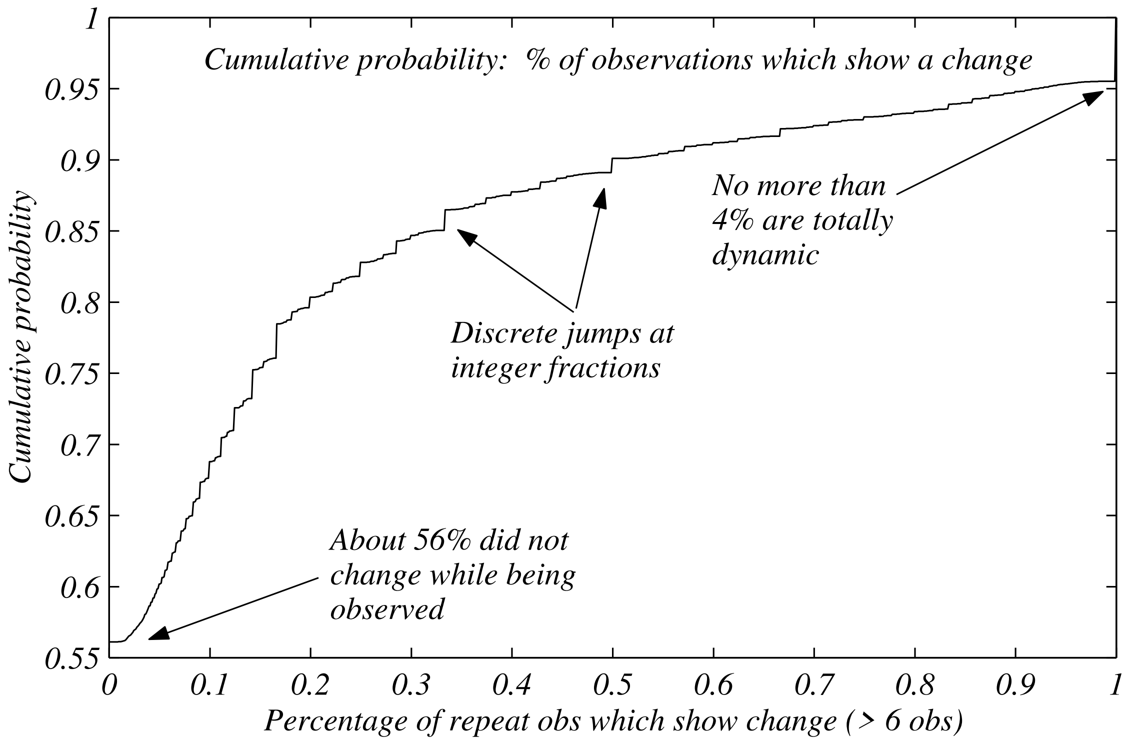


Figure 3.6: **Cumulative distribution of change ratios.** The "change ratio" for a page is defined as the number of changes observed divided by the number of repeat accesses. We have plotted the cumulative distribution of this statistic for pages which have been observed six times or more. This shows that no more than 4% of these pages are totally dynamic, while we have never observed any sort of change for 56% of pages. These values are very dependent upon the sampling scheme and are therefore not directly comparable to numbers taken from web caching studies.

dynamic, although it may just change very quickly. The plot shows that 4% of pages changed on every repeat observation (70% of these pages did not give a timestamp), while no change was observed for 56% of pages. The average page is observed 12 times over an average of 37 days, so the percentage of pages that did not change would be much smaller if the monitoring were over a longer timespan.

3.1.4 Web page age statistics

The difference between a downloaded page's last-modified timestamp and the time of observation is defined as the page's *age*, just as it was used in the previous chapter. Recording the ages of the pages in the Informant database allows us to make several inferences about how those ages are distributed.

Estimates of the probability density function (PDF) and the cumulative distribution function (CDF) of web page age are shown in Figures 3.7 and 3.8. A few observations about these plots give insight into the distribution of document ages. From the CDF, we see that about one page in five is younger than eleven days. The median age is around 100 days, so about half of the web's content is younger than three months. The older half has a very long tail: about one page in four is older than one year and sometimes much older than that. In a few rare cases, server clocks are set incorrectly, making the timestamp inaccurate. The oldest pages that appear to have correct timestamps are from around 1992, some of which are "archaeologically" interesting[5]. Our data on page age is similar to that found in an earlier study [DFKM97]; when the histograms in Figure 3.7 are altered so that the bins have the same size as in [DFKM97], our distribution matches their data for "infrequently-accessed" HTML pages.

Typical age observations are shown in Figures 3.9 and 3.10. Since pages are only observed for as long as they remain in any user's search results, the majority of pages in our collection are only monitored for a timespan less than the total length of the study. As such, no alterations are ever observed on about 56% of the pages we have monitored[6]. This type of behavior often appears like

[5]These may not be around for long; before they disappear, see http://www.w3.org/Out-Of-Date/ hypertext/DataSources/WWW/Servers.html (a listing of web servers from 1992) or http://www.hcc.hawaii.edu/guide/ www.guide.html (a web guide from 1993)

[6]This statistic obviously depends upon the length of time we monitor any given web page

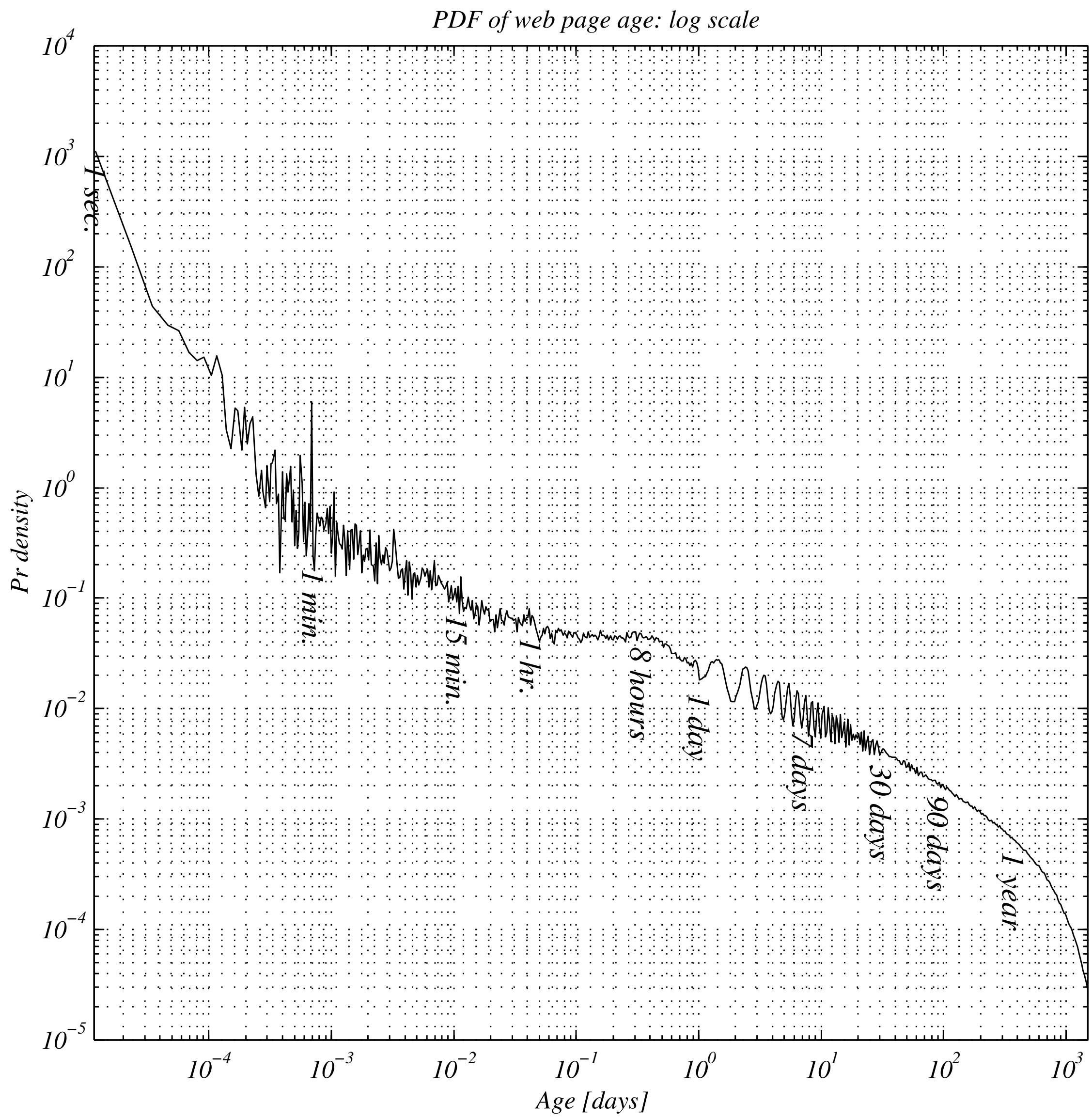


Figure 3.7: **PDF of web page ages.** Here we show an estimate of the probability density function (PDF) of web page age, generated using a histogram having logarithmically sized bins. Obviously this only represents pages for which `Last-Modified` dates were given. Several times are labeled; we infer from the peak at 1 minute that many servers round both modification and request timestamps to the nearest minute. Notice that ages between about 2 and 10 hours are almost uniformly distributed; this may be a consequence of running our downloads at night when pages are much less likely to be modified. Last, note the periodicity in observed ages past 1 day, corresponding to the period shown in Figure 3.1.

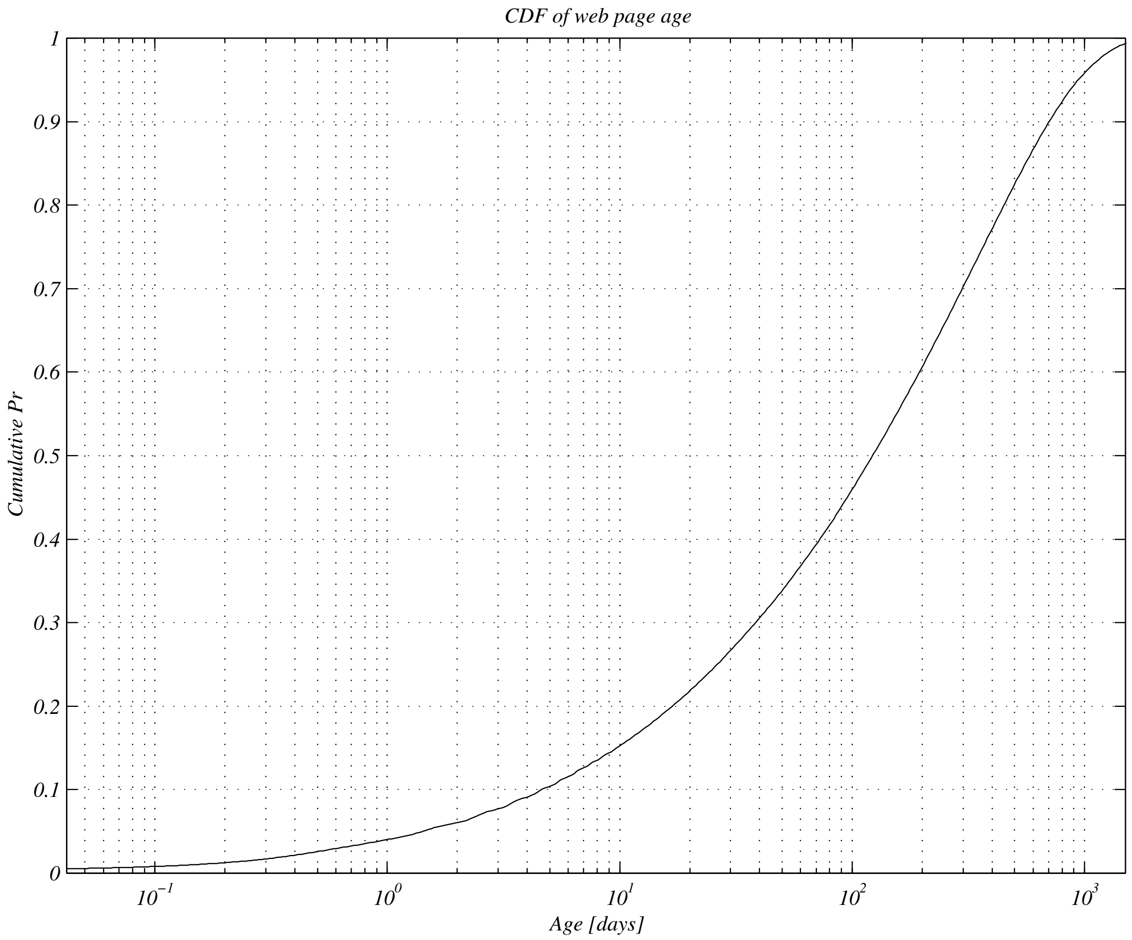


Figure 3.8: **CDF of web page ages.** The cumulative probability density function (CDF) of web page age was formed by estimating the integral of the function shown in Figure 3.7.

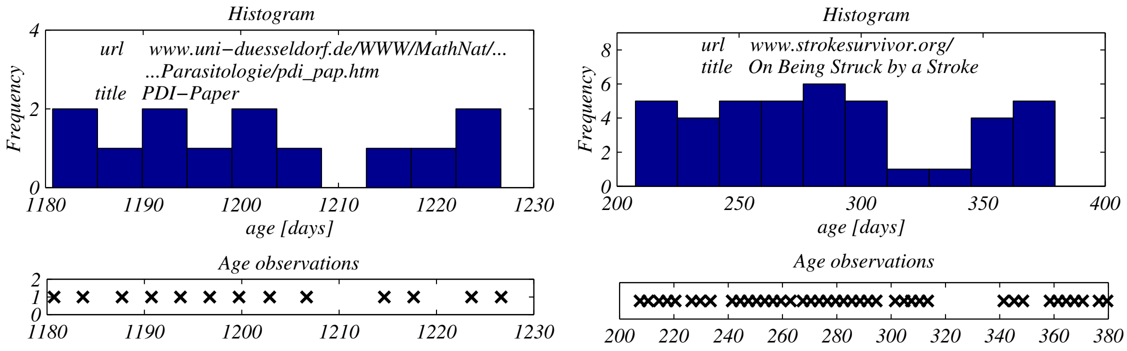


Figure 3.9: **Example age observations for relatively static pages.** Many of the pages we monitor do not change during the time they are observed, like the examples shown here. The upper plots are histograms, and the lower plots show the raw data. These examples show that many of the pages are quite old, and for some of them, the only change they will ever experience is their eventual disappearance (as when a server no longer exists, the page is moved by its owner, and so forth).

the examples shown in Figure 3.9. From these observations it cannot be determined whether these pages are totally static or that they may just change at a very low rate. We proceed with the assumption that all pages are dynamic, even if the only change they will ever experience is their disappearance. This assumption is necessary to avoid having to guess what fraction of pages will never change; such a figure would be necessarily be pure speculation. For pages that do have these longer lifetimes, the best we can do with our data is to obtain several (dependent) samples of the age distribution.

When web pages are more dynamic, their age samples look more like the examples in Figure 3.10, where the pages progressed through many changes while we have observed their ages. This usually produces distributions close to an exponential PDF. Some rapidly changing pages appear to be periodic, though the period is rarely larger than one day. Periodicity can be inferred from age distributions that appear to be approximately uniform. Still other pages are entirely dynamic, generated anew with each access, but these are not more than 4% of our collection, as could be seen from Figure 3.6.

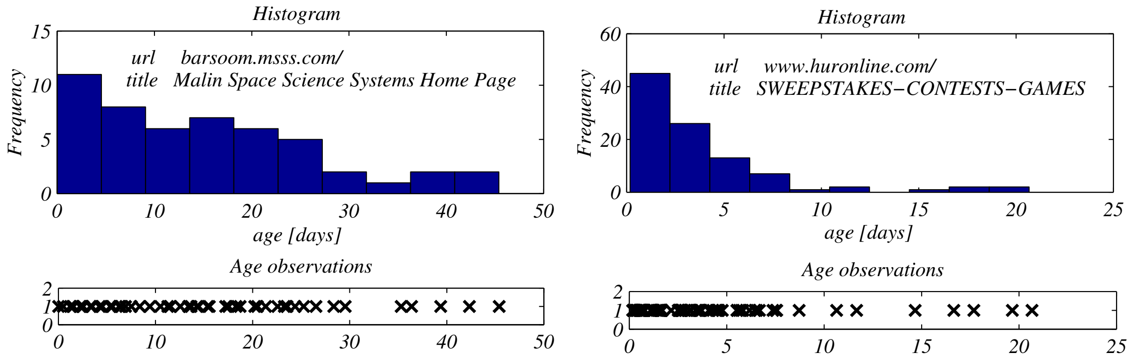


Figure 3.10: **Example age observations for changing web pages.** For some of our pages, we have observed a number of changes over a long timespan. The distribution of ages over this time is often approximately exponential, as can be seen in the histograms. The raw data is shown in the lower plots.

3.2 Term vector-based change analysis

At some point in the study of changing documents on the web, one must ask which changes are important. Towards answering this question, part of our work has focused on examining exactly what is altered within the textual content of a page. Using a term-vector approach, we have tracked the evolution of thousands of web pages in this space and present the results of that study in this section.

To this point, we have discussed statistics on rate of change with no regard for the size of the changes in question. Here are two possibilities between which we would like to distinguish: one web page might experience many small changes that do not significantly change its content, while another web page might change so radically that successive instances have almost no similarity to one another. A discussion of change is much more meaningful when the definition of change is normalized in some way.

The question of a change's importance is a difficult one. Two types of error can occur. After a change to the page, it may no longer be among the best responses to a user's query, and the changed page may now be a good response for another query where it had not been listed before. Restated, does the page still belong among the same query results after the change has occurred?

Alternatively, if the change were observed, would this cause the page to be listed as a result for another query? Towards answering these questions, we consider how a simple search engine's query response accuracy might be affected by changing pages.

3.2.1 Term vector evolution

One of the oldest models used in information retrieval is the term vector model ([WMB94] has a good discussion), in which a document is modeled as a collection of words and their corresponding frequency within that document. The frequency measure can be absolute (number of occurrences of each word) or relative (as a fraction of all possible words). For example, if there were a hypothetical language containing only two words, "`a`" and "`b`", the string "`ababbbbb`" could have the term vector representation $< 0.25, 0.75 >$. In some cases it is more convenient not to normalize the term vector, so that the magnitude of the term vector gives an idea of the length of the document. This would make the vector above $< 2, 6 >$ instead. Using binary term vectors (presence or absence of a term), Boolean queries can easily be expressed as dot products of bit vectors. Though simple search engines can be built using these representations, most commercial engines go well beyond, even so far as to store entire documents. Nonetheless, term vectors do provide a convenient mathematical means to represent a document's content. The "terms" in question need not be single words per se, as stemmed words, phrases, or projections of the vector onto subspaces of the term vector space could be used as components instead.

However a term vector is constructed, the rate of change of documents on the web makes it reasonable to consider the *dynamics* that describe changes from one vector in this sequence to the next. Web pages can then be visualized as in Figure 3.11 as vectors that drift in term space as their content is altered. Some pages will simply experience small, random motions in the vicinity of some mean value, while others will appear to take more of a "random walk" from their initial positions. In this section, we present some statistics on how a large sample of term vectors has evolved over a 45-day period. These data were collected from the Informant's web cache between February 20, 2000 and April 4, 2000, and the same biases that were explained earlier apply here as well.

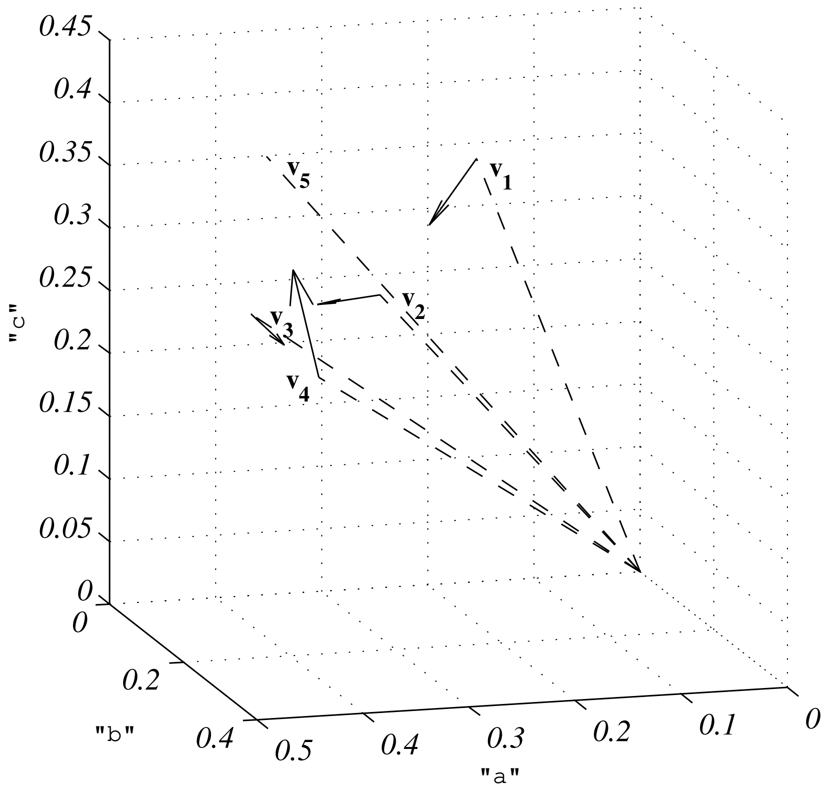


Figure 3.11: **Term vector drift** One way to visualize changing document content is to imagine the term vector representation as a dynamic entity. For a simple language of only three words, "a", "b", and "c", this figure denotes a sequence of changes to that document over a period of time. The lines from the origin are the term vectors $\mathbf{v_k}$, and the arrows between these are the successive change vectors, $\mathbf{v_k} - \mathbf{v_{k-1}}$. The length of these change vectors can indicate the degree of a shift in a document's main focus. The same information can be inferred from the angle between successive term vectors. It can also be interesting to consider the limiting behavior of $\mathbf{v_k} - \mathbf{v_1}$, or the sequence of vector differences between an observation and all observations that follow it; this shows how change accumulates over time.

3.2.2 Gauging change magnitude using term vectors

Term vectors can be used to measure textual change in (at least) two ways. First, the angle between successive term vector representations can be used to show the magnitude of a change. Second, the component-wise vector difference, when suitably normalized, conveys similar information. Using these two measurements, we will demonstrate that large textual changes are relatively rare, and large relative changes are much more common for very small documents. The implications for a search engine are that many changes are small and can probably be ignored. The prevalence of small changes further reinforces the idea of storing a document's history using delta compression[7], as was investigated with application to caching in [MDFK97].

The differences discussed in this section are understood to be between two instances of the same URL. We represent these two observations as term vectors $\mathbf{v_1}$ and $\mathbf{v_2}$, where the subscript denotes their relative positions in a time sequence. We adopt the convention of not normalizing the term vectors, so that each component represents the (integer) number of occurrences of that word in the document. Large magnitude vectors therefore correspond to long documents.

Cosine of the angle between term vectors

The first technique of comparing two documents is to measure the angle θ_{12} between the two term vectors. For the convenience of having a measurement on $[0, 1]$, the cosine of this angle is often used instead. It is calculated as

$$\cos(\theta_{12}) = \frac{\mathbf{v_1} \cdot \mathbf{v_2}}{||\mathbf{v_1}|| \; ||\mathbf{v_2}||}. \tag{3.1}$$

In this interpretation, when $\cos(\theta_{12}) = 1$, the vectors have identical relative term frequencies (although they may have different magnitudes), and $\cos(\theta_{12}) = 0$ indicates that the documents have no terms in common. We can use $\cos(\theta_{12})$ to study the amount of change that has taken place between two instances $\mathbf{v_1}$ and $\mathbf{v_2}$ of the same document. To make this a more intuitive measure of change magnitude (rather than a similarity measure) we will work with $1 - \cos(\theta_{12})$ instead.

[7] Delta compression is the practice of serving differences with previous versions of a resource rather than providing the entire resource again. This is the idea behind UNIX `patch` distributions for source code version updates.

For the subset of our collection under scrutiny, we have plotted the joint distribution of $1-\cos(\theta_{12})$ and $\|\mathbf{v_1}\|$ in Figure 3.12. This figure (as well as Figures 3.13 and 3.14) has three parts: a scatter plot indicating the joint distribution of the two variables in question, and a marginal CDF for each variable in the joint distribution. The CDF on the left only includes observations on which a change occurred, while the CDF on the bottom includes all observations.

Notice from Figure 3.12 that most large-angle differences occur in smaller term vectors. The larger this angle between two observations, the more likely it becomes that the document is miscategorized after a change goes unobserved. From the marginal CDF at the left, it is apparent that 90% of differences have $(1 - \cos\theta_{12}) < 0.05$. The curvilinear features near the origin correspond to document changes of only one or two words.

Vector difference

The second measure of change size is the magnitude $\|\mathbf{v_1} - \mathbf{v_2}\|$ of the vector difference between pairs of observations. We show the joint distribution of this statistic and $\|\mathbf{v_1}\|$ in Figure 3.13. The diagonal feature in this graph represents documents that went from some "normal" status to either an error state (such as not found, server error, document moved) or some other very short document, since the magnitudes of these changes are roughly equal to the magnitudes of the original documents. Conversely, the vertical feature on the far left represents the opposite type of change, where very short documents became much longer.

The strong feature along $\|\mathbf{v_1}\| \approx \|\mathbf{v_2} - \mathbf{v_1}\|$ suggests that a normalization by $\|\mathbf{v_1}\|$ might be helpful, especially for getting a picture of what the population looks like along that diagonal. Similarly, when $\mathbf{v_2}$ is larger, normalizing both vectors in the difference by $\|\mathbf{v_2}\|$ would scale the dense section along the left side of Figure 3.13 into a smaller range. This is desirable in order to map all of the large changes into a single, small region. To draw out these characteristics, the vector difference is scaled: both $\mathbf{v_1}$ and $\mathbf{v_2}$ are normalized by the larger of their two magnitudes, before taking their vector difference[8]. This guarantees that the magnitude of the vector difference will be bounded by [0,2], since both vectors will be within an n-dimensional sphere of unit radius (for a

[8]This is equivalent to scaling the difference, but doing the normalization first makes the bound [0,2] more intuitive.

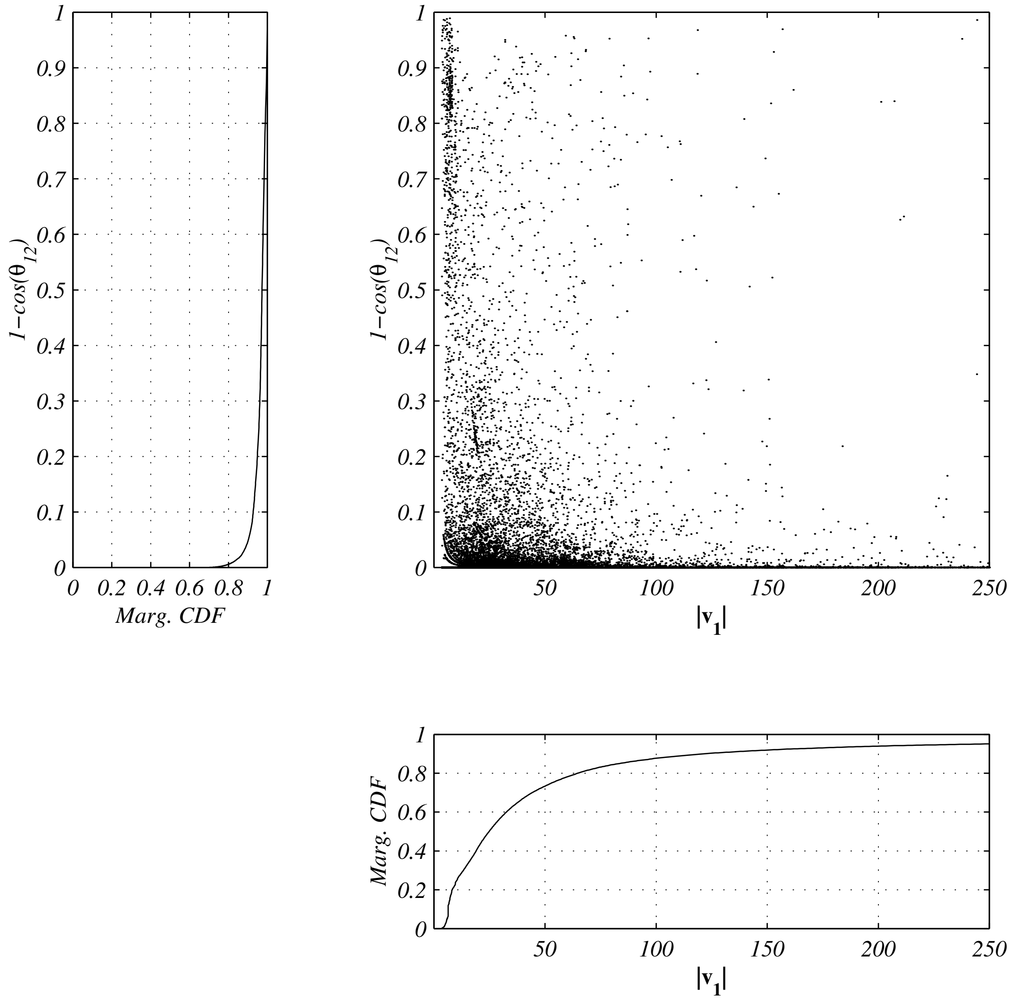


Figure 3.12: **Magnitude $\|\mathbf{v}_1\|$ of original vector vs.** $1 - \cos(\theta_{12})$**.** At the upper right, we show a scatter plot that illustrates how the magnitude of the vector in question is related to the angle between two instances of a document's term vector. Each dot is a single observed change, thus the marginal distribution on the left only includes observed changes.

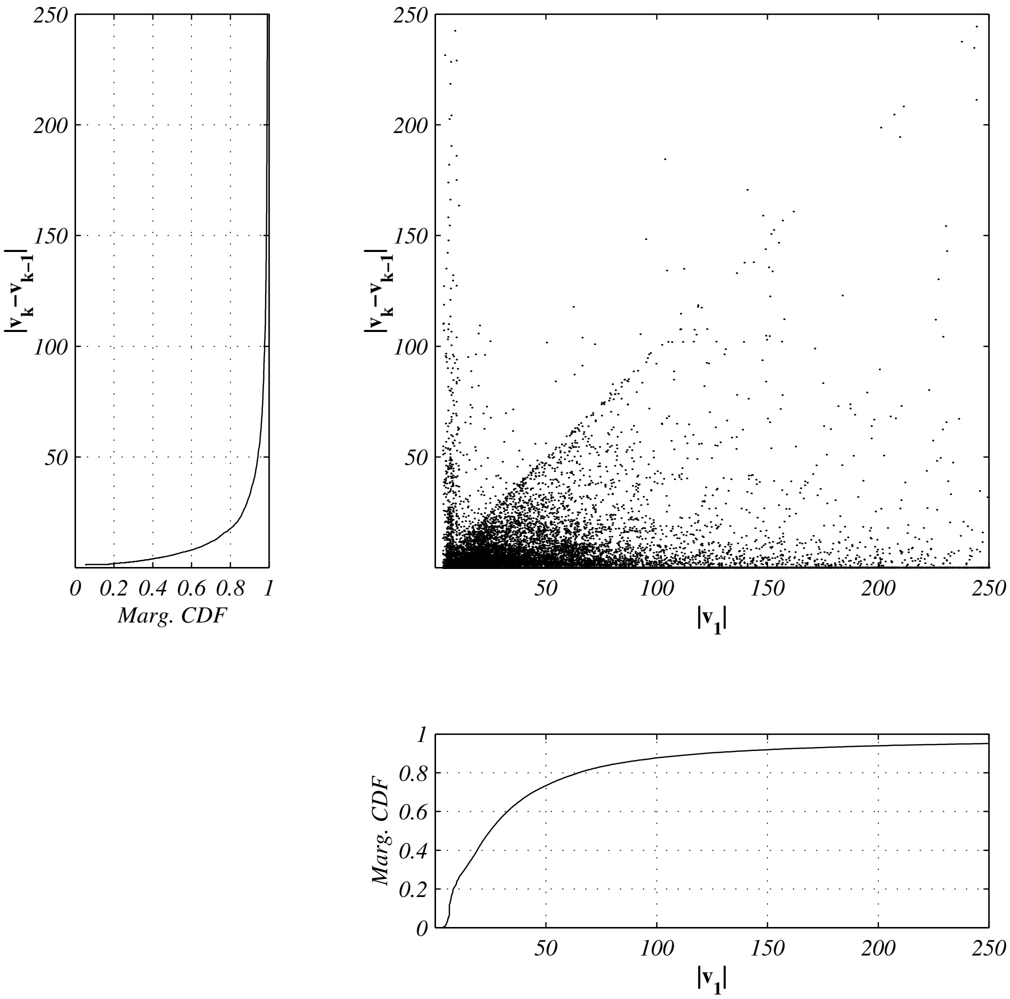


Figure 3.13: **Magnitude $\|\mathbf{v_{k-1}}\|$ of original vector vs. magnitude $\|\mathbf{v_k} - \mathbf{v_{k-1}}\|$ of change vector.** At the upper right, the scatter plot shows how the magnitude of the original term vector is related to the magnitude of the the change vector. The linear features in the plot are explained in the main text.

term space having n possible words), inside of which the maximum distance between vectors is 2. The statistic so obtained is a relative change magnitude. A scatter plot of this statistic, shown in Figure 3.14, shows how the diagonal feature in Figure 3.13 has been mapped onto a horizontal line at unit magnitude. Additionally, much of the vertical stripe on the left hand side of Figure 3.13 is re-mapped into a tight region also at unit magnitude. The dense horizontal feature at unit relative difference represents "disappearance" or "appearance" events. These are wholesale changes, where the document went from being very short to very long, or vice versa. Notice from the density of points at unit relative difference and $\|\mathbf{v_1}\| \approx 10$ that this type of change is most common for very short documents. The marginal CDF at left shows that "wholesale" changes are not more than about 5% of all changes. Last, the curves near the origin represent very small changes, where just a few words differ.

3.2.3 Mapping document changes onto user interests

The only truly relevant changes in a document are the ones that users of the index care about. That is, a document's content is user-specific, so it is always necessary to assess what portion of a page's content is of interest. Applied to a single user who only runs queries on "`Bayesian belief networks`", a term vector-based retrieval system need not be so careful about tracking changes in the usage of other words in the language. Of course, even single users have needs that are far more complex than the particular words chosen for the query—of equal importance are the words the user *might* have chosen instead. Just as tests are applied to determine if documents are of interest, tests are needed to determine if a document's changing is of interest as well.

For this reason, the term vector representation of a document should be skewed to reflect the relative importance of terms that are used in queries, just as was originally suggested by [EW61] for rescaling document terms by their frequency. Query keywords are the most direct representation of user interest that exists, aside from perhaps actually knowing the pages that most interest users. Certainly query term frequency could be used to build much smaller indices. The small set of search keywords entered by Informant users contains only around 19,000 unique words (from about 57,000 queries), as compared to the more than 1 million unique words occurring in our web document test

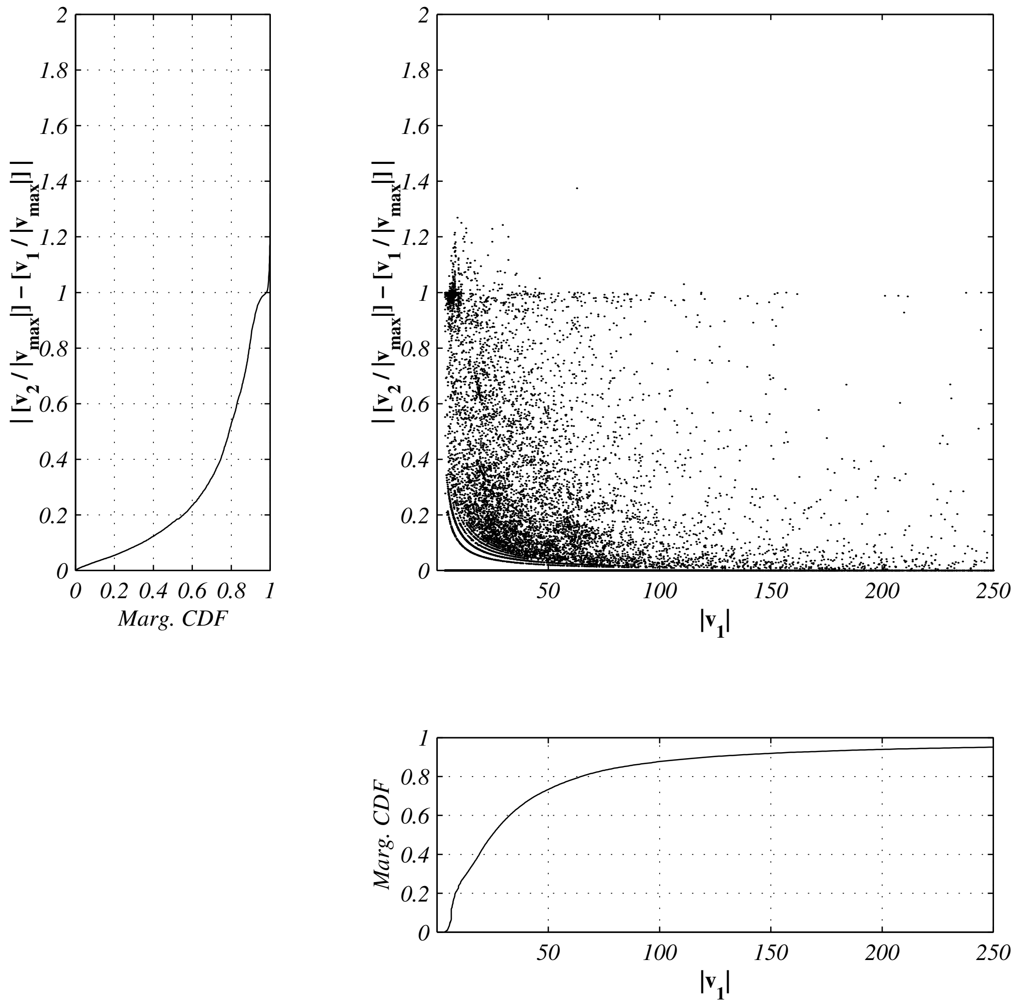


Figure 3.14: **Magnitude $\|\mathbf{v_1}\|$ vs. change magnitude for normalized $\mathbf{v_2}$ and $\mathbf{v_1}$.** Here, $\|\mathbf{v_{max}}\|$ is the larger of $\|\mathbf{v_1}\|$ and $\|\mathbf{v_2}\|$. In this representation, the linear feature boundaries have all been collapsed into the region at unit relative change magnitude. The CDF on the left thus shows that "wholesale" changes are not more than about 5% of all changes. Moreover, this type of change is most common for the smallest documents; in this case the wholesale change generally means the document is changing from some error state back to a normal one.

set. While it is certainly true that a larger sample of queries would yield a larger set of words, many words that appear in documents will never be used in queries. For example, verbs are comparatively rare in queries as compared to nouns (or noun phrases) and adjectives. It is reported in [Nic97] that the Lycos[9] search engine uses a "reduced" index image of the top 100 words to represent a page, although it is quite possible that has changed in the three years since. User query terms suggest a different way of forming such a reduced index.

Query terms vs. web document terms

The term vectors used earlier in this section were not rescaled in any way to account for the importance of terms within a document. So, occurrences of common words like "`the`" counted exactly the same as occurrences of rare words like "`actinophage`". Additionally, all word changes were noted, not just the words that were used in queries. Fortunately, our work with the Informant has allowed us to gather not only a list of common query terms, but also a list of common terms within documents in general.

Since terms used in queries define in some way what terms are most relevant in documents, it is important to understand the difference between query vocabulary and general web document terms. As mentioned earlier, the words in each set are certainly different: the query word list is nothing like average text. The words in queries almost form a language unto themselves, which has statistics that reflect the fact. For example, many languages exhibit power-law scaling in word frequency; figure 3.15 shows three probability densities that illustrate this point. The x-axis represents the number of occurrences of a term in a corpus (each of the three lines plotted is such a corpus), and the y-axis is the probability density corresponding to that particular count. First, we plot the probability density for all words appearing in our short-duration (45-day) collection of term vectors. Notice the strong power law relation, very much evocative of a Zipf-type distribution of word rankings [Zip49][10]. Second, we show the same type of distribution, except for words appearing in Informant queries. This is a much smaller sample (30,000 users with roughly 1.9 queries each), hence the plot does not extend to occurrence counts higher than around the 1,000 range. The words used in

[9] http://www.lycos.com/

[10] This is not the same method used in generating Zipf's Law distributions; we revisit this topic again in an Appendix.

queries do not appear with the same frequency in queries as they do in the general population, as is shown by the third line in the Figure, which is the probability density of occurrences of query terms within the general population. It is clearly not a power-law type relation; the bump-like feature in Figure 3.15 is suggestive of ideal curves for "resolution power" [vR79] of query terms. Just as extremely common words are not helpful in finding information, extremely rare words are not as helpful either. Common words are contained in far too many documents, while rare words may not be in the index at all. Thus users tend to select words having intermediate frequency instead of words at the endpoints. Perhaps this is the "principle of least effort" [Zip49] in action, as query term selection is being pulled towards an optimal curve through human learning of the language of search engines. The point of this exercise is to assert that rescaling by term frequency is not the same as rescaling by term frequency in queries. If optimizing an index to accurately rank the importance of changes, this difference must be taken into account.

Interestingly, the Zipf-type power law distribution appears not only for query terms and words in web documents, but also for unique hyperlinks (`HREF`'s) and hostnames (like `cnn.com`) in our observed document set. Though not directly relevant to the topic of dynamics, these plots are included for the interested reader in the Appendix.

Just as we suggest rescaling term frequency with respect to usage in queries as a means for improving the performance of retrieval algorithms, this also helps in defining what is a "meaningful" change to a web page. Terms with greater resolution power, or those that appear to be used more often in queries, should be scaled so that a change in their frequency is deemed more worthy of requiring that a document be re-indexed. Text-based change filtering and classification is a large topic and is a subject left for future work. We mention it merely to note that the whole subject of change detection for web pages is only part of the larger problem of satisfying user requests for information. Other observation domains will exhibit similar challenges in state definition and determining what changes are important in proper context.

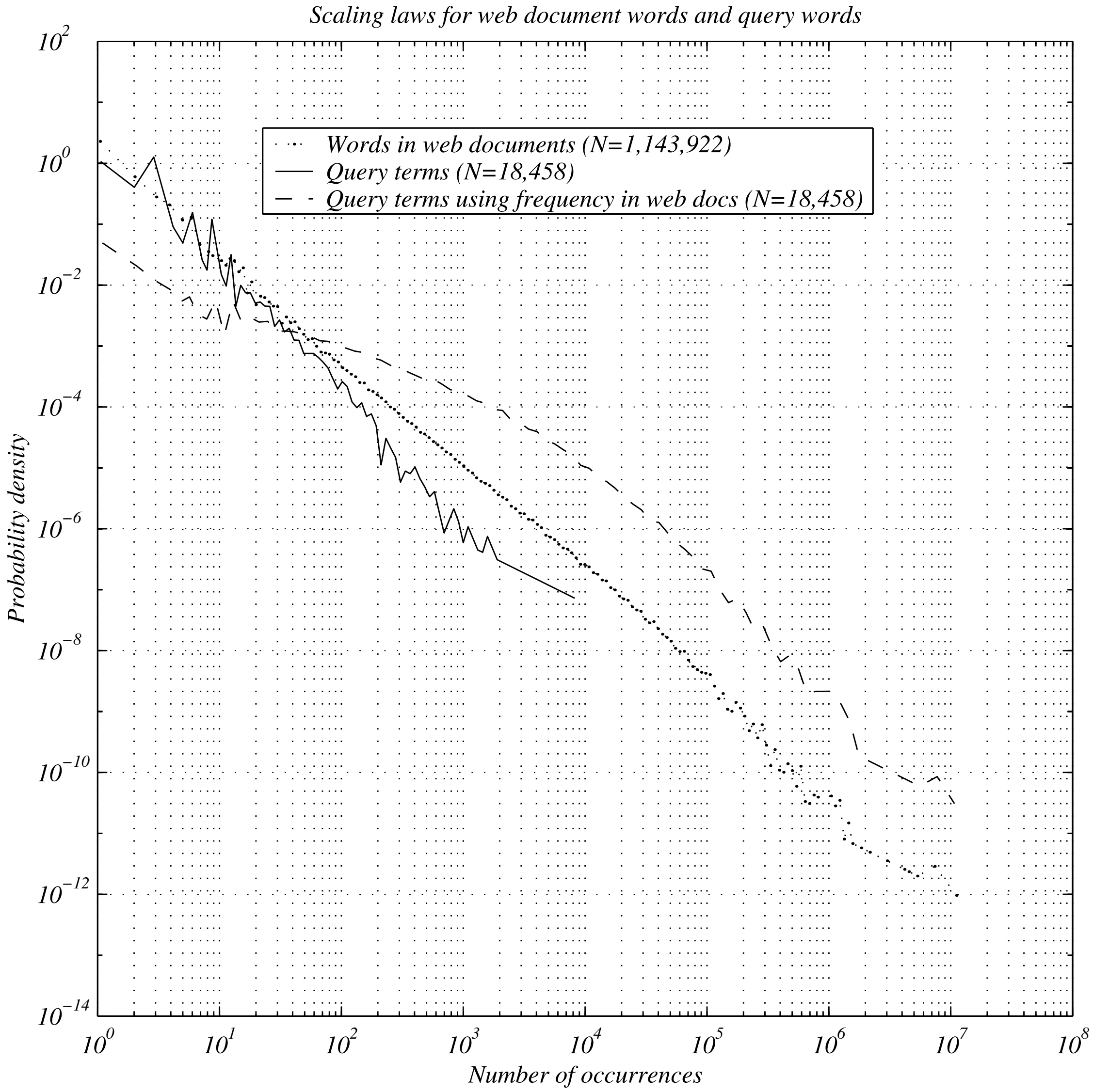


Figure 3.15: **Probability densities of query and document term frequencies.** In this graph we show three probability densities on a log-log scale. The presentation takes a moment to grasp. The horizontal axis is the count (or number of occurrences) for words in each list, and the vertical axis is the probability density corresponding to each count. So for each corpus of text, the most likely count (by far) is a single occurrence, hence the maximum at 1. The similar power-law scaling of the set of query terms and the set of normal web document terms suggests that query terms are something of a language unto themselves. The dashed line, formed by taking the frequency of query terms in web documents, shows that query terms are chosen for better indexing power and are not representative of ordinary language. We present this to emphasize that change detection for web pages must be sensitized to user requests.

3.3 Estimating the change rate distribution for the web

We expect it will be difficult to choose a representation that is sufficiently sensitive to catch important changes, while being restrained enough not to over-react to minor ones. Still, to move forward in our characterization of the rate of change of web pages, a baseline characterization is needed. For this we will use the simplest definition of change, namely, any alteration is considered a change. This has the advantage that it overestimates the rate at which *important* changes happen, however importance is defined. To this end, we will rely on server-provided `Last-Modified` timestamps for these change statistics. From these timestamps we can make both age and lifetime observations. We consider the estimation problem using each kind of observation in turn.

3.3.1 Age-based estimation

It is clear from the empirical page age distribution shown in Figure 3.8 that the majority of web pages are young. The origins of this trend are less clear. Different explanations can give rise to the same observed age distribution. If we propose to associate age with dynamics, as is done when setting cached documents' time-to-live (TTL) values to a percentage of the most recently observed age ([Cat92], [BDH+94], [CDN+96]), it is necessary to understand the nature of an age distribution in a growing population. On the one hand, a fixed population of pages whose change times are governed by identical exponential PDF's will produce an exponential age distribution when sampled collectively, as in (2.18). At the other extreme, an exponentially growing population of web pages in which changes are rare or even nonexistent will be skewed towards youth as well—there will be exponentially more pages in one generation relative to the previous generation. These kinds of dynamic and growing populations can be visualized very nicely using models like (2.14); we consider a very simple case here.

Consider two very different models for the web. First, an exponentially-growing population of completely static web pages will produce an exponential distribution of observed page ages. To see this, note that the population at time t is given by an expression of the form $P_0 e^{\xi t}$ where P_0 is the initial population and ξ is the exponential growth rate parameter. An age distribution at time τ

can be formed by reversing the sense of time, and normalizing by the population size:

$$g_{growing}(t,\tau) = \begin{cases} \frac{\xi e^{-\xi t}}{1-e^{-\xi\tau}} & t \in [0,\tau] \\ 0 & t \notin [0,\tau] \end{cases} . \tag{3.2}$$

This distribution will approach an exponential density with parameter ξ as τ gets large.

But an exponential distribution of page ages can arise for completely different reasons. Consider a fixed-size group of identical pages, each of which changes at time intervals governed by an exponential distribution. Each page undergoes many changes, with each change returning that page to age zero. Such a population also gives rise to essentially an exponential age distribution (see 2.18). In particular, the age distribution for such a population is

$$g_{dynamic}\,(t,\tau) = \begin{cases} \lambda e^{-\lambda t} & t \in (0,\tau) \\ \left(e^{-\lambda\tau}\right)\delta(t-\tau) & t = \tau \\ 0 & t \notin [0,\tau] \end{cases} . \tag{3.3}$$

As the time since the population's birth, τ, becomes large, the distribution of observed page ages will also approach an exponential distribution and will be hard to distinguish from that of a growing population of unchanging web pages. The hybrid model we present next represents the middle ground—the web is growing *and* pages change according to exponential time distributions.

We now combine the effects of web growth and page change dynamics. The web has been growing for several years so that the time since creation of web pages is distributed approximately exponentially:

$$h(t_c) = \xi e^{-\xi t_c}. \tag{3.4}$$

where ξ is the growth rate and t_c is the time since creation of a page. We emphasize that t_c is not to be confused with our definition of the page's age, since age refers to the time since the last modification.

For an exponentially-growing population of dynamic pages, each of which has an exponential age distribution as described by (3.3), the aggregate age distribution $g(t,\lambda)$ will be a weighted average over time since creation, weighted by the number of pages created at the same time. Specifically,

$$g(t,\lambda) \quad = \quad \int_0^\infty g(t,\lambda,t_c)h(t_c)dt_c \tag{3.5}$$

$$
\begin{aligned}
&= \int_0^\infty \xi e^{-\xi t_c} e^{-\lambda t_c} \delta(t - t_c) dt_c + \int_0^\infty \xi e^{-\xi t_c} \left[U(t) - U(t - t_c)\right] \lambda e^{-\lambda t} dt_c && (3.6) \\
&= \xi e^{-(\xi+\lambda)t} + \int_0^\infty \xi e^{-\xi t_c} \lambda e^{-\lambda t} dt_c - \int_0^t \xi e^{-\xi t_c} \lambda e^{-\lambda t} dt_c \\
&= \xi e^{-(\xi+\lambda)t} + \lambda e^{-\lambda t} - \lambda e^{-\lambda t} \left(1 - e^{-\xi t}\right) \\
&= (\xi + \lambda) e^{-(\xi+\lambda)t}. && (3.7)
\end{aligned}
$$

This means that the age distribution of an exponentially growing population of objects with (identical) exponential age distributions remains exponential, with parameter given by the sum of the population growth and page change rate constants.

The age distribution for the entire population (namely the whole web) is yet another mixture, in which we take expectation of (3.7) with respect to a joint distribution of growth rate ξ and change rate λ. For simplicity we use the same growth rate for all change rates. Using a distribution $w(x)$ over the inverse rate $\lambda = 1/x$, with a uniform growth rate ξ, we express the mixture as

$$
g(t) = \int_0^\infty \left(\xi + \frac{1}{x}\right) e^{\left(\xi + \frac{1}{x}\right)t} w(x) dx. \tag{3.8}
$$

The only factor remaining before this distribution can be matched to the data is the shape of the distribution $w(x)$ of inverse change rates. In our initial development, we use a generalized exponential (Weibull) distribution over the inverse change rate (which is also the mean change time), such that

$$
w(t) = \frac{\sigma}{\delta} \left(\frac{t}{\delta}\right)^{\sigma - 1} e^{-(t/\delta)^\sigma}. \tag{3.9}
$$

where δ is a scale parameter and σ is a shape parameter. See [MR94] for a discussion of Weibull distributions, or [Fel71] for a more general discussion of this family of exponential distributions. The shape parameter can be varied to change the shape from a very sharply-peaked distribution (for $\sigma < 1$) to an exponential (for $\sigma = 1$), to a unimodal distribution with maximum at some positive t (for $\sigma > 1$). The scale parameter δ adjusts the mean of the distribution.

To determine what values of ξ, σ, and δ best model our age observations, we numerically evaluate (3.8) at a number of ages t. This is used to estimate the cumulative age distribution $G(t)$ at N

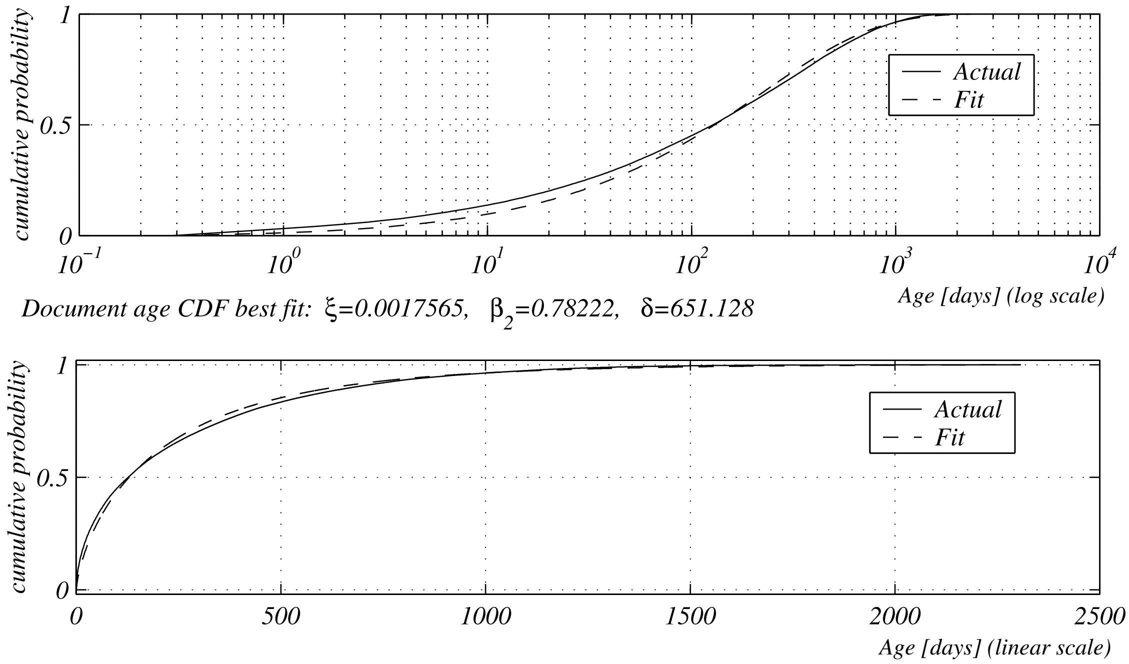


Figure 3.16: **Best-fit age CDF.** These plots show the distribution which results from a numerical optimization of (3.10), yielding the values $\xi = 0.00176$ (growth rate), $\sigma = 0.78$ (shape parameter), and $\delta = 651.1$ (scale parameter). The top plot uses a log scale to show the deviations in the fit for small age. The minimization was carried out using linearly-spaced points.

points t_i. These estimates, $\hat{G}(t)$, are compared with samples from the empirical distribution $G(t)$ (as diagrammed in Figure 3.8) at points t_i. A sum of the squared error over all sample times t_i provides a scalar error function of the vector (ξ, σ, δ). This error function

$$SE_{age}\left(\xi, \sigma, \delta\right) = \frac{1}{N} \sum_{i=1}^{N} \left(\hat{G}(\xi, \sigma, \delta, t_i) - G(t_i)\right)^2 \tag{3.10}$$

can be minimized numerically, whereupon the optimal values are found to be $\xi = 0.00176$, $\sigma = 0.78$, and $\delta = 651.1$. The idea in this fit is to minimize the variance of the residuals. The fitted age distribution is shown in Figure 3.16. These parameters imply a steeper-than-exponential age distribution (since $\sigma = 0.78$) and a growth rate that implies a doubling time of around 390 days. This is not unreasonable, as [LG98] estimated a lower bound size of 320 million pages in December

1997, which increased in [LG99] to 800 million pages by February 1999. This would imply a growth constant over the 14 months of $\xi = 0.0022$, or a doubling time of 318 days. However, the difference in these estimates tells us to proceed with caution, understanding that estimates based on these results are somewhat uncertain. Moreover, the assumption of exponential growth in the number of *documents* is based on assertions of exponential growth in the number of web *hosts* (as in [Gra97a] and [ISC99], for example). Growth rates have slowed appreciably, especially in the last year; lifetime-based estimation methods will prove more reliable.

3.3.2 Lifetime-based estimation

As mentioned previously, inferring change rates from observed lifetimes is somewhat tricky, since an observed change may only be the most recent of many changes that took place since the last observation. Moreover, changes that take a long time to happen are inherently more difficult to catch. For example, if one were to watch a calendar for three consecutive days, waiting for the month to change, there is a good chance that this event will not be observed. However, as the timespan gets longer it becomes more probable that a change will be seen. In the same way, it is necessary to account for the probability of observing a change, given the timespan of observation.

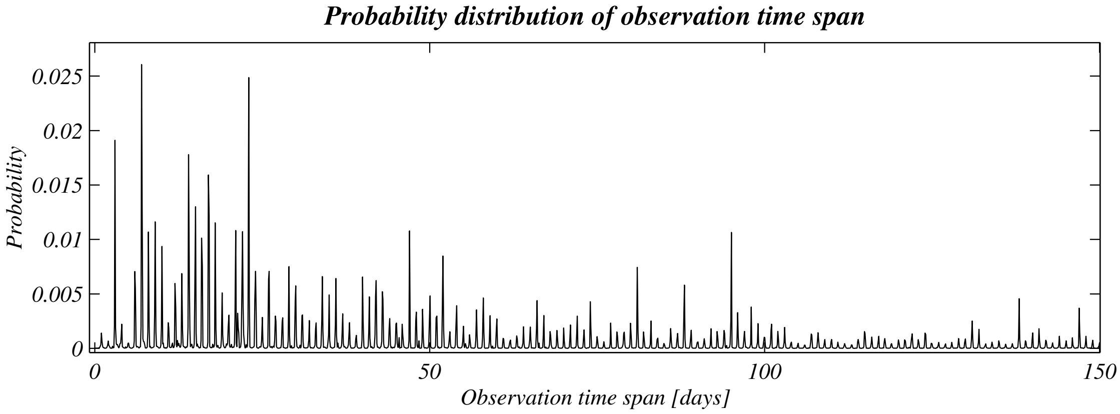
For a page that changes exponentially at rate λ, the probability that at least one change will be observed within a timespan τ is

$$\Pr(\text{change observed}|\tau, \lambda) = 1 - e^{-\lambda\tau}. \tag{3.11}$$

The pages in our collection are observed over many different timespans τ. Therefore, to determine the probability of observing changes for pages having change rate λ, we assume that change rate and timespan are independent, and weight (3.11) with respect to the probability of all possible observation timespans τ_i (discretized):

$$\Pr(\text{change observed}|\lambda) = Z_{bias}(1/\lambda) = \sum_{i=1}^{i=N} \Pr(\tau_i)(1 - e^{-\lambda\tau_i}). \tag{3.12}$$

Possible timespans τ_i are distributed as shown in Figure 3.17. Combining this data with (3.12) allows us to compute Z_{bias}, which in turn allows us to weight each mean lifetime's probability of



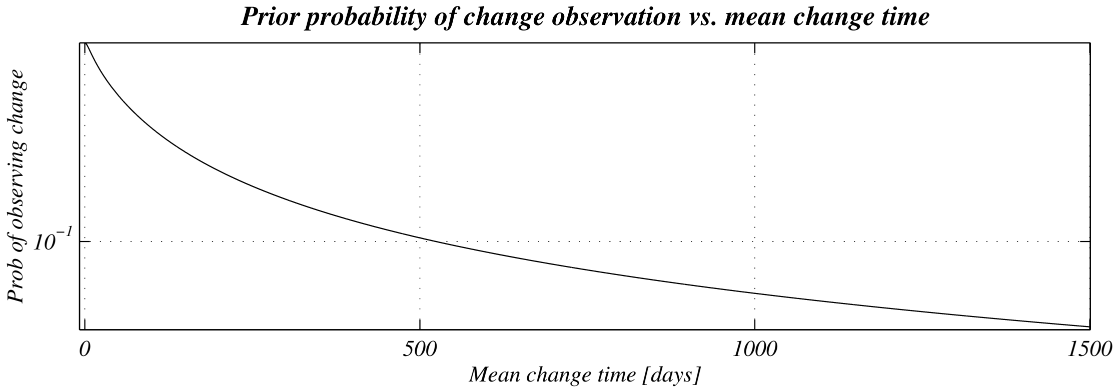
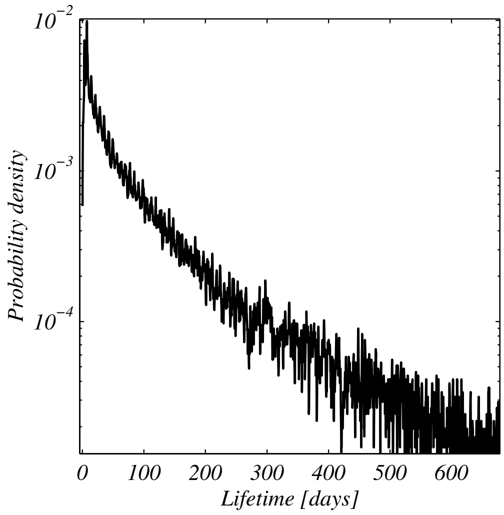


Figure 3.17: **Observation time distribution and induced finite time span bias.** The top plot shows the distribution of observation time spans, or the time difference between the first and last observation timestamps for individual pages. The spikes appear in this graph because we only run our checks at night, so timespans tend to cluster around 24-hour intervals. Using (3.12), these timespans translate into the probability of any mean change time being represented among our observed web page changes.



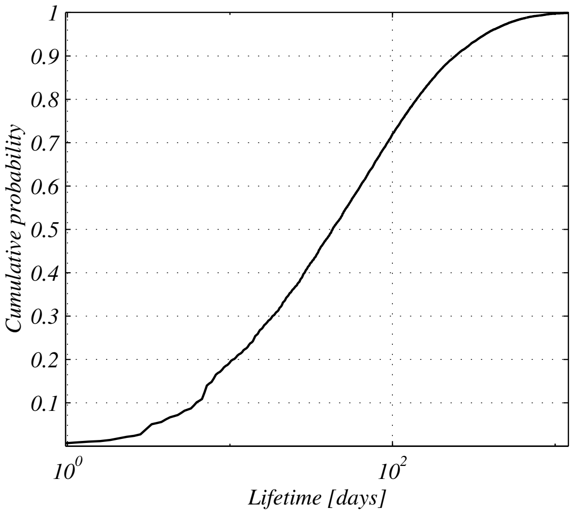


Figure 3.18: **PDF and CDF of observed lifetimes.** On the left, a rescaled histogram approximates the PDF of observed lifetimes, or differences in successive modification timestamps. On the right, we show the corresponding CDF.

being among the observed data. Put another way, the distribution of change rates sampled in our experiment is not the true rate distribution, but rather one that is weighted by (3.12). If the *actual* density of mean lifetimes is $f_{mean}(t)$, then the *observed* density of mean lifetimes is

$$f'_{mean}(t) = \frac{f_{mean}(t) Z_{bias}(t)}{\int_0^{\infty} f_{mean}(t) Z_{bias}(t) dt}. \tag{3.13}$$

These mean lifetimes are only seen through a mixture of exponential distributions, so the observed lifetimes should approximate the probability density

$$f_{observed}(t) = \int_0^{\infty} \lambda e^{-\lambda t} f'_{mean}(1/\lambda) d(1/\lambda). \tag{3.14}$$

As with the age-based estimates, we can form a mean squared-error function like (3.10) and fit the CDF corresponding to (3.14) to the observed lifetime distribution. We show the distribution of observed lifetimes in Figure 3.18. Using $F(t)$ as the cumulative lifetime distribution, and $\hat{F}(\sigma, \delta, t)$

as the estimator, the error function is

$$SE_{lifetime}(\sigma, \delta) = \frac{1}{N} \sum_{i=1}^{N} (\hat{F}(\sigma, \delta, t_i) - F(t_i))^2 \tag{3.15}$$

As before, we use a Weibull density (3.9) for the distribution of inverse rates (mean times) $\bar{t}$. This results in an error surface having a minimum at $(\sigma = 1.4, \delta = 152.2)$. An intensity plot of (3.15) is shown in Figure 3.19. The CDF and its optimal estimator are overlaid in Figure 3.20, and the error in this fit is magnified in Figure 3.21. Using our estimates, the *mean* lifetime PDF and CDF are shown in Figure 3.22. Our mean change time of 138 is much higher than those previously published. Kahle [Kah97] claims 75 days, and has a very large sample size. Few details are given in the article regarding sampling techniques, popularity bias, and so forth. In particular, it is unclear whether the 75 day estimate is an average of all observed lifetimes (where single pages contribute one term to the average corresponding to each instance of the page), or only one contribution per page. Prior to this, as stated in [Pit98], estimates had been as low as 45 days. It is not unreasonable to expect that the aging of the web's older documents raise the average lifetime by their very existence. We do expect, however, that we have underestimated web pages' rates of change by virtue of the fact that these estimates are derived from a less volatile subpopulation (namely those which provide timestamps).

Regardless of the veracity of these estimates for the entire web, they certainly do an excellent job of describing this population. The quality of the lifetime-based estimates differs substantially from that of the age-based estimates, as can been seen by comparing the fit in Figures 3.20 and 3.16. There are two reasons for the difference. First, the assumption of exponential growth used for the age-based estimation is probably a poor one, as true growth is much slower. Forcing exponential growth on a more slowly growing population causes the dynamics to be under-represented, driving our estimates away from their true value. The lifetime-based estimation is not perfect either, as change rates may not be independent of observation timespan. A change in a page might very well push it into or out of a user's set of search results. We count on the fact that in observing faster than the search engines, we can observe changes before those changes force a result from the top of the list. It is difficult to justify an assumption of any particular dependence, since this relationship is

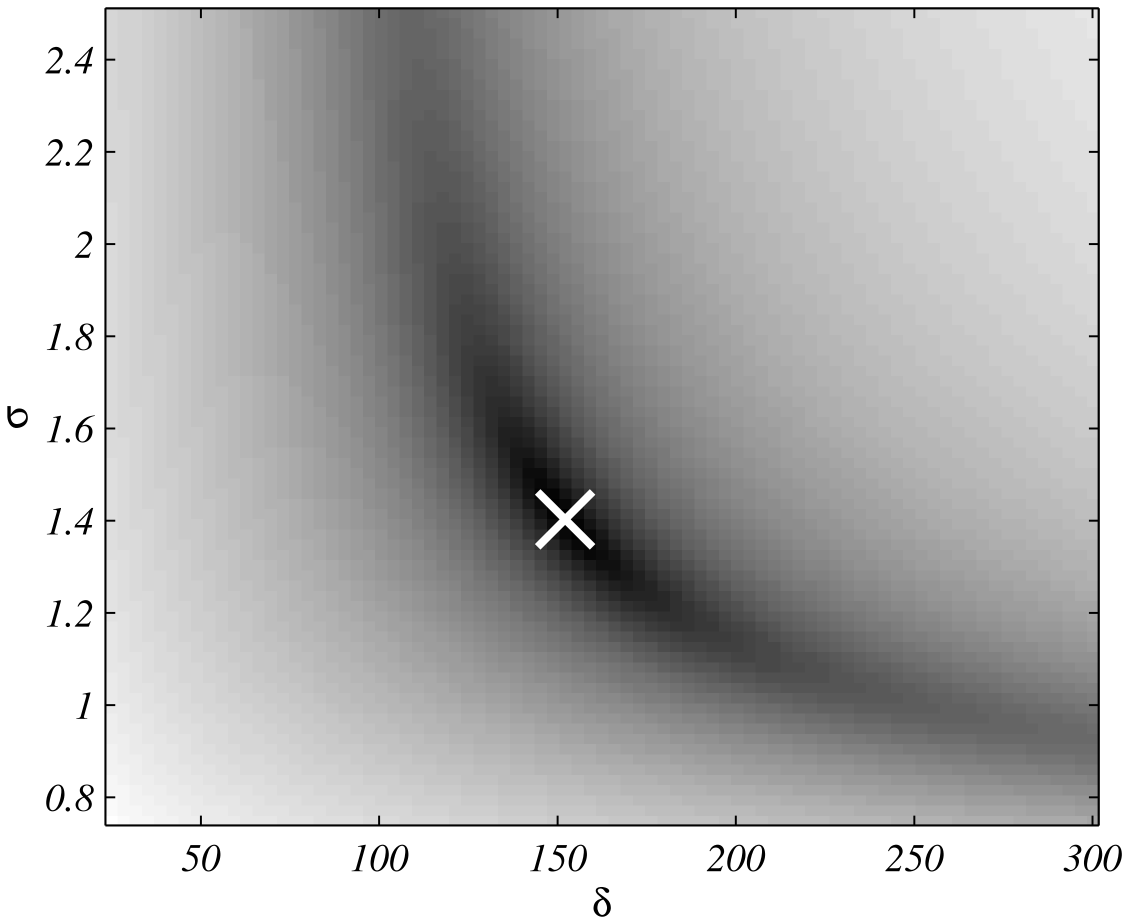


Figure 3.19: **Intensity plot of mean squared-error.** The minimum of (3.15) in the space of shape parameters σ and scale parameters δ is marked by a white "$\times$" in the center of the dark patch, at ($\sigma = 1.4, \delta = 152.2$). The error function appears to be unimodal.

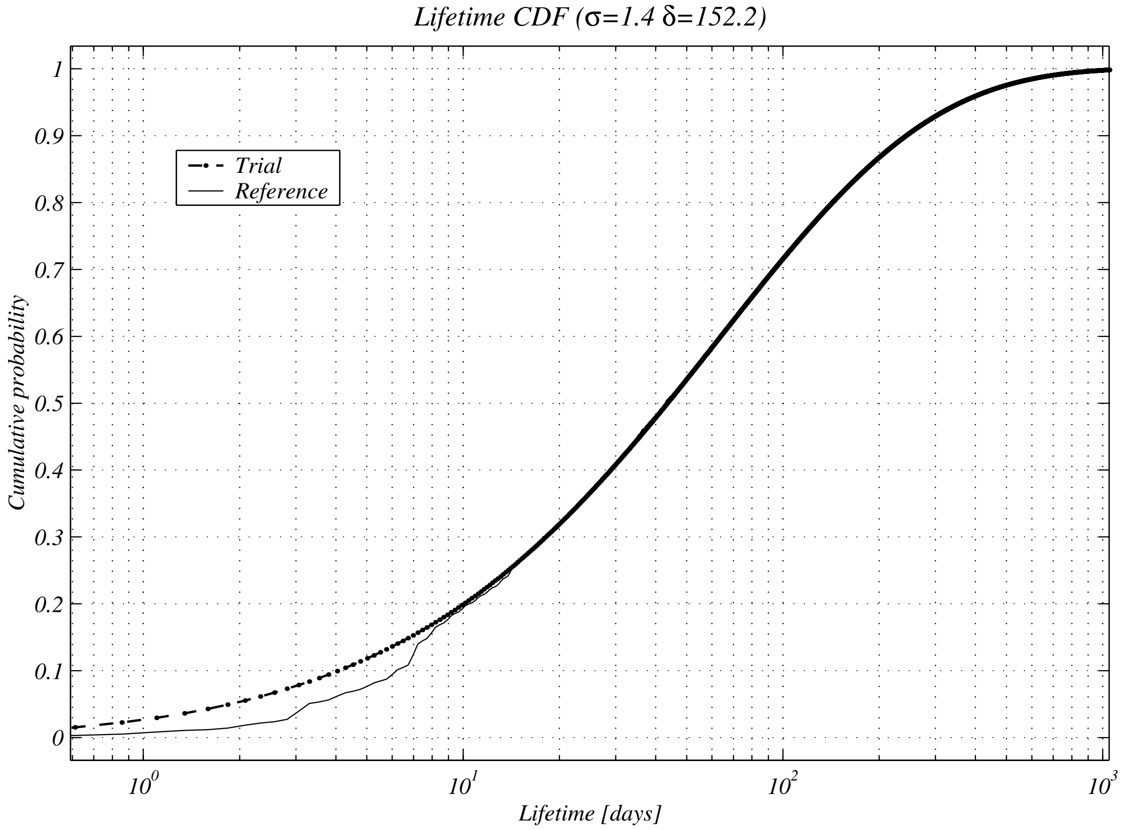


Figure 3.20: **Overlay of model lifetime CDF and observed CDF.** The minimum error distribution (marked "Trial" above) found by minimizing (3.15) is shown along with the observed lifetime distribution (marked "Reference"). Errors in the fit below around 8 days are due to aliasing, where multiple changes are masked and treated as a single, larger lifetime. Our estimates are only extrapolations in this region and may be inaccurate. The region above 8 days is an extremely precise fit; the two curves are nearly identical. For an even better fit, points below the aliasing bound should not be considered in evaluating the error function.

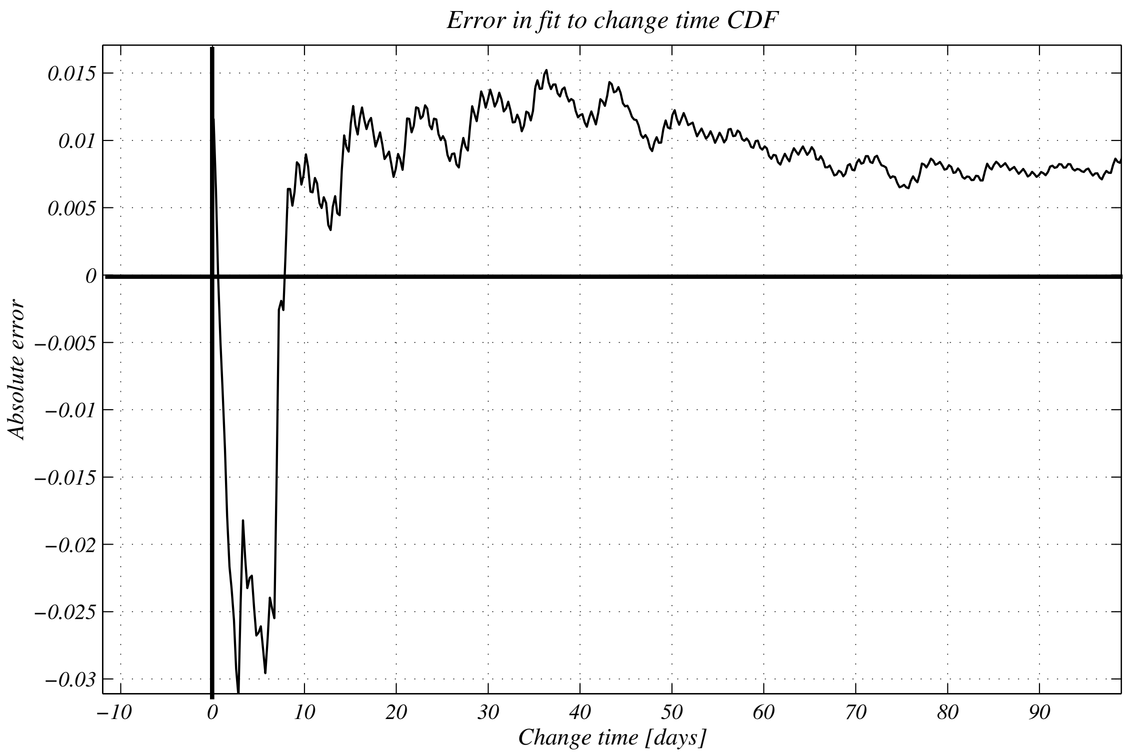


Figure 3.21: **Absolute error $\hat{F} - F$ vs. lifetime.** These are the errors in the fit shown in Figure 3.20; we have used a linear scale and just show the leftmost region. Note the large errors below around 8 days due to aliasing. The effect of diurnal and weekly trends, as plotted in Figure 3.1, is clearly visible in the long and short period ripples. As mentioned in Figure 3.20, slight improvements in our estimates could be had if we restricted the fit to samples above 8 days.

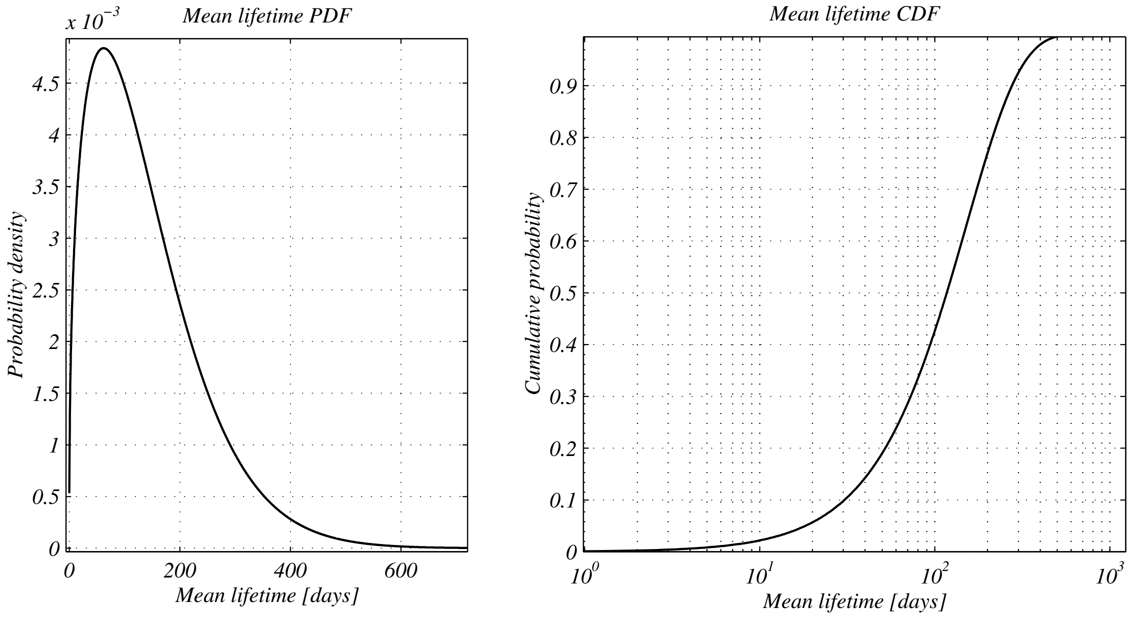


Figure 3.22: **Mean lifetime** $(1/\lambda)$ **estimated PDF and CDF.** Our lifetime-based population parameter estimation implies these distributions of mean lifetimes for the documents observed by the Informant. Note that these *mean values* are to be distinguished from the distribution of *observed* lifetimes. The average is around 138 days, the most likely value is 62 days, and the median is 117 days.

controlled by many unknown factors (re-indexing time and result ranking strategy for search engines, for example).

3.4 (α, β)-currency for web search engines

3.4.1 Probability of β-currency for a theoretical collection

Next, we apply these estimates to the problem of assessing the (α, β) currency of a search engine's index when that engine re-indexes its collection on a fixed-period schedule. Choosing a random object to which we apply (2.40) is equivalent to selecting a value for λ. In our collections, as discussed earlier, we have observed that the mean time $\bar{t}$ between changes roughly follows a Weibull distribution, (3.9), which is given by

$$w(\bar{t}) = \frac{\sigma}{\delta} \left(\frac{\bar{t}}{\delta} \right)^{\sigma - 1} e^{-(\bar{t}/\delta)^{\sigma}}. \tag{3.16}$$

The change rate λ is the inverse of the mean time between changes, so we can replace λ in the integral with the change rate $1/\bar{t}$.

Using (3.16), along with the parameter values that resulted from our numerical optimization, we can determine the expected value of (2.40) over λ for our collection. This calculation for other collections or other demand distributions depends only on finding the distribution $w(\bar{t})$ of mean change times for those collections. This initial analysis uses a simple periodic, round-robin re-indexing schedule, where the revisitation time T is the same for all sources. We propose visiting each page every T time units, though an accurate model for a real index would need to account for the growth of the collection over time by allowing T to vary. It would also be necessary to include the natural statistical variation in retrieval period, but doing so depends upon a number of implementation details.

For this preliminary analysis, we assume a constant web size to avoid this difficulty. In this case, the integration of (2.33) with respect to T is just integration over a delta function at the chosen period and simply filters out the desired period T_0. Using the Weibull distribution for inverse change rates, the expected probability α that a uniformly randomly selected page will be β-current in the search engine index is

$$\alpha = \int_0^\infty \left[\frac{\sigma}{\delta} \left(\frac{t}{\delta} \right)^{\sigma - 1} e^{-(t/\delta)^\sigma} \right] \left[\frac{\beta}{T_0} + \frac{1 - e^{-(1/t)(T_0 - \beta)}}{(1/t)T_0} \right] dt. \tag{3.17}$$

The integral (3.17) can only be evaluated in closed form when the Weibull shape parameter σ is 1; otherwise, numerical evaluation is required. The integral gives an α for every pair (T_0, β), defining a search engine "performance surface." This surface can be interpreted in a number of ways. For example, we can choose a probability α and determine all pairs (T_0, β) that give that probability. Using our parameter choices from the lifetime-based optimization of (3.15), we have evaluated the integral and plotted it in Figures 3.23 and 3.24, which show the level set for $\alpha = 95\%$. We note again that the revisitation times that result from this analysis are upper bounds since our analysis is based on less volatile pages (those that provide timestamps).

From that plot, we can see that in order to maintain (0.95, 1-day)-current search engine, a re-indexing period of 8.5 days is necessary. For (0.95, 1-week)-currency, a re-indexing period of 18 days is necessary. Notice that these figures do not depend upon the number of documents in an index, so a re-indexing period defines a set of pairs (α, β), regardless of changes in the size of the index. Alternatively, we can estimate effective bandwidth requirements to maintain a given level of currency for a uniform index of a given size. By "uniform" we mean that no documents are given any sort of preference; all are re-indexed at the same rate. The effective bandwidth is not to be confused with the link bandwidth, it simply describes the overall processing rate, including download and analysis. For a single incoming link, though, it is the necessary link bandwidth. It is possible that the download and analysis might be done from multiple distributed machines, thus all machines need not be supplied by a single link having that bandwidth.

For example, an (0.95, 1-day) index of the entire web, using the estimate of 800 million pages from [LG99], would require a total effective bandwidth of (approximately)

$$\frac{800 \times 10^6 \text{ pages}}{8.5 \text{ days}} \times \frac{12 \text{ kilobytes}}{1 \text{ page}} = \frac{104 \text{ Mbits}}{\text{sec}} \text{ for (0.95, 1-day) current.} \tag{3.18}$$

A more modest index, closer to those actually in use, might have 150 million documents at (0.95, 1-week) currency, requiring an effective bandwidth of around

$$\frac{150 \times 10^6 \text{ pages}}{18 \text{ days}} \times \frac{12 \text{ kilobytes}}{1 \text{ page}} = \frac{9.4 \text{ Mbits}}{\text{sec}} \text{ for (0.95, 1-week)-current; 20\% cover.} \tag{3.19}$$

Clearly, other re-indexing schemes exist where T is not constant but is a function of λ; see [CLW97] for some good discussion on possible schemes. In comparison to our work, [CLW97] assumes $\beta = 0$ and scales the cost function in a particular way. When T is a function of λ, the integral (3.17) is modified by substituting in the function $T(1/\bar{t})$ and evaluating along the appropriate line in the $(T, \bar{t})$-plane. Additional modifications to this development might include the addition of a noise term to the observation period and choosing the grace period β as a function of the change rate λ. We consider how it is possible to choose T to maximize α in Chapter 4.

Figures are scant regarding the actual number of pages indexed per day by commercial search engines, but vary between 3.5 million per day [Nic97] and 10 million per day [CvdBD99]. For (3.18), we would require 94 million per day, while (3.19) corresponds to 8.3 million per day and is probably achievable. Note, however, that (3.19) assumes a uniform sample of 20% of the web as represented by our data. It is likely that the collections of major search engines may not fit this description, but rather contain a subsample of a more interesting fraction of the web (namely one containing more useful and dynamic sites). Moreover, many of the pages downloaded and indexed are being seen for the first time. The figures on number of downloads per day include the process of exploration in addition to re-indexing; therefore only some fraction of the effort goes into maintaining a current image.

3.4.2 Empirical determination of (α, β)-currency

Web searchers have an intuitive feel for how up-to-date they expect an engine to be. For current events, a search engine is almost out of the question, while for more esoteric (and less volatile) information, search engines are ideal. Where exactly a search engine could be expected to lie within this spectrum is not as well established. Using the theory we have developed, this section details our estimation of the (α, β) currency of four commercial search engines with respect to a one-time

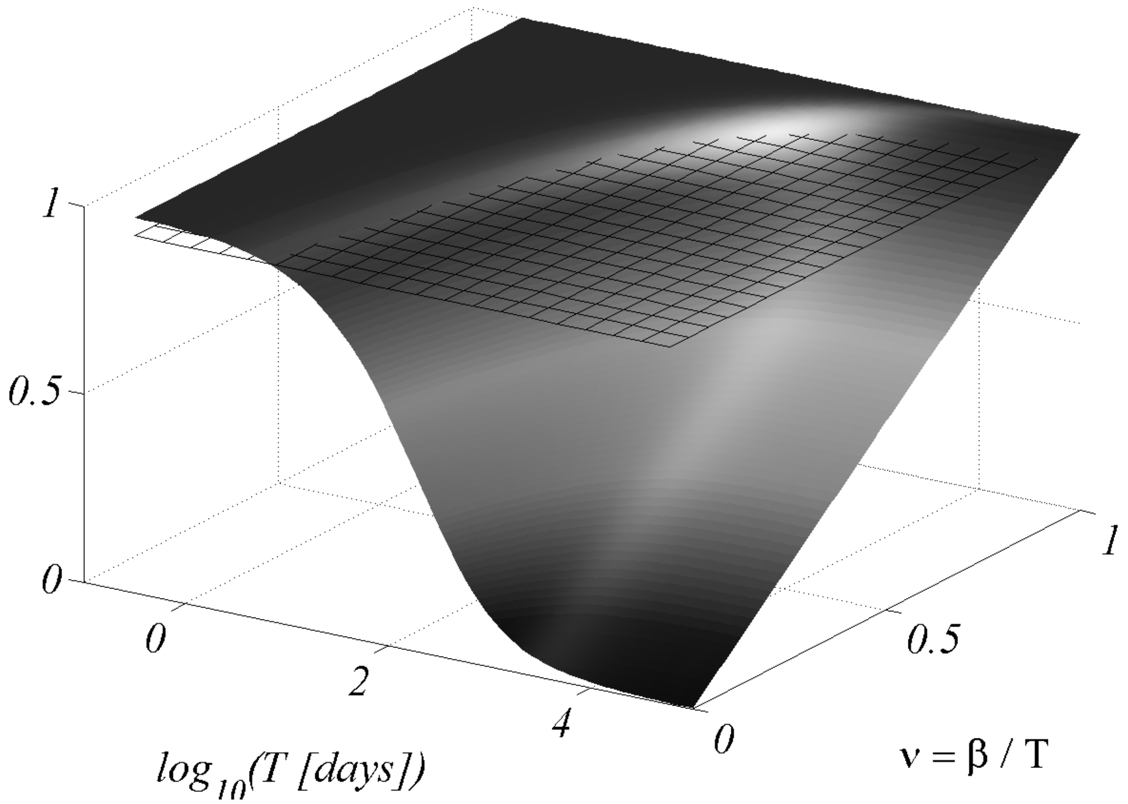


Figure 3.23: **Probability α as a function of ν and T_0.** Here, we plot the probability surface α as a function of the grace period fraction $\nu = \beta/T_0$ and fixed re-indexing period T_0. This surface results from using the (more accurate) lifetime-based population parameters, although this surface could be constructed for any population. The plane at $\alpha = 0.95$ intersects the surface in a level set, which is plotted in Figure 3.24 (with β values used instead of percentages ν).

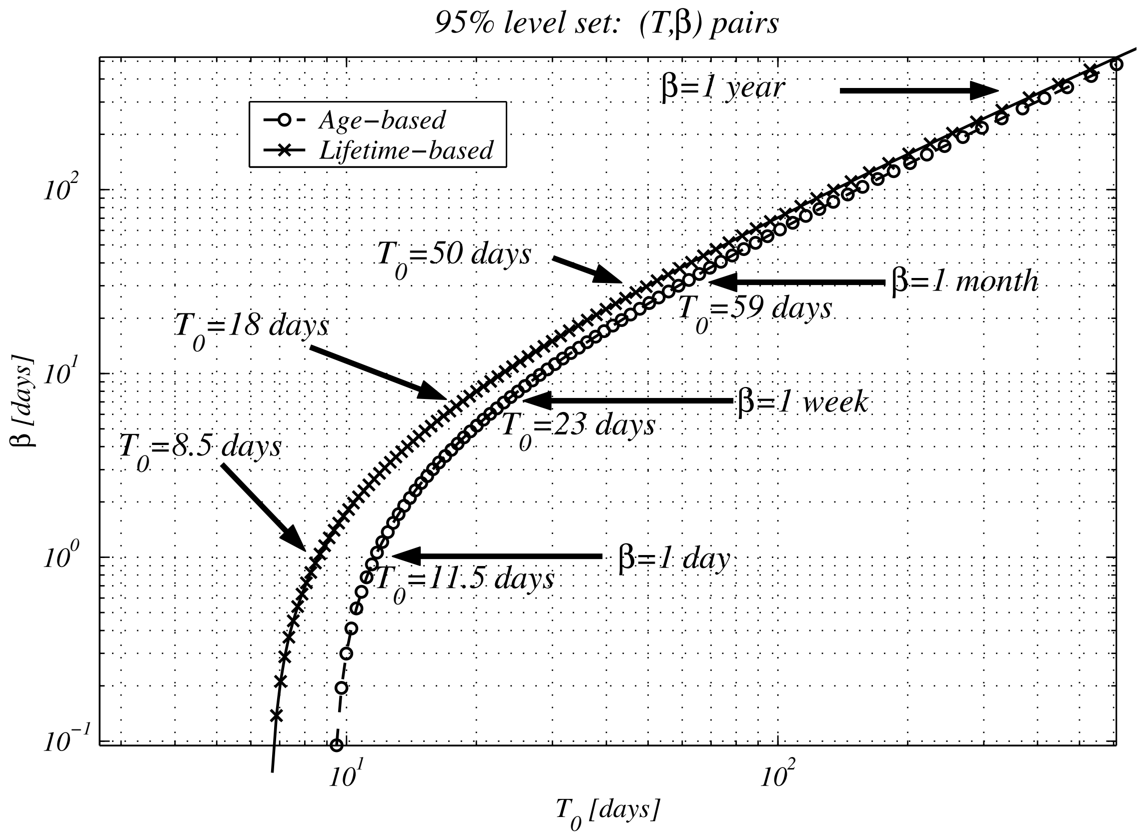


Figure 3.24: **Relating β and T_0: α=95% level set.** Here we have plotted two level sets of pairs (T_0, β) which yield a probability $\alpha = 0.95$ of being β-current. The two curves are derived from two different estimation methods, minimizing (3.10) or (3.15). The lifetime-based estimates are much more accurate. Regardless of the size of the collection, this data can be used to estimate how current an engine is when the indexing period T_0 takes on a value (in days) along the horizontal axis. As T_0 becomes large, the relative check rate is too slow, and β approaches $0.95 \times T_0$—namely, the only path to 95% correctness is to ignore the all-too-frequent changes 95% of the time.

experiment.

Any search engine that asserts a timestamp along with each search result provides a verifiable means of measuring how (α, β)-current its index is with respect to an imposed demand distribution. Specifically, timestamps claimed by the engine can be compared to those returned by the pages themselves, telling whether or not the index entry is truly up-to-date. If an engine does not return a timestamp, it is not possible to measure its (α, β)-currency in this fashion. In order for the currency to be measurable, there must be a way to ascertain (externally) whether the engine has a current image of the document in question or not.

Fortunately, some engines do provide this information. AltaVista[11], Northern Light[12], and Infoseek[13] assert timestamps along with each search result. The Hotbot[14] engine returned timestamps with results as well, but a bug in their software at experiment time caused erroneous timestamps for many documents. Specifically, results for which timestamps were not available were listed with the time "Dec 31 1899", or even more confusing, "12/31/99" This unreliability may indicate that other timestamps from this source would also be unreliable, so this engine was not tested. The Google[15] search engine provides links to cached copies of many result documents from which a modification time can be inferred. It is likely (although uncertain) that any document observed by their crawler has a corresponding cache entry; result references without cached copies are probably derived from links found (but not followed) on indexed pages [HN99]. This is part of how Google works–pages are ranked according to incoming links and need not actually be downloaded in order to be considered possibly relevant. At the time of our experiment (early February, 2000), each cached document was returned by the Google engine along with the header given by that page's server when it was originally returned to the Google crawler. Presumably the cached copy is the most recent observation made.

To test the engines, we performed the following steps:

1. Run a randomly-selected query from a list of query terms taken from the Informant.

[11] http://www.altavista.com/
[12] http://www.northernlight.com/
[13] http://infoseek.go.com/
[14] http://hotbot.lycos.com/
[15] http://www.google.com/

2. Within the results list, parse the page to extract the URLs and the corresponding modification times asserted by the engine, which we will call t_{SE} (for "S"earch "E"ngine).

3. Fetch each URL in the list, and note the actual `Last-Modified` time for each one (if provided by their respective servers). These times will be referred to as t_{WP} (for "W"eb "P"age).

4. For each URL, record the two timestamps, along with the time of the query, t_{obs}.

5. When both timestamps are available for a document, determine the difference between them.

6. Test for (α, β)-currency over all URLs having both timestamps by noting when the page was last altered, and whether or not this happened during a grace period. Repeat this step for many values of β.

If the timestamps (asserted and observed) differ by more than can be explained by roundoff error or clock skew (we used ±one day), then the index entry is considered out of date. Formally, given our assumption, such an index entry is not 1-day current. Whether or not this error is to be forgiven is decided according to whether the observed timestamp on the page was within β time units of the query time. This can be done for all possible values of β over some reasonable range for all the test queries in the set. The percentage of pages for which the most recent change lies within the grace period are counted as up-to-date, while the remainder are not. The fraction of all such pages that are up-to-date is an estimate of the probability α that the index is β-current.

Errors in this estimate of α can arise from inaccurate timestamps asserted by the search engine. On more than one occasion, engines were even observed to have "post-dated" assertions. It is possible that nefarious content providers set their clocks far in the future to make their information appear more current. When a user would sort by date, these documents would magically bubble to the top. Of course, the true explanation could be less insidious. For example, an engine might try to guess the date if it could not be determined, or use the observation time as an estimate of the modification time. Incorrectly set server clocks could also cause inaccurate timestamps if the server's time is not validated or adjusted in some way.

Estimation errors can also arise from forms of aliasing. When the most recent change in a long

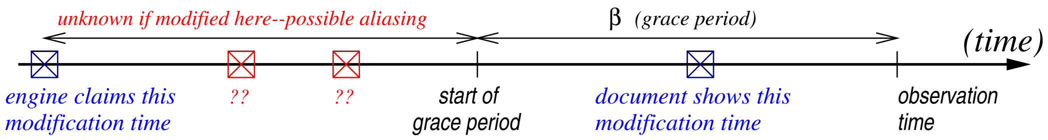


Figure 3.25: **Aliasing in test for engine β-currency.** There is a possibility that unobserved changes may lie outside the grace period. When determining an engine's probability of being β-current, the probability of this event must be taken into account.

sequence falls within the grace period, any changes which occurred prior to that change are never seen. Any of these changes that lie outside the grace period might therefore be incorrectly forgiven, if we fail to account for the probability of their occurrence. Without this correction, fast-changing pages will have a tendency to incorrectly appear β-current since they have a higher probability of unobserved changes lying outside the grace period. Moreover, engines which give greater rank preference to such faster-changing content will appear more β-current.

The situation is shown in Figure 3.25. This timeline shows how aliasing can arise in our test for the probability α of a search engine being β current. First, we note that the times claimed by the search engine and that returned by the document are different. Next, note that the modification time lies within the grace period, β, preceding the observation time. Now if this is the first change since the search engine last observed the document, then it should be forgiven in accordance with our definition. But if changes did go unobserved outside the grace period, this should not count as β-current. Therefore we must estimate the probability of a change between the claimed time and the start of the grace period, and weight the observations accordingly.

To determine the probability of aliasing, we use (3.11) again, except now we have a fixed time interval and an estimate for the change rate λ, instead of the other way around. We repeat the equation here, with some new terminology:

$$\Pr(\text{aliasing}) = 1 - e^{-\lambda(t_{obs} - \beta - t_{SE})} \tag{3.20}$$

Again, we have used t_{obs} for the time of observation, t_{SE} for the modification time claimed by the search engine. Thus (3.20) is the probability of a change during the time period between t_{SE} and

and the start of the grace period in question, at $t_{obs} - \beta$.

The complement of this probability then weights how much an observed document age inside the grace period counts towards the β-currency of an engine for a particular value of β. For example, if we find that there is a probability 98% that a modification occurred outside the grace period, but after the search engine's claimed time, then we should only forgive 2% of such instances as actually being β-current.

Evaluating (3.20) requires that we have a value for λ. Using the age-based estimation method, the change rate is estimated by the inverse of the document's age. At the time of observation we obtain a single age sample. Obviously the variance when using this single sample as an estimator is quite large (the square of the mean), but since this experiment is making estimates of aggregate search engine behavior, this problem is not serious. The effect of using such a poor estimator is mitigated by the large number of observations of similar documents. Provided the creation time of the web page is far enough in the past, (3.3) tells us that the age distribution is very close to the lifetime distribution. Since these estimates are only applied to documents which have gone through at least one change, we have some assurance that the document is not "brand new".

3.4.3 Experimental results for four search engines

The test data show the accuracy of the perception that search engines do return more than the occasional inaccurate or incorrect link. Bear in mind that this experiment has used an extremely unforgiving model of what is meant by a "change", including far more than large changes like the complete absence of a document. Even minor changes that have little impact on a user's perception of index performance were counted. From a user's perspective, much larger changes could be tolerable if the reference returned was still an accurate response to their query.

At the outset, it should be emphasized that our results represent the state of the engines at experiment time and may not reflect their general character. Of the four engines tested, the most current for small β was AltaVista, and the least current was Northern Light. However, the data further show that even though the Google engine had relatively poor performance for low values of β (< 10 days), it came out ahead when β was larger than about one month. Continuing in this

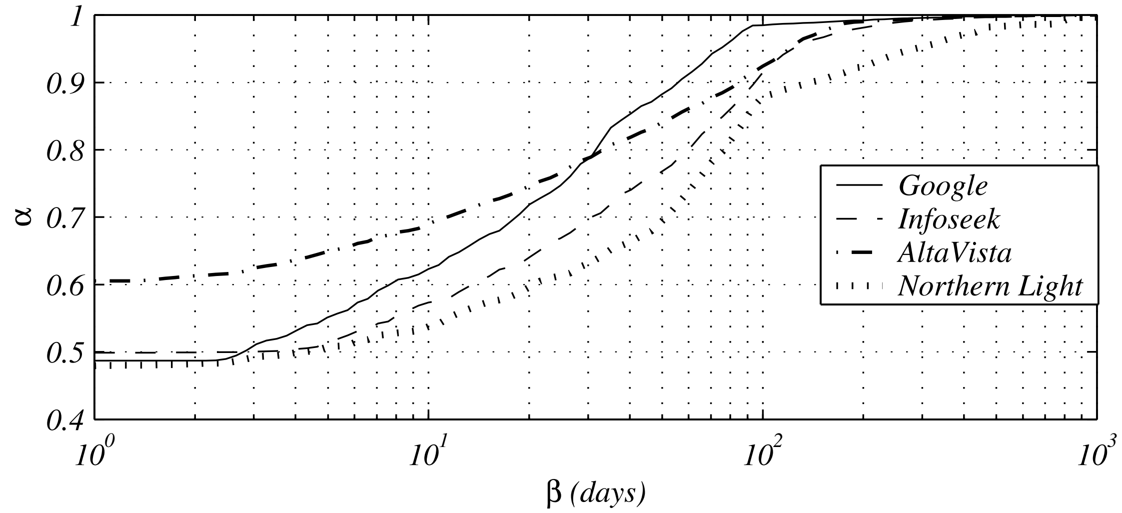


Figure 3.26: **Empirical probability of being β-current.** This graph shows an experimental determination of the probability of an index being β-current. The horizontal axis shows the grace period time, and the vertical axis shows the probability of being β-current for that grace period. This figure was generated using the aliasing correction described by (3.20).

trend, Google reaches the $\alpha = 0.95$ mark at around $\beta = 73$ days, while AltaVista reaches that plateau at $\beta = 104$ days. The odd shape of their performance curve may reflect a design decision on Google's part of how current their users expected the information to be. Alternatively, there might be a significant lag between the time an observation is made by their crawler, and when it is posted as having been cached (recall that timestamps for Google were obtained from their cached copy of each web page).

It appears likely that Google does short-duration, high-volume index refresh operations, such that observation times tend to be clustered rather strangely. This is as opposed to AltaVista, which has claimed in the past [Nic97] to have a more continuous (nightly) index building procedure. It is also possible that AltaVista had only just recently done an index build and therefore appeared much more current than Google. It could be that Google believes their users are expecting things to be current to within a few months (rather than a few days) and that they have therefore allocated resources accordingly, affording themselves a greater breadth of coverage. The tradeoff between having a large index versus a current one is discussed in [LG98]. Furthermore, given the discussion about ranking and crawling algorithms presented in [BP98] and also in [JCP98], it is quite likely that Google preferentially crawls and prioritizes documents expected to have a high ranking within their results.

From the region where each engine reached the 95% mark, we can take a rough guess as to the re-indexing period T_0 used with these documents. By comparing the value of β for each engine with those shown in the $\alpha = 95\%$ curve in Figure 3.24, we can estimate that T_0 is between around 75 and 120 days for these collections. We return to this point in Chapter 4 when presenting continuous-time optimization methods as a basis for comparison of our work with engines currently in use.

Another feature of the graph is that there is little change in probability of β-currency for small values of β. This is largely due to the correction applied to remove the effects of aliasing. The kinds of web pages that might be incorrectly forgiven for small β (as shown in Figure 3.25) are precisely those which change the most, so applying the correction for aliasing tends to flatten the curve in that region.

This kind of evaluation is by no means limited to search engines–similar data could be determined for any kind of index, whether it be a list of stock quotes (free, web-based indices often lag by 20 minutes or more), a local weather report, military intelligence, or any index of dynamic information sources. The only requirements are that one be able to obtain an estimated timestamp as well as an actual one.

Chapter 4

Optimizing observation of dynamic sources

From the search engine tests presented at the conclusion of Chapter 3, it is clear that different search engines have different technologies and strategies for keeping their indices up to date. It is clear that the different search engines have made these design choices, either explicitly or implicitly. Indeed, any index of changing information sources must include a strategy for re-indexing objects in the collection. This chapter is concerned with the different ways to make decisions about choosing a sequence of objects to be observed so as to best maintain an index. We apply the theory from earlier chapters to develop algorithms for optimizing observation strategies, and evaluate these algorithms for their effectiveness.

To state the problem more precisely, in what order should objects in a collection be observed so as to maximize the probability α that a randomly-selected (according to a known distribution) index entry is β-current? We answer this question under two very different sets of assumptions, corresponding to the models presented in Chapter 2. First, we assume a discrete time change process governed by Markovian transition probabilities between ages. In this case, we use $\beta = 0$ and consider only finite-horizon versions of the problem. Solutions take the form of choosing the N_o objects to be observed on the next time step, so as to achieve the greatest expected finite-horizon cost reduction. In this framework, we maximize α by maximizing the expected number of objects up-to-date (with $\beta = 0$) within the index, summed over a finite number of time steps. The expectation in this case is uniform over the objects. By taking the expected number of objects out-of-date as the cost, and maximizing the cost reduction summed over many time steps, α is maximized. These concepts were presented in Chapter 2. We show how this optimization can be expressed as a parametric network flow problem. While useful as an analytical tool, the discrete time method does not scale well, and may not be practical for large monitoring systems. The second half of the chapter adds some assumptions to make the optimization more practical—continuous time Poisson processes are used instead of the more general Markov models. Thus each object is characterized by a single number, namely the change rate. This reduces the problem from selection the next set of objects to be observed to assignment of a desired observation rate for each object. For an object having a known change rate, this means that only one decision is necessary, as opposed to the continual reassessment required within the discrete-time framework. The continuous algorithm seeks to partition the objects under observation according to their popularity and rate of change, and then observe uniformly within each bin of the partition. The rate assignment is constrained such that the sum of the rates cannot exceed some fixed value. We demonstrate a simulated annealing approach to this process of division and rate assignment. In this approach, a solution consists of a partition of the monitored object space, by change rate and importance of each index entry, and an allocation of rate into the corresponding bins. The fitness of such solutions are calculated by directly evaluating numerical integrals for α within the partitions and then combining these via a weighted sum. We conclude the chapter with a discussion of the practical issues surrounding the implementation of our algorithms.

4.1 Optimizing observation for discrete-time systems

Recall for a moment the goals that were spelled out in Chapter 2: on page 23, we defined two ways to measure the performance of an indexing system using the discrete-time Markov chain model. These were the expected number of objects out-of-date and the expected number of time units out of date. This section concerns the minimization of these functions.

4.1.1 Greedy cost minimization

The simplest possible strategy for both discrete and continuous time cases is to immediately re-index the objects having the highest probability of going stale in the near future. For the discrete time domain, this translates into sorting objects in the index by their contribution to the total cost, and scheduling those objects for observation for which the greatest cost reduction is taken immediately. Specifically, applying (2.20)

$$C_{(1)} = \sum_{r=1}^{d} \left(1 - \left[\mathbf{M}_r^{k_r}\right]_{i_r(i_r+k_r)}\right), \tag{4.1}$$

or (2.22),

$$C_{(2)} = \sum_{r=1}^{d} Z\left(i_r, k_r\right), \tag{4.2}$$

to calculate costs, one can find the best way to reduce the costs that will be incurred in the coming single time unit, and thereby maximize the expected α over a finite time horizon. Please refer back to Chapter 2 for the full definition of the expected number of time units out of date, $Z\left(i_r, k_r\right)$, and the age transition probability matrix, $\mathbf{M}$.

Proceeding with the optimization, assume that the system can check N_o objects per time interval. The smallest possible cost for the coming time unit can be obtained if we fetch and re-index the N_o objects corresponding to the largest terms in the cost summation. These terms correspond to those objects with the largest probability of being out of date (2.20), or those that we expect to have been out-of-date the longest (2.22). For all of the N_o objects we fetch, the probability of being incorrectly indexed, at the end of that time interval, is zero. If there is no single best choice for the N_o objects, as is possible when applying (2.20), then we can select N_o at random from the pool of best choices. This situation would occur, for example, when applying (2.20) if more than N_o objects have unit probability.

These are the greedy versions of cost minimization, inasmuch as the greatest immediate cost reduction is chosen. Refinements of these methods operate along the same lines, observing those N_o objects which afford the greatest reduction in cost. More complex methods can find the expected cost for more than a single time unit, and the actions corresponding to the lowest long-term expected cost can be used. These methods, which will be discussed in a later section, give more assurance of long-term optimal performance.

4.1.2 "Liveness" conditions

While having the advantage of being relatively simple, greedy algorithms often force suboptimal long-term performance. For example, in the re-indexing system, there is the possibility of never checking some subset of the objects. It may be desirable to avoid this, especially if all items to be indexed are equally important. In queueing models for computer operating systems, the analogous constraint that all processes be served is termed a "liveness" condition, which would not be met if there were a subset of objects that changed so quickly that its members always contributed the largest terms in the cost function. Fortunately, the two discrete-time algorithms presented both can be shown to eventually check all objects, under the key assumption that we select the N_o objects without any preference for particular pages (when there are more than N_o valid candidates).

We first consider liveness with respect to the expected objects out-of-date described by (2.20). Since each term in the summation is a probability, no term can be larger than 1. Moreover, once a term becomes unity, it remains so until the object to which it corresponds is checked. If all terms eventually become 1, then all objects will be checked. Examining the form of each probability term (as given in (2.19)), it is clear that this probability becomes unity for object r for any time that forces the sum to include an entire row of the matrix $\mathbf{M}_r$. As mentioned above, if there are more than N_o objects corresponding to terms having probability 1, then some subsidiary selection process must be used. If no preference is given to any object, then even a random selection will guarantee that all objects having the maximum probability (in this case, 1) will be checked eventually. Therefore, the liveness condition is satisfied by the one-step greedy minimization of the expected number of non-current objects.

Liveness follows in similar fashion for the expected number of time units out-of-date in (2.31). If the objects with the N_o largest values of $Z(i_r, k_r)$ are selected for observation, as in the greedy algorithm, then any object will eventually be included in the observed set. This stems from the fact that $Z(i_r, k_r)$ will increase without bound if object r is unchecked, thereby guaranteeing that object r will have a large enough cost to guarantee its inclusion in the observed set (if we wait long enough).

To show this, we define the smallest member of the inspected set of N_o inspected objects at time step t to have expected time out-of-date $C_{min}(t)$. For the collection, there will be some time C_{max} that is greater than or equal to $C_{min}(t)$ for all t. That is, there is a largest member of the set of smallest costs. In order for an object to be included in the inspected set of N_o objects, it is sufficient to guarantee that its expected total time out-of-date be greater than C_{max}. From (2.31), as k_r grows without bound, we are eventually just adding unity to the cost with each passing time unit. Therefore, the expected time units out-of-date $Z\left(i_r, k_r\right)$ will eventually become greater than any finite value C_{max}. By analogy, imagine we have a lawn in which each blade of grass will grow forever (becoming more out-of-date) if we never cut (observe) it. Although grass in the lawn grows at different rates, no matter how high we set the wheels of the mower, we can always be assured that any blade will eventually grow high enough to be cut.

4.1.3 Extending the horizon: two-step cost functions

Having an assurance of liveness is not enough to be satisfied with the long-term performance of the system. The one-step algorithm does not take into account anything other than the current probability of change for various objects. Indeed, no one-step method will make proper use of the difference in change rates among objects. By using the behavior expected to be seen beyond the first time step, we can improve expected performance in future phases of observation. This can be seen by considering cost functions evaluated over two (or more) time units.

A simple two-step example

Consider the following simple system that demonstrates how to take advantage of object change rates. There are two objects, A and B, only one of which can be observed per time step. Object A changes quickly: it has a probability of 85% of having been changed in the first time interval, and if unobserved, it will have a 95% chance during the next interval. If, however, we observe it in the

Sequence	Cost	Comment
AB	$0.8 + 0.25 = 1.05$	lower first time unit cost; one-time unit algorithm would pick this one
BA	$0.85 + 0.05 = 0.90$	lower total cost; two-time unit algorithm would pick this one
AA	$0.8 + 0.81 = 1.61$	object B ignored
BB	$0.85 + 0.95 = 1.80$	object A ignored

Table 4.1: **Possible costs in example two-object, one-check system.** For this simple system, we have listed each strategy and its cost. The lesson is clear; it is quite possible to have lower long term cost by extending the planning horizon.

first time interval, there will be a 25% chance of it having been changed by the end of the second time unit. Object B changes more slowly. It has an 80% chance of being altered on the first time step, which increases to 81% on the second step if B is unchecked on the first step. If it is checked on the first time step, then the probability of it changing on the next step will be 5%. Assume we can only choose one of these objects to observe per time step, and that we wish to minimize the total expected number of objects out-of-date over a period of two time units:

$$\sum_{t=0}^{1} \kappa^t \left[\text{Prob}(A \text{ changed at } t) + \text{Prob}(B \text{ changed at } t)\right] \tag{4.3}$$

Here, $\kappa \in [0, 1]$ is a "discount factor" with which we account for the relative value of cost avoided in this time unit versus that on subsequent units. This is a reflection of the possibility that immediate benefit may be more valuable than deferred benefit. For this illustration, we assume $\kappa = 1$.

There are four possible strategies for the two time units. These can be written as sequences of observations, namely AA, AB, BA, and BB. If we observe an object, then it contributes zero cost on that time unit, since we consider it "up-to-date" if indexed within the last time unit. Therefore, the cost for observing A on both time units is exactly the cost of not observing B on those time units, namely, $0.8 + 0.81 = 1.61$. Likewise, we can find the two time unit cost for each possible sequence of observations, as shown in Table 4.1. As can be seen, the greedy algorithm would select a suboptimal strategy.

General two-step costs In the general case, we have d objects and N_o observations. Each of the four possible strategies chosen for object r will have a single cost associated with it: (i) the cost of not observing on the first time unit but then observing on the second; (ii) the cost of not observing on either time unit; (iii) the cost of observing on the first time unit but not on the second; and (iv) the cost of observing on both time units. The current cost calculations have the appealing feature that choosing to observe one object does not affect the cost that might be contributed by other objects. That is, the cost of not observing an object on any given time unit depends only on its age, dynamics, and how recently it was observed. For longer horizon planning, note that the number of possible possible strategies for a single object is 2^n, for n time steps.

For the two-step problem, consider these cost possibilities in order. First, the cost of not observing on the first time step is the same as was given in (2.19):

$$C_{XO_r} = \sum_{j\in[0,i_r+k_r-1]} \left[\mathbf{M}_r^{k_r}\right]_{i_r j} = 1 - \left[\mathbf{M}_r^{k_r}\right]_{i_r(i_r+k_r)} \leq 1 \tag{4.4}$$

Here, we introduce some new notation; we write two-step strategies for object r as sequences of X's (do not observe) and O's (observe). The first letter is the action on the first time unit, and the second letter is the action on the second time unit. Thus the two-time unit cost of not observing object r on the first time unit and then observing it on the second time unit is written C_{XO_r}.

Alternatively, if we choose not to observe on the second time unit, we will add additional cost to C_{XO_r} to obtain C_{XX_r}. The probability that the object has gone out-of-date by the second time unit is exactly like (4.4), except that the object has aged by one time unit. This is accounted for by incrementing the value of k_r; if we choose not to observe on the second step, the cost for that time unit only is:

$$\sum_{j\in[0,i_r+k_r]} \left[\mathbf{M}_r^{k_r+1}\right]_{i_r j} = 1 - \left[\mathbf{M}_r^{k_r+1}\right]_{i_r(i_r+k_r+1)} \leq 1 \tag{4.5}$$

We note that this second step cost is greater than or equal to the first step's cost: by waiting an additional time unit, we can only add to the probability of the object having gone out of date. Specifically, we add the probability that the object went out of date on the additional time unit, which we calculated in (2.26). Adding (4.5) to the first-time unit cost (4.4), we obtain C_{XX_r}: the two-time unit cost of not observing on either time unit (denoted C_{XX_r}):

$$\begin{aligned} C_{XX_r} &= 2 - \left[\mathbf{M}_r^{k_r+1}\right]_{i_r(i_r+k_r+1)} - \left[\mathbf{M}_r^{k_r}\right]_{i_r(i_r+k_r)} \\ &= 2C_{XO_r} + P(k_r + i_r + 1, i_r) \leq 2 \end{aligned} \tag{4.6}$$

This cost is the entire cost for both the AA and BB options in Table 4.1, since both required that one of the objects not be observed for both time units. In general, this will be the largest possible cost for a single object over the two-time unit period, and it cannot be greater than 2. Moreover, from the second line of (4.6) we know that

$$C_{XX_r} \geq 2C_{XO_r} \geq 0 \tag{4.7}$$

Next, we consider the cost C_{OX_r}, incurred if we observe object r on the first time unit but not on the second time unit. Zero cost is contributed on the first time unit, and the second time unit contribution depends on the new value of $k_r^+ = 1$ (time units since the most recent observation), and the newly-observed age i_r^+. Since only a distribution of possible values of i_r^+ is known, the second-time unit cost will be a weighted sum of probabilities from the matrix $\mathbf{M}_r$ of one-time unit state transition probabilities (2.1). If the object was observed to be in state j_r on the first time unit, then the probability of it being out-of-date at the end of the second time unit is just $p_{reset}(j_r)$, as defined in (2.2). These probabilities are also the first column of the matrix $\mathbf{M}_r$. In our cost function, the values in this column vector will contribute in proportion to their probability of occurrence. These probabilities are obtained from the row over which we sum in (4.4), or the distribution of possible ages on the previous time unit. Therefore, the probability that the object is out-of-date at the end of the second time unit, given that the object was observed on the first time unit, is

$$C_{OX_r} = \sum_{j_r=0}^{N} \left[\mathbf{M}_r^{k_r}\right]_{i_r j_r} \mathbf{M}_{j_r 1} \leq 1 \tag{4.8}$$

Note that this cost, like C_{XO_r}, can be no greater than 1. If we expect that object r has changed with some large probability, then we would rather observe it on the first time unit rather than the second: $C_{XO_r} > C_{OX_r}$. Alternatively, if we expect that no change has occurred yet, then we would rather wait to make an observation. For example, if an object will change on the second unit with probability 1, but definitely will not have changed by the end of the first time unit, then it is beneficial to wait until the next time unit to observe it: $0 = C_{XO_r} < C_{OX_r} = 1$. Most cases are less extreme than this, but we emphasize that the sign of the difference $C_{XO_r} - C_{OX_r}$ can be positive or negative.

Finally, for completeness, two observations of object r will force it to contribute zero cost:

$$C_{OO_r} = 0 \tag{4.9}$$

This may be the best option when object r is changing very rapidly, or when $C_{XO_r} \approx 1$ and $C_{XX_r} \approx 2$. Clearly, this implies that the object changes essentially every time unit, making it an almost guaranteed success for cost reduction. However, too many such objects in a collection might cause much of the collection to be ignored on a regular basis, and performance will suffer.

Now that we can determine a cost for all possible strategies, we are able to add these costs for an entire collection. Any observation plan for n time units will partition the d objects into 2^n groups corresponding to the possible n-time unit strategies. In the case of $n = 2$, the groups are:

$$\begin{aligned}
\mathcal{OO} &= \left[OO_1, OO_2, \ldots OO_{|\mathcal{OO}|}\right], \\
\mathcal{XO} &= \left[XO_1, XO_2, \ldots XO_{|\mathcal{XO}|}\right], \\
\mathcal{OX} &= \left[OX_1, OX_2, \ldots OX_{|\mathcal{OX}|}\right], \text{ and} \\
\mathcal{XX} &= \left[XX_1, XX_2, \ldots XX_{|\mathcal{XX}|}\right].
\end{aligned} \tag{4.10}$$

The sizes of the groups are constrained so that no more than N_o observations can be made in a single time unit, and all objects are accounted for:

$$\begin{aligned}
|\mathcal{OO}| + |\mathcal{OX}| + |\mathcal{XO}| + |\mathcal{XX}| &= d \\
|\mathcal{OO}| + |\mathcal{OX}| &= N_o \\
|\mathcal{OO}| + |\mathcal{XO}| &= N_o
\end{aligned} \tag{4.11}$$

The first relation is a constraint that each object is assigned exactly one strategy, and the next two are constraints on the number of observations per time unit. Notice that these imply $|\mathcal{OX}| = |\mathcal{XO}|$.

Since there are three equations and four unknowns, the system reduces to a single parameter choice. Once any one of the sizes is fixed, then the others are fixed as well—we could, for example, freely choose the number of objects that would be observed on both time units at some $0 \leq k \leq N_o$.

The "observe-n-times" group (e.g., group $\mathcal{OO}$) will always contribute zero cost; other groups may have nonzero cost. The cost for the collection is the sum of the cost over all members of all nonzero cost groups:

$$C_{(1)} = \sum_{k \in XX} C_{XX_k} + \sum_{l \in OX} C_{OX_l} + \sum_{m \in XO} C_{XO_m} \tag{4.12}$$

In a collection of d objects, in which we can check N_o per time unit, there are ${}_dC^2_{N_o}$ possible combinations of two-time unit strategies. A brute-force approach, in which we evaluate a cost for each option, is entirely infeasible for the collection sizes under consideration.

Fortunately, the two-time unit problem can be stated as a simple minimum-cost network flow problem, as diagrammed in Figure 4.1. In this context, we think of objects "flowing" to particular two-time unit strategies, building the sets (4.10). Each graph of this type has d unit sources corresponding to the objects in the collection, and variable-size sinks corresponding to each allowable strategy. The sink sizes enforce the constraints on the number of observations and the number of strategies to be assigned. All sources are connected to all sinks by unit capacity links. Costs on these links correspond to those presented above as C_{XO_r} (4.4), C_{XX_r} (4.6), C_{OX_r} (4.8), and C_{OO_r} (4.9). Only a few costs are (generically) labeled in Figure 4.1 to avoid clutter.

Turning attention again to the sinks, we recall from (4.11) that the set sizes for the two-step problem can be fixed by making one free parameter choice, say $|\mathcal{OO}| = k, 0 \leq k \leq N_o$. This determines the sizes of the other strategy groups, which is also the size of the corresponding sinks. For each allowable value of k, there is a single lowest cost "strategy flow". As an example, in Figure 4.2 we show the networks corresponding to the two subproblems which correspond to paired strategies shown in Table 4.1. Since no object can be assigned to more than one strategy, this is an integer flow problem. Since the sources and sinks are all of integer size, integer flows can be found using standard algorithms without additional constraints [Ber98]. It is also possible to allow real-valued flows, where a mixed strategy is selected first, and then actual strategies are selected

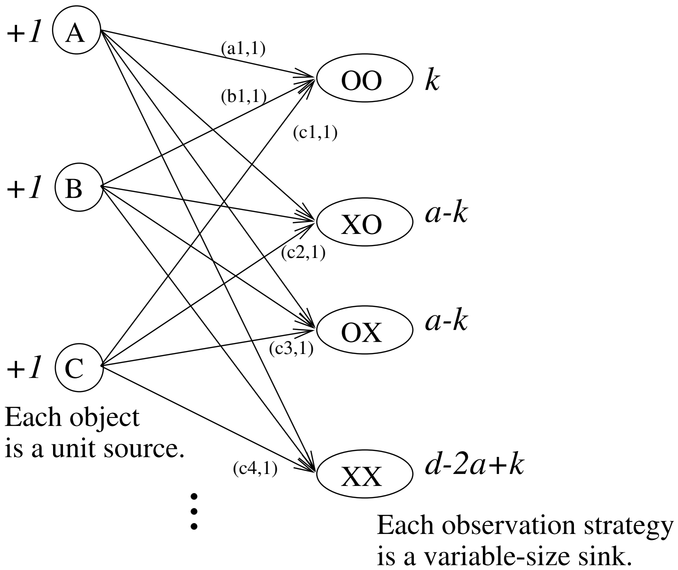


Figure 4.1: **Two time unit strategy assignment** $(capacity, cost)$ **network.** This figure shows the network representation of the two step problem, having k observations in class $\mathcal{OO}$. Unit size strategy sources on the left feed into variable size strategy sinks on the right hand side, and each link has an associated cost.

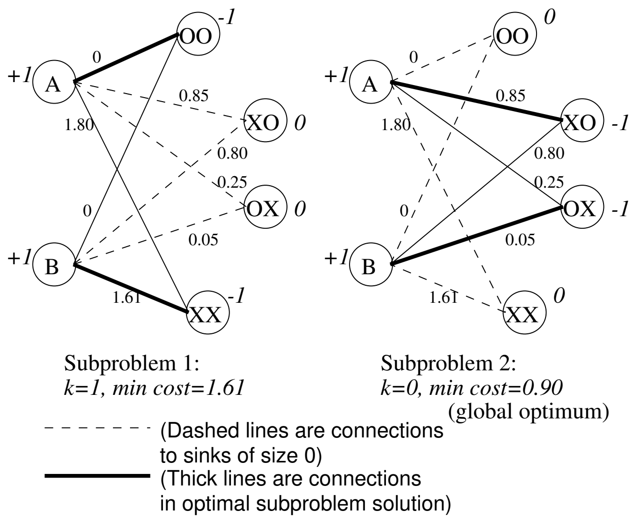


Figure 4.2: **Two-step strategy assignment network corresponding to Table 4.1.** This figure shows the possible networks for each subproblem, as described for Figure 4.1. There are only two possible sizes $|\mathcal{OO}|$, each corresponding to one network representation.

probabilistically according to the proportion of flow from an object to each strategy. Once the sink sizes are set, the solution of the subproblem proceeds according to the standard methodology (see [Ber98] for more) for finding minimum-cost flows.

The two-step problem is thus (at worst) a series of N_o minimum-cost flow problems. Addition of more time steps to the problem adds one constraint per time step, corresponding to limiting the observations per step to N_o, but the number of possible strategies grows exponentially. The number of nodes (and therefore edges) grows as 2^n for n steps, so the problem size can become unmanageable if planning covers many time periods. Therefore, there are $2^n - n - 1$ free parameters for the n step problem, so solving all the flow subproblems does not scale well in the number of time steps. Regardless of the number of time steps, each subproblem is a capacitated transportation problem from d unit sources to 2^n sinks (for n time units).

As an alternative to the problem of solving multiple network flows (for each subproblem), the free parameter k and the associated constraint equations (4.11) could be combined to form a single linear program [Lue84]. For the two-step problem, this can be written as the combined system of the three size constraint equations and a set of constraints that allow only one strategy to be chosen for each of the objects in the collection. Thus we are trying to solve

$$\min \mathbf{c^T x} \text{ subject to } \mathbf{Ax} = \mathbf{b}, \mathbf{x} \geq \mathbf{0} \tag{4.13}$$

where $\mathbf{c}$, $\mathbf{x}$, $\mathbf{A}$, and $\mathbf{b}$ are given by the equations

$$\mathbf{c^T} = \begin{bmatrix} C_{OO_A} & C_{OX_A} & C_{XO_A} & C_{XX_A} & C_{OO_B} & C_{OX_B} & C_{XO_B} & C_{XX_B} & \cdots \end{bmatrix}, \tag{4.14}$$

$$\mathbf{x^T} = [\text{ vector of (1,0) flow assignments }], \tag{4.15}$$

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & \\ \vdots & & & & \vdots & & & & \vdots & & & & \ddots \end{bmatrix}, \text{ and} \tag{4.16}$$

$$\mathbf{b^T} = \begin{bmatrix} d & N_o & N_o & 1 & 1 & 1 & \cdots \end{bmatrix}. \tag{4.17}$$

The first three rows of the matrix $\mathbf{A}$ correspond to the observation set size constraints; these require the sum of all assignments to be d, the number of objects, and that the number of observations made during each time unit be N_o. Note that additional time steps will add one more row (constraint) per time step and will increase the number of columns in proportion to 2^n for n time steps. The remainder of the rows of $\mathbf{A}$ constrain the solution such that there is only one strategy chosen per node.

This linear program representation still represents a network flow, but established means for minimum cost flow cannot be applied here, since the size of the sink nodes is allowed to vary as part of the problem. Since this is not a traditional network flow problem, let us state it more formally for the benefit of future investigators who might wish to classify it and investigate its complexity:

The n-step binary strategy minimum cost flow problem
Consider a network having d source nodes, each of unit size, and 2^n sink nodes of size as described below. All source nodes are connected to all sink nodes by edges having unit capacity and real-valued, nonnegative cost proportional to the flow along the edge. A flow is represented by a vector $\mathbf{x}$ of flows along all edges; the proportional costs along these edges form a vector $\mathbf{c}$ of the same dimension. The i^{th} sink is labeled by the n-bit binary representation of i, $\mathbf{L}_i$. The j^{th} component (bit) of this vector is $[\mathbf{L}_i]_j$. The sink sizes S_i must obey the n constraints

$$N_o = \sum_{i=0}^{2^n-1} [\mathbf{L}_i]_j \, S_i \quad \forall \text{ integers } j \in [1, n]$$

where $N_o \leq d$ is an positive integer. We further require flow conservation, so that

$$d = \sum_{i=0}^{2^n-1} S_i.$$

The cost of a flow $\mathbf{x}$ is the product

$$C = \mathbf{x}^{\mathbf{T}}\mathbf{c}.$$

We seek the conservative flow $\mathbf{x}$ and corresponding sink sizes S_i that minimize C.

The bits within the labels $\mathbf{L}_i$ in this statement correspond to "observe/don't observe" decisions at each time step.

Regarding the expected problem size, as well as the problem's complexity, it is sufficient to note two things: first, for the method to accurately represent the change probabilities, the time step size must be small enough that change probabilities remain relatively constant over that period. Second, once a time step size is chosen, planning must be over a large enough number of time steps to account for the long-term behavior of the more slowly changing objects. These two factors can easily create a situtìon in which a large number of time steps are necessary for good long-term performance. The resulting network may be very large, since the sink (strategy) nodes grow exponentially, as do the edges that connect them to the source nodes. Moreover, the cost calculation along each edge must be repeated with every time step. As a result, more practical means of solving the problem are needed if the method is to be applied on a large scale. Such specialized algorithms might make the problem less intimidating for large numbers of objects, observations, and planning steps (time units).

Since the graphs involved may have a large number of nodes, approximation methods for obtaining near-optimal flows are important. Approximation will be indispensable in a system where global optimality is less of a concern than execution speed. The next section concerns optimization in the continuous time domain, using two key assumptions: first, that documents change memorylessly, and second, that many documents having similar properties can be assigned a single periodic observation strategy.

4.2 Optimization with assumptions

In this section, we will explore the optimization of an observation strategy in the presence of two approximations. First, we will assume that objects change memorylessly, according to exponential lifetime distributions. For such an object collection, the scheduling problem is reduced to determining an optimal observation *period* rather than a sequence of observations to be made on successive time steps. This scheme presents a tremendous advantage, since an object can simply be assigned a single observation period that remains valid as long as the object stays in the index and has a stable change rate and popularity. As estimates of change rate and popularity are updated or improved, the object is simply recategorized and assigned to a new fixed period schedule.

When this reduction in complexity is available, we can afford to include the importance of monitored objects in forming an observation strategy, expanding our view of information sources in general. The information landscape can be mapped nicely onto three factors, each of which can be properly taken into account when building an observation schedule. First, there is the underlying change rate of a resource. By some definition of change, however complex, one can construct a statistical model of how often an object is expected to change, just as has been done throughout this work. Second, there is the demand for that resource. How interested are users in knowing what the resource has to offer, and consequently, how much would they like to know about changes to that resource? Third, there is the expected magnitude of the changes that take place. The effect of this is that some changes will cause more error in the index than others. A large change will make the index entry for an object completely wrong, while a small (cosmetic) change will not have a large effect on the accuracy of an index entry. This distinction was addressed in Chapter 3, where we considered the types of changes that happen to web pages. Similar analysis is possible in other domains as well; it is probably best to normalize change in such a way that all changes can be thought of as having the same magnitude. This removes the effect of the third factor, leaving us with (effectively) a two-dimensional space of information sources.

We seek to optimize observation by partitioning the space of objects to be observed into groups of similar documents, and then treating all objects within a single group in the same way. A group of objects will have similar change rates, change magnitudes, and demand. The reasoning for treating all objects within a group in the same way is simple. When objects in a group change at the same rate, and there is no attempt to arrange the observations in a particular order, there is no reason to sample any of them any faster or slower than any other. This is only true for objects that change memorylessly, since the probability of having changed is simply a function of the time since the last observation, as given in (2.35). The idea behind uniform sampling is to keep the probability of change the same for all objects within a group. A much more formal discussion of this is given in [CLW97].

The observation system consisting of a single queue that uses all the bandwidth available is the best solution when all objects under observation change memorylessly and at the same rate. Naturally, this system might manifest as a large number of parallel but identical queues which are used to pipeline requests like instructions in a computer. Of course, in a system for which change rates and values differ between objects, the collection can be subdivided into classes that require the same observation strategy.

The idea of dividing the collection and applying a uniform treatment within each bin is analogous to non-uniform quantization of binary signals[Cou93] to maximize information. One example of this is $\mu-$law encoding of sounds[1]. In this case, the objective is to maximize the information transmitted by a given number of bits by making every symbol equally probable. This partition is known to maximize the entropy, $\sum_i [-P_i \ \log_2 P_i]$, which is an aggregate function of the bin size boundaries and the sound for which they are encoded. A comparison of uniform and nonuniform partitions is shown in Figure 4.3, where two possible 4-bit quantizations of a speech signal are compared. The optimization method presented in this section is quite similar in spirit: we seek an optimal selection of bins for the monitored object space, as well as a rate allocation into each bin, such that the overall

[1]As is found in files having a "`.au`" extension, as opposed to "`.wav`" files, which are uniformly quantized.

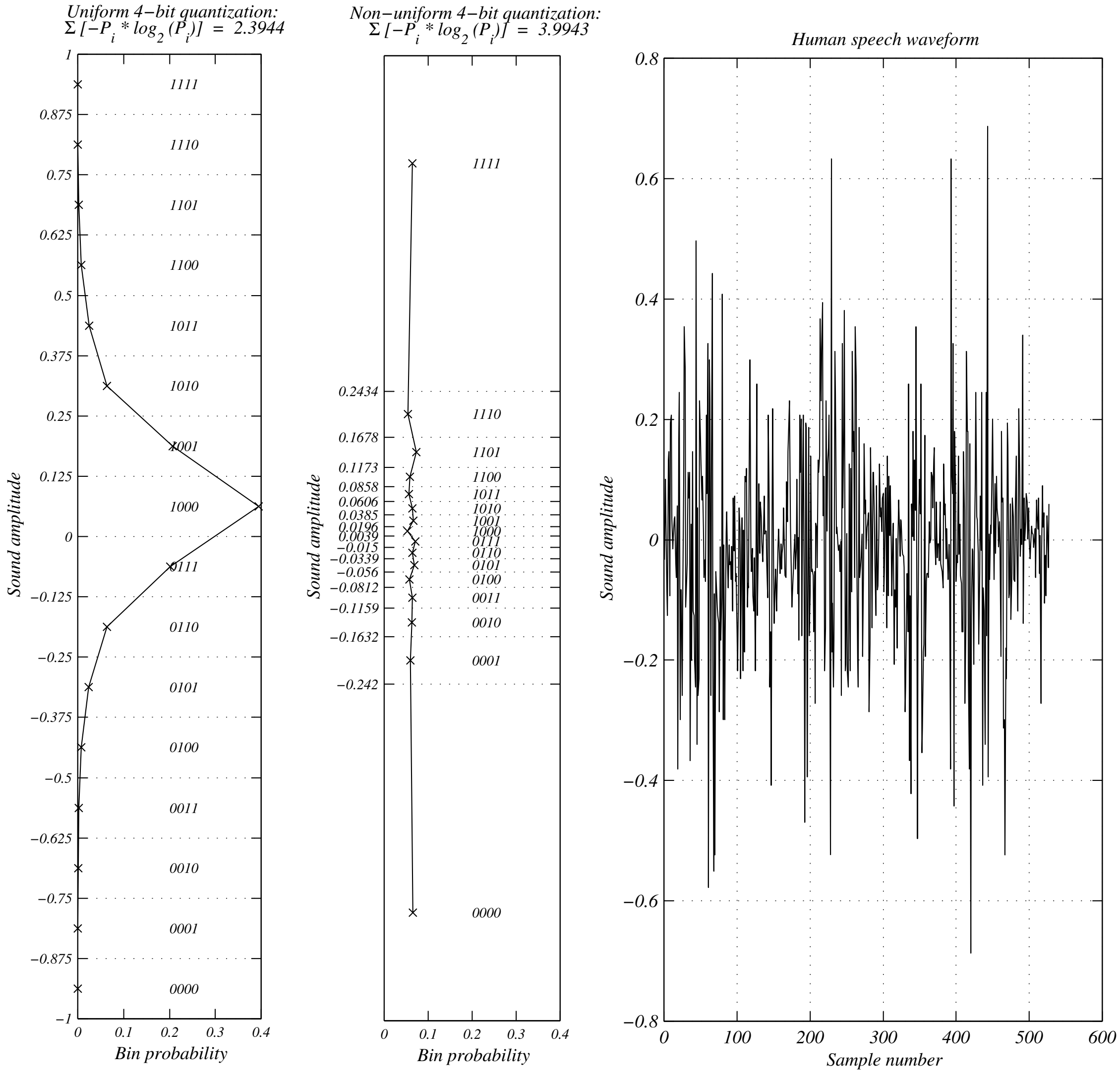


Figure 4.3: **Sound encoding: example of nonuniform quantization for optimizing an aggregate function** For the sound signal on the far right (which is a sample of human speech), we show two quantizations on the left. At far left is a 4-bit uniform quantization, for which this signal yields an entropy of 2.39. The center plot shows a non-uniform quantization in which all bins are close to equiprobable, thus making the entropy almost maximum (3.99 bits per symbol being very close to 4). This quantization was found by selecting the bin boundaries to maximize entropy; we will quantize monitored objects to optimize the probability of their being correctly indexed.

probability α of being β-current is maximized for this system. Obviously, the probability weighted sum of $-\log_2 P_i$ is much easier to work with than our complex objective functions, as we will see. We first discuss the division of rates, then the bin sizing, and last present the optimization routine along with some test results.

4.2.1 Rate-equivalent partitioned systems

The main constraint to be applied in subdividing a single-queue system is the conservation of bandwidth. The overall processing rate is to be held fixed in making any partition. Specifically, any system composed of a single queue Q_0 operating at processing rate λ_0 can be subdivided into N smaller queues Q_i each having rate λ_i, such that the sum of the rates matches the single-queue rate.

$$\lambda_{max} \geq \lambda_0 = \sum_{i=1}^{N} \lambda_i \tag{4.18}$$

Here λ_{max} is the maximum available bandwidth which will generally exceed the λ_0 actually devoted to downloading. Practically speaking, when N becomes large, it may be difficult to achieve this balance due to the overhead involved in splitting the workload. A realistic observation system will eventually suffer from over-parallelizing a single queue. Where this balance point lies is largely a matter of the application under consideration and the performance of the observers in question.

Consider a queueing system in which M objects are being observed in round-robin fashion. That is, an object is observed, then is moved to the back of the queue. This object then moves forward as objects are observed and rotate in behind it. When placed in a single queue, each object is observed on average once every T_0 time units (where by definition, the observation rate $\lambda_0 = M/T_0$). Now, assume these M objects are then distributed amongst N round-robin queues, each of which contains M_i objects. Objects in queue i are processed once every T_i time units, so that $\lambda_i = M_i/T_i$. Rewriting the rate conservation expression (4.18) using the sizes of each of the queues,

$$\frac{M}{T_0} = \sum_{i=1}^{N} \frac{M_i}{T_i}, \tag{4.19}$$

or in terms of the probabilities

$$P_i = \frac{M_i}{\sum_{i=1}^{N} M_i} \tag{4.20}$$

we can write (4.19) as

$$\frac{1}{T_0} = \sum_{i=1}^{N} \frac{P_i}{T_i}. \tag{4.21}$$

A simple example of such a rate-equivalent partition, and one that finds application in many actual systems, is to turn a single queue into a number of equal size, equal rate queues, each of which runs at a fraction of the single-queue rate. This is like choosing between one supermarket line at rate $n\mu$, versus n lines each running at rate μ. These kind of rate-equivalent systems are discussed in standard texts on queueing theory [Coo72] and will be the foundation of the division of rate among partitions in an observation system.

4.2.2 Subdividing an observed population

The first method we apply to split the observed objects into classes is to group them by change rate. This division takes the form of a N-way partition $[0, P_1, P_1 + P_2, \ldots 1]$ of the population's CDF of change rates, $F(t)$. For the moment, assume that the population is sampled uniformly by the users of an index, so that the distribution of accesses is the same as the population's distribution. Each value of P_i defines the fraction of the population in bin i. One such division of a population into four bins is diagrammed in Figure 4.4. The partition is visible along the vertical axis of this figure. Each value P_i is the fraction of the population contained in bin i. Note that the partition can also be expressed in terms of mean change times, so that a partition is a set of times $[0, t_1, t_2, \ldots, t_i, \ldots \infty]$. This equivalence is shown in Figure 4.4 as intersections along the horizontal axis.

After subdividing the population by mean change time, a processing rate can be assigned to each bin in such a way as to satisfy (4.21). Once the bin boundaries and the processing rate are set, the probability α_i of that bin being β-current can be determined. For all NT bins, the probability α that a randomly selected object's index entry is β-current is just a weighted sum of the probabilities

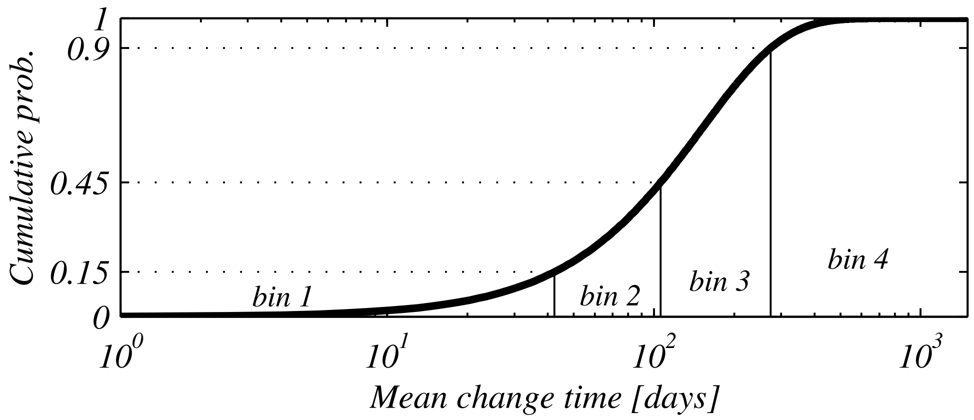


Figure 4.4: **A partition of the mean change time CDF.** To subdivide a population of observed objects, the mean change time CDF can be partitioned into bins of similar change rate. Along the vertical axis, the CDF is partitioned such that the height of each bin is the fraction of the population that would be included in that bin. These intersect the horizontal axis at a corresponding set of mean change times. This illustration uses four bins of arbitrary sizes, but in principle, any number or size of bins could be used.

α_i, so that

$$\alpha = \sum_{i=1}^{NT} P_i \alpha_i \tag{4.22}$$

Keep in mind that the probabilities P_i are both access probabilities and population probabilities. Later in this discussion the two probabilities will be different.

Finding each of the α_i is almost exactly like finding α for the single-queue system, as was calculated in (3.17). The only difference is the range of integration and the access distribution used. Given a PDF $f(t)$ of the mean time between changes for an imposed demand, and the corresponding CDF $F(t)$, the integral to find α_i for each bin is

$$\alpha_i = \int_{t_{i-1}}^{t_i} \left[\frac{f(t)}{F(t_i) - F(t_{i-1})} \right] \left[\frac{\beta}{T_i} + \frac{1 - e^{-(1/t)(T_i - \beta)}}{T_i / t} \right] dt \tag{4.23}$$

The first half of the integrand is the probability distribution of mean change times within the bin. The second half is the familiar relation giving the probability of β-currency for a particular mean change time, t. For web pages, if we use the Weibull distribution of mean change times that was developed earlier, this integral can be evaluated numerically for any partition and set of observation rates T_i.

The function $f(t)$ can also be thought of as a *marginal* distribution of mean lifetimes, where the dependence on resource value has been integrated out. Clearly, it is entirely possible to have a class of objects that change at the same rate but have widely varying value. When there is a wide variation in access probability across pages at a single change rate, it can be advantageous to add a dependence of the access distribution on resource value.

How one should represent value is another matter; we provide three possibilities. One interpretation of an object's value within an index is in terms of the number of references returned by the index to that particular object. Obviously it is more important to be correct with respect to the objects that are always returned to users. A second possibility for a value representation is a measure of how frequently users follow a reference provided. For a web search engine, this would translate into the a user selecting and following a link to a result. Many of the major engines either have used or are using tracking methods for gathering this data, by capturing and redirecting user selections (this is true for Excite, Lycos/Hotbot, Direct Hit[2] and others). This is a convenient means for engines to obtain simple relevance feedback.

The third approach to defining the value axis, which can be applied in the absence of direct user feedback, is some external measure of the value of the page within the index. Presumably this value would be positively correlated with popularity and would therefore be a good discriminant for dividing the population. Nonetheless, this kind of normalized measure of value may have no direct correlation with the frequency of access or usage–the Google search engine, for example, utilizes a measure called PageRank [BP98] that indicates how many documents link to a given document. Additionally, their algorithm weights links from high-rank pages such that they have a greater contribution to the rank of a given page. That is, a link from Yahoo counts much more than a link from a personal homepage. In some sense, having a high rank indicates that it is more important to keep track of that document, user references aside. This feature has helped elevate Google to

[2] http://www.directhit.com/

the top of user satisfaction surveys [Goo99]. Being what appears to be a good indicator of resource importance, it would make a natural measure of normalized value as well.

When the population was partitioned only by mean change time, it was implicitly assumed that the population of documents was uniformly sampled by the users of the index. That is, all objects were equally likely to be demanded at any time. Obviously, when the notion of value used is the access frequency (or the mean time between accesses), there is a clear relation between the distribution of this factor within the *population* as opposed to its distribution among *accesses*, and the distribution of accesses is not necessarily the same as the distribution of the population.

Let us explore this further. We will discuss a direct method where an object is described by two parameters: a mean time between changes, and a mean time between accesses (inverse demand, if you like). One can estimate the shape of this *population* distribution from index usage and observed changes. How the population is sampled to determine the probability of β-currency is another matter. This is where the mean time between accesses becomes important. The sampling distribution, or how many accesses are made relative to other resources, follows directly from the definition of "mean time between accesses". For example, we know that the number of accesses made for any object having a mean time τ between accesses should be twice that (on average) of an object having mean time between accesses of 2τ. Thus we have that the sampling distribution must obey

$$v(\tau) = k\ v(k\tau), \tag{4.24}$$

for any positive k, from which it follows that

$$v(\tau) = C_0 \tau^{-1} \tag{4.25}$$

is a valid scaling function. Here C_0 can be chosen for normalization. The access PDF that follows is the product of the scaling function and the population PDF, suitably renormalized so as to remain a PDF. As such, the population density must fall off faster than $1/\tau$ as $\tau \to 0$. Otherwise, the access PDF would be infinite at mean access time 0, which is unacceptable. Summarizing, the access PDF $h(t, \tau)$ is related to the population PDF $f(t, \tau)$ by

$$h(t, \tau) = v(t, \tau) f(t, \tau) = \frac{C_0 f(t, \tau)}{\tau} \tag{4.26}$$

with C_0 chosen for normalization. Using a population constructed such that the marginal density with respect to mean time between changes is the same as that determined for the web in an earlier section (see Figure 3.22), Figure 4.5 shows how the population and access densities are related.

With this notion of value, the access density $f(t)$ can be expanded to be a joint distribution $f(t, \tau)$ that includes a dependence on the mean time between accesses. In this way, the integral (4.23) is altered to account for the possibility of having monitored objects that have the same change rate but differ in terms of access frequency. The integral then becomes a double integral over mean time between accesses, τ, and mean change time, t. For a rectangular area in this space, the probability of a random access being β-current is

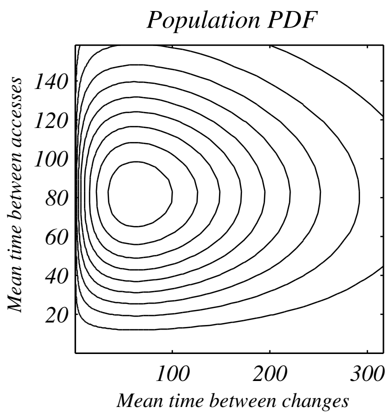
$$\alpha_{ij} = \int_{\tau=\tau_{j-1}}^{\tau=\tau_j} \int_{t=t_{i-1}}^{t=t_i} \left[\frac{h\left(t,\tau\right)}{P_{ij}}\right] \left[\frac{\beta}{T_{ij}} + \frac{1 - e^{-(1/t)(T_{ij}-\beta)}}{T_{ij}/t}\right] dt d\tau. \tag{4.27}$$

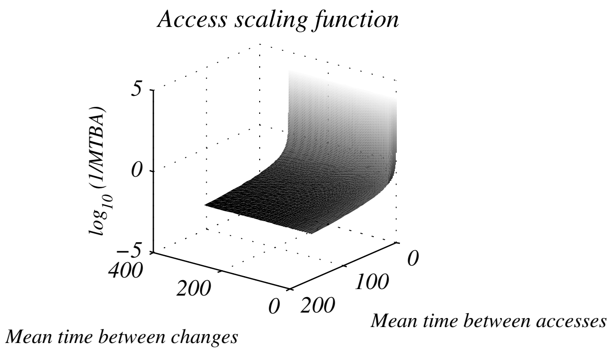
To find all the α_{ij}, the τ axis is sectioned like the mean change time axis was above, so that the partition $[\tau_{min}, \tau_1, \tau_2, \ldots, \tau_j, \ldots \tau_{max}]$, in conjunction with a partition of the t axis, defines blocks in which to evaluate (4.27). Finally, to normalize the contribution of each α_{ij} to the total sum α, we divide by the probability P_{ij} of a given access falling within bin ij. This probability is defined as

$$P_{ij} = \int_{\tau=\tau_{j-1}}^{\tau=\tau_j} \int_{t=t_{i-1}}^{t=t_i} h(t,\tau) dt d\tau. \tag{4.28}$$

Note that this differs from the probability of a given population member lying within this region. We also point out that (4.27) could be factored differently to account for the fact that there is no dependence of the inner probability term on τ, but only on the mean change time t. This might speed the calculation slightly. Additional weighting functions that skew the probability result according to τ may also be added, but this does not change the fundamental form of the problem, only the inner term of the integrand.

Additionally, the value chosen for β could reflect a greater or lesser tolerance for error. A function $\beta(t, \tau)$ might have larger values of β for low-access or slowly-changing objects, but very low values of β for fast-changing, popular objects. The choice of this function will be directed by examining objects within each bin, and making a determination of how much error is tolerable. One reasonable





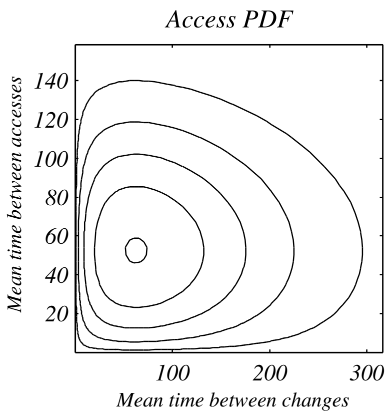


Figure 4.5: **Progression from the population PDF to the access PDF** In the top plot, we have a hypothetical population PDF $f(t, \tau)$ that depends jointly upon the mean time between changes (MTBC) t and the mean time between accesses (MTBA) τ. The function was chosen such that the access times and change times have the same shape (all conditionals are just scaled versions of the corresponding marginal distribution). The center plot shows the scaling function τ^{-1}. Last, the lower plot is the PDF $h(t, \tau)$ that results from multiplying the top two plots and renormalizing. Note how the peak in this plot has been shifted to a lower mean time between accesses, and the contour lines have been flattened and compressed near the MTBC axis.

choice in the absence of better information would be the plane $\beta = k_t t + k_\tau \tau$, where the constants are chosen to give a value of $\beta = \nu T$ for the slowest changing, least popular object (the upper right corner of our plots). For this object, some large fraction of changes would be forgiven. The plane is the simplest choice; any desired function could be used. The idea is to reflect a decreased concern for instantaneously correct indexing of more slowly changing or less popular objects.

Finally, the expected probability of being β-current with respect to a demand distribution is found as in (4.22), by summing the contributions from all bins ij, over NV access times and NT mean change times:

$$\alpha = \sum_{i=1}^{NT} \sum_{j=1}^{NV} P_{ij} \alpha_{ij}. \tag{4.29}$$

4.2.3 Population partitioning for α maximization

Optimal observation within this scheme translates into creating a partition and bandwidth allocation for a given number of time and value bins that maximizes (4.29). Let us summarize the problem we are solving:

> **Rate allocation and population partition to maximize** $\Pr(\beta -$ **current**$)$
> Consider a collection of objects that change according to independent Poisson processes. The objects are described by a probability density $f(t, \tau)$ over the mean time t between object changes, and the mean time τ between user accesses to an object's index entry. The total observation rate available is λ_0 objects indexed per unit time. The objects are divided into regions R_i, each of which is allocated a portion λ_i of the total rate. The sum of the rates is constrained by
> $$\lambda_0 = \sum_i \lambda_i.$$
> The probability α that the index is β-current is given by a sum
> $$\alpha = \sum_i P_i \Pr(R_i \text{ is } \beta\text{-current})$$
> where $\Pr(R_i$ is β-current$)$ is as stated in equation (4.27), and P_i is the probability of an index access in region R_i. We seek the rate allocations λ_i and the associated regions R_i that maximize α.

In general, this is a very high-dimensional problem space, having many locally optimal solutions. Additionally, the solution set is constrained by rate conservation (4.21) and cardinality constraints such as having non-overlapping bins that completely cover the object space. Obviously, since the terms of (4.22) and (4.29) result from integrals like (4.27), the problem is nonlinear as well.

Solutions **S** to this problem take the form of a partition of (t, τ) space (which are mean time between changes and mean time between accesses, respectively), and a rate allocation among the bins. We implement the partition as a rectangular grid over the space of mean change times and object values, although any partition of the space would suffice. Partition schemes having finer quantization in regions where the solution is sensitive to quantization errors should perform better, for example. In our simple rectangular case, a solution consists of:

- **S.change_times** : a set of $NT + 1$ time grid lines; the first and last boundary lines are fixed.
- **S.access_times** : a set of $NV + 1$ value grid lines
- **S.rate_matrix** : a matrix of $NT \times NV$ rates, corresponding to the bins formed by the grid lines

We show an example partition of the population in Figure 4.6, which uses the same population and access density that is used in our examples on the web, shown in Figure 4.5.

Any optimization method used in this problem must be robust enough to work with the constraints laid out above, and also be able to search for a globally optimal value (rather than a locally optimal value). For its simplicity and ability to search for global optima, we describe a simulated annealing ([Hay94] for example) approach to maximizing α for a given value of β (or a function $\beta(t, \nu)$).

The idea in a simulated annealing approach is to have a means for moving from one solution to another candidate solution that is within some local neighborhood, then to swap solutions under certain conditions. Specifically, the new solution's fitness is evaluated and compared with the present solution. With an ever-decreasing probability, a worse solution will be accepted, while a better solution is always accepted. In this way, the method may descend into a local performance valley in order to ascend a performance hill on the other side, although the probability of taking downhill steps is reduced as time advances to force convergence on an optimal value. For high-dimensional spaces like this problem presents, it is necessary to have a schedule that enables the method to explore a large part of the space before settling into a neighborhood of the global optimum.

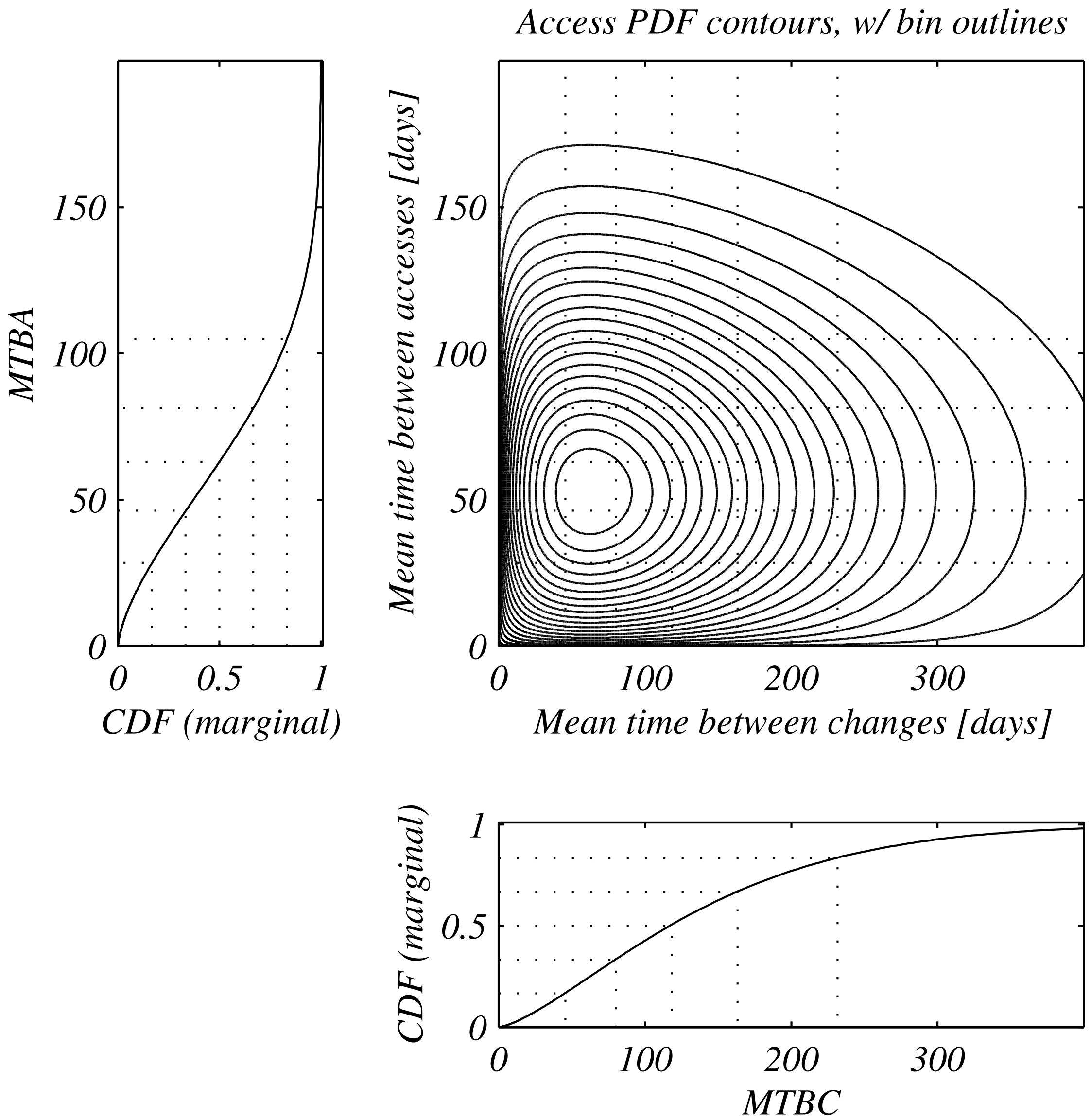


Figure 4.6: **Partition of an access PDF** This figure shows an example partition of a population, derived by splitting the marginal distributions (the left and bottom plots) into equiprobable bins. The resulting partition of access space has bins which are close to equiprobable. It is not possible in general to choose $NT + NV$ gridlines that result in $NT \times NV$ equiprobable bins. For that matter, bins with equal probability are not necessarily the optimal partition of the space, either. This is, however, a good initial partition for most problems.

Three things are necessary for implementing a simulated annealing algorithm:

- A function for evaluating the fitness of a solution **S**
- A function choosing an additional solution within a local neighborhood of solution **S**, chosen such that all solutions are reachable from all other solutions by a connected series of neighborhoods
- A schedule for decreasing the probability of choosing a less optimal solution to replace the present one, or a *cooling schedule*

We have already described the function for evaluating the fitness of a solution **S**; the various manifestations of (4.22) and (4.29) accomplish this. For our neighborhood function, we will use a function which either "nudges" a randomly-chosen boundary line, or swaps a bit of check rate between two randomly chosen bins. A connected series of such neighborhoods will be able to reach any solution in the space. The function for choosing new candidate solutions from a local neighborhood is best expressed as pseudo-code:

```
if (rand(1)< (1/NT*NV) ),
  # -- nudge a boundary value, somewhat infrequently

  if (rand(1)>0.5),
    # -- nudge a mean change time boundary
    time_boundary=select_random_time_boundary();
    nudge_time(time_boundary);
  else,
    # -- nudge a value boundary
    value_boundary=select_random_value_boundary();
    nudge_value(value_boundary);
  end

else,
  # -- swap some check rate from one bin to another

  bin1=select_random_bin();
  bin2=select_different_random_bin(bin1);
  swap_some_rate(bin1, bin2);
end
```

When nudging a boundary, the check rate is kept constant for all bins that change in size (namely the bins on either side of the boundary that moved). This is done by balancing pairs of terms in (4.21) such that any change in P_i is offset by a proportional change in T_i. Likewise, when exchanging the check rate between two bins, the boundaries of the bin are not adjusted. For both nudging boundary lines and moving rate from one bin to another, we use a maximum percentage change. That is, the maximum rate exchanged from one bin into the other is some percentage of the smaller of the two values. The choice of this "damping factor" is discretionary. Values around 25% were good for coarse exploration of the solution space, while smaller values were good for fine-tuning in the neighborhood of a solution thought to be globally optimal.

Of course, any partition of the object space is acceptable; we will only discuss the problem for a rectangular grid. The problem becomes more difficult with irregularly shaped partitions, since integrating (4.28) and (4.27) over these regions may be challenging. Many possible partition schemes consist entirely of rectangular pieces though not necessarily defined by grid lines. Such schemes would be equally easy to integrate, but would require slightly greater algorithmic complexity.

4.3 Optimizing web observation

This subsection will outline the use of the above optimization scheme for a hypothetical search engine. Throughout this discussion, it is assumed that it will be possible to determine the distributions $f(t, \tau)$ and $h(t, \tau)$ by experiment. Once these functions are available, running the above algorithm is straightforward; the integrals are evaluated numerically. The examples to come are computed using a hypothetical population PDF and CDF in which the mean time between accesses is a scaled version of the mean time between changes.

4.3.1 Effect of varying total bandwidth

A number of experiments were run using different amounts of total bandwidth. As in (4.21), the total time T_0 to re-index the entire collection gives the rate budget $\lambda_0 = 1/T_0$. In summary, our experiments demonstrate that the optimization yields only modest gains for observers that have a large amount of bandwidth (meaning, T_0 is only a fraction of the average t for the access distribution). Conversely, when observers are overworked and T_0 is much larger than the average value of t for the access distribution, the optimization can improve performance significantly. Recall from the

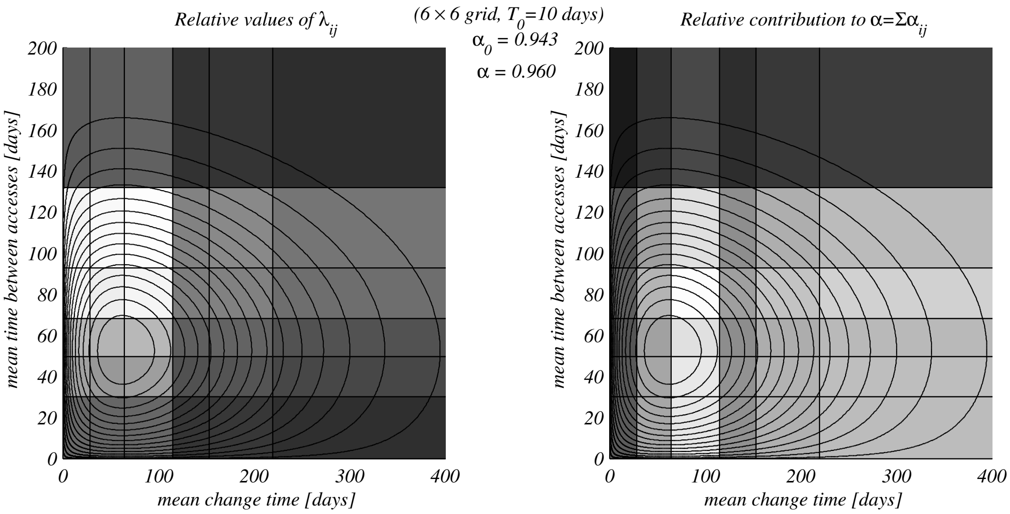


Figure 4.7: 6×6 **grid,** $T_0 = 10$ **days** With so much bandwidth available, the gains resulting from the optimization are minimal, yielding a performance increase of only about 2% from the single-queue case, α_0.

parameter estimation shown in Figure 3.22 that the average mean time between changes for this population is 138 days. Generally the performance increase for the overworked observer comes at the expense of fast-changing, relatively unpopular resources that are all but ignored, so that the rate that might have been allocated there can be spent elsewhere.

For a 6×6 grid, and a value of $\beta = 1$ day, Figures 4.7 through 4.11 show the best solutions discovered in 2×10^5 steps, which took about 8 hours per solution on our machine[3] Each figure in the sequence shows the effect of a reduction in bandwidth, spanning the total re-indexing times $T_0 = 10$ days to $T_0 = 300$ days.

These figures contain a number of pieces of information. First, all the plots show the contours of the access PDF (as was used earlier to generate Figure 4.5) superimposed on two images. The left-hand image shows the relative allocation of rates amongst the bins, where light color denotes a high rate allocation and dark indicates a low rate allocation. The right-hand image shows the relative contribution of each bin's α_{ij} to the total sum α, as in (4.29). Again, light color indicates

[3]The machine in question is an Intel Pentium III 600 MHz running Windows NT 4.0, having 512 MB of RAM and a 9.1 GB ULTRA SCSI II disk. Our implementation uses MATLAB 5.3R11.

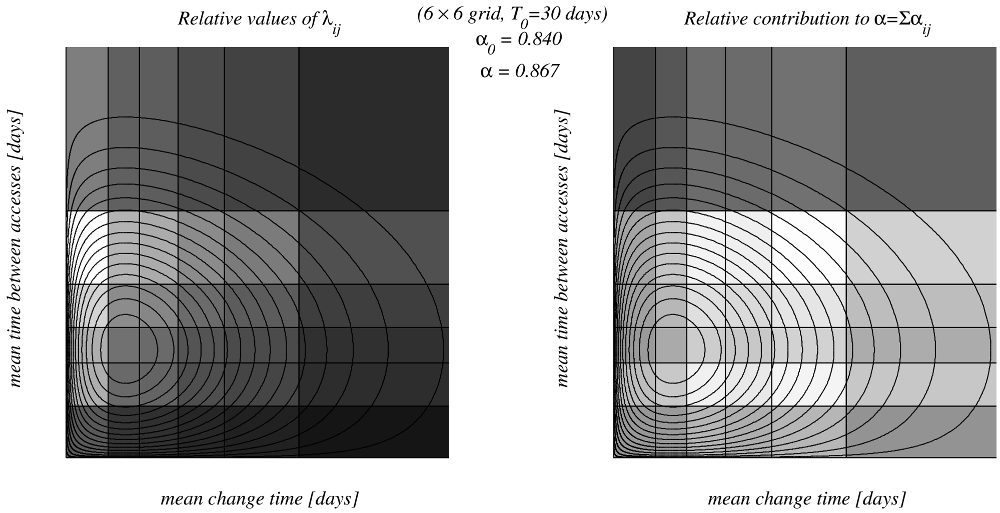


Figure 4.8: 6 × 6 **grid,** $T_0 = 30$ **days** The rate allocation in this solution is somewhat curious in that it appears the upper left-hand bin could have its share reduced. As above in Figure 4.7, the percentage gain from the optimization is fairly small, only around 3%.

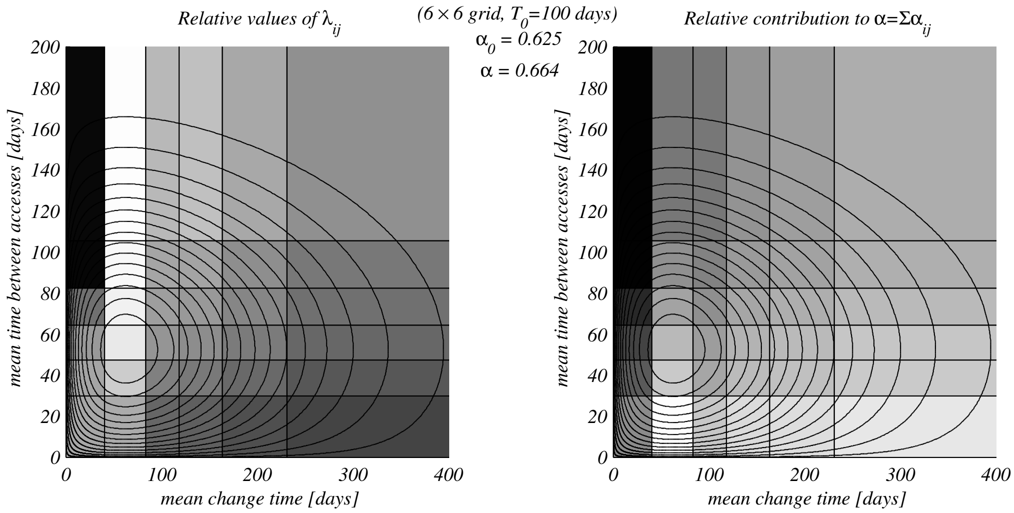


Figure 4.9: 6 × 6 **grid,** $T_0 = 100$ **days** Here, where bandwidth is only a tenth of that available in Figure 4.7, the optimization begins to pay off somewhat more. Note also that the fast changing, low-popularity portion of the collection is beginning to be ignored.

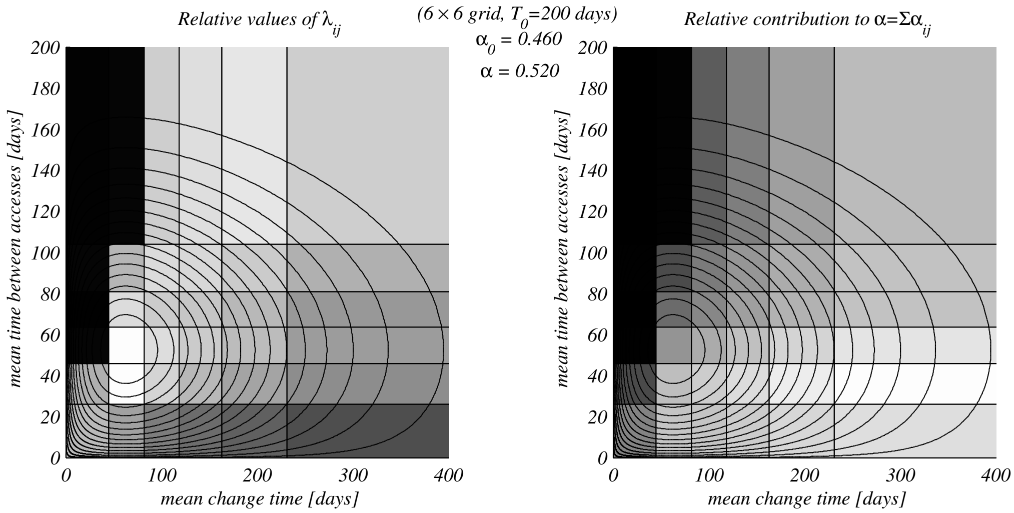


Figure 4.10: 6×6 **grid,** $T_0 = 200$ **days** Comparison of this figure with Figure 4.10 shows how the ignored set grows as bandwidth is reduced. Optimization yields around a 15% (relative) improvement.

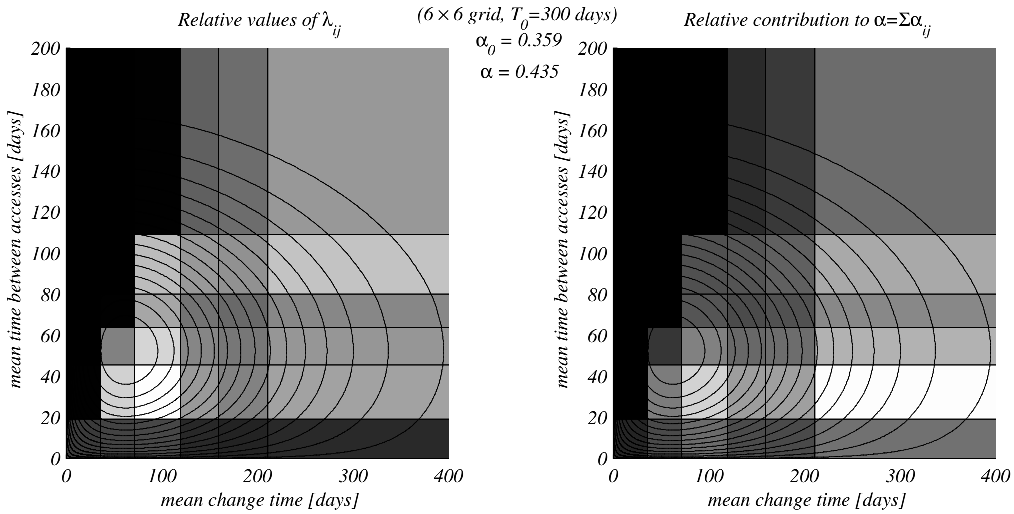


Figure 4.11: 6×6 **grid,** $T_0 = 300$ **days** When bandwidth is reduced even further, the optimization continues to be more worthwhile, at the expense of the low-popularity, fast-changing document set. The performance increase is around 20% (relative).

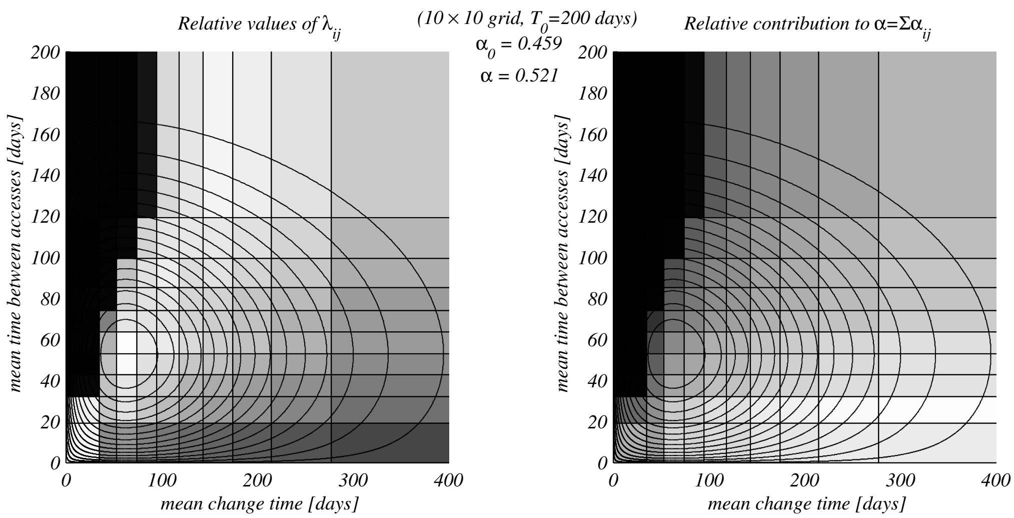


Figure 4.12: 10×10 **grid,** $T_0 = 200$ **days** This figure shows that a slight improvement can result from the addition of more grid lines, but it is very much a case of diminishing returns. The only benefit of having a 10×10 grid is a few tenths of a percent performance improvement. From the numerical side, this simulation is valuable in that it shows how the optimal rate allocation is very similar to that in Figure 4.10. Both appear to be discretized versions of the same surface. This fact could be useful in refining the optimization technique used since a finer quantization could be used along this line.

a large contribution and a dark color is a small contribution. Both plots show the strategy from different perspectives. On the left-hand side, we get a feel for the amount of "effort" expended to receive the benefits shown on the right hand side. For each plot, at the top of the figure between the two images, we state the probability α_0 that results from using a scheme with one large bin that uses all the bandwidth, or a single observation queue that runs at a set rate. This simple solution is the baseline against which other solutions are compared. It is calculated using a suitably altered version of (2.39) in which the access distribution $h(t, \tau)$ has replaced the population distribution $f(t)$. Each solution's value of α, as determined by (4.29), is listed just below the baseline value α_0.

In most of these examples, high rate allocation is correlated with a low contribution to the total sum α, up to a point. When the contribution to α is very near zero, the rate expended is near zero as well. Curiously, this indicates that most of the bandwidth is spent watching objects that lie along a certain boundary line, beyond which almost no effort is expended. The situation is especially pronounced when T_0 is large, as in Figures 4.9 through 4.12. This is exactly the situation described in the introduction, where we presented the choice between observing 30 different monthly-changing sources and a single daily-changing source. The best solution was to watch the 30 monthly-changing sources and ignore the daily changing source, although we admitted the possibility that a very popular daily-changing source might not be safe to ignore. In all these example plots, the higher T_0 becomes, the sharper the boundary is between the observed and ignored sets.

4.3.2 Estimation of possible search engine benefits

There are two ways to estimate the possible benefits available to search engines if this optimization is applied. Both involve finding an estimate of T_0 for the engines. One possibility is to assume that search engines use a periodic re-indexing schedule, and use our experimental values of α (from Figure 3.26) to estimate T_0 from the values of α_0 in Figures 4.7 through 4.12. Each of these gives a value for α_0; by interpolation we can estimate what T_0 will be for an intermediate value of α. This is equivalent to just reading the figure off of the point on the surface in Figure 3.23 where $\alpha = \alpha_0$ and $\beta = 1$ day. When this is done, we find that the search engines would have values of T_0 roughly between 100 and 150 days. The second method for guessing T_0 is similar. We can note where each engine crosses the $\alpha = 0.95$ boundary in Figure 3.26, and use the data presented in Figure 3.24 to find a T_0 corresponding to that value of β. This is the method employed in Chapter 3, where it was estimated that typical engines have a value of T_0 between around 75 and 120 days.

Using an estimate of T_0, we can estimate the amount of improvement that could be expected were the algorithm applied for each search engine. On the faster side, for $T_0 \approx 100$ days, we have that $\alpha - \alpha_0 \approx 0.04$, so that the fast search engine in this spectrum might expect a $(0.04/0.62) = 6.4\%$ improvement. For a slower engine, closer to $T_0 \approx 150$ days, $\alpha - \alpha_0 \approx 0.055$, which would represent a $(0.055/0.47) = 11\%$ relative improvement.

Applying these percent improvements to the experimental $\beta = 1$ day performance figures, we estimate that the faster engines could have $\alpha = 64\%$ for $\beta = 1$ day (versus the experimental $\alpha = 60\%$), and the slower engines could attain $\alpha = 56\%$ for $\beta = 1$ day (versus 50% by experiment). By experiment, between 40 and 50% of pages had changed at least once since last visited. If our algorithm is applied this range could be 36 and 44% instead. Again, this assumes that the search engines tested currently use some form of periodic re-indexing.

Our examples have taught some valuable lessons. First, when bandwidth is plentiful, it may not be worth the trouble to attempt to divide a collection in this way in order to increase α. When we used $T_0 = 10$ days, about 1/14 of the expected mean time between changes for web pages, there was only a 2% performance improvement in α for $\beta = 1$ day, from about 94 to 96%. Optimal solutions in the high-bandwidth case are only slightly better than extremely simple ones, and it is surprisingly easy to construct sub-optimal solutions. However, when the optimization is run for $T_0 = 600$ days, the value of α increases from the round-robin value of around 0.2 to a value closer to 0.3. In situations like this where bandwidth is relatively scarce, the optimization becomes more and more worthwhile, and generally succeeds based upon ignoring the portion of the population that is low-popularity but fast-changing. Examples of such low-bandwidth scenarios include observations made by single users on wireless links—remember that the idea of (α, β)-current can be applied to any observer, not just search engines.

Additionally, it can be seen that a very effective way to increase the probability of an engine being β-current is to shrink the collection by eliminating redundancy. This is a huge area of research unto itself. As has been mentioned in [LG99], there is a definite tradeoff between index size and how up to date the index is. Since most queries can be answered with a much smaller set of documents, reduction of collection size can pay great dividends without too much adverse impact on user satisfaction. The portion of the collection that remains will have a much greater probability of being β-current.

4.3.3 Further considerations for observing the web

The algorithm's performance depends upon the ability to accurately lump documents into bins of similar change rate and similar value. To the extent that this can be done, the algorithm will perform as advertised. However, as addressed earlier in the thesis, it can be quite difficult to obtain accurate estimates for the rate of change when perhaps only one or two changes may have ever been observed for an object. Conversely, fast-changing object will present plenty of data on change rate. Likewise, it is simple to make an accurate determination of mean time between accesses for very popular resources, but it can be very difficult to do so for infrequently accessed ones.

It would seem logical that a search engine would have many more samples from which to estimate the access distribution than it would for changes in those resources, since access rates are generally faster than change rates. Search engines record millions of hits per day, each of which involves many accesses (whether "access" means following a link or just returning a reference). This is probably larger than the number of pages re-indexed per day, many of which will not have changed since the last observation. As mentioned in the discussion of age and lifetime distributions, the only reliable means of estimating change rates in the absence of a growth model is to average sampled lifetime lengths. As such, it may take many observations before even one sample of a lifetime length is available. This disparity in the number of independent samples will generally mean that the variance in the estimate of a page's mean lifetime will be larger than the variance in the estimate of its mean time between accesses.

Let us consider the form of the variance to see how variance depends upon sample size. Assume the mean time between accesses τ is exponential, like the mean time between changes. In this case, the maximum likelihood estimator $\hat{\tau}$ for τ is just the average of the observed time intervals. Since each sample time τ_i between two accesses is exponential, their sum is Erlang distributed. Dividing by the number of samples (to average) simply re-scales the distribution, such that the distribution of averages x for N samples is Erlang-N with mean τ_0, or

$$\hat{\tau} = E_N(x) = \frac{1}{\Gamma(N)} \left(\frac{N}{\tau_0}\right)^N x^{N-1} e^{-Nx/\tau_0} \tag{4.30}$$

Unfortunately for our estimation problem, the standard deviation for this distribution only falls off as $\tau_0/\sqrt{N}$, so it can take a large number of samples to obtain reasonably accurate estimates. We show the distribution for some values of N and mean value $\tau_0 = 1$ in Figure 4.13.

The distribution (4.30) of means only uses observed lifetimes or ages, but using other data can give estimates that have lower variance. For example, the content of a document often suggests in some way that the author does not intend to change it. This is often true of news articles or discussion group postings. It might be possible to identify such content stylistically or by some other means. In addition, the dynamics of documents at the same site can be related. Obviously if many documents have the same maintainer, they may also change with comprable rate. Use of these additional properties to guess change rates in the absence of a large number of lifetime or age samples will be necessary to bring the variance in estimated lifetime down to a more reasonable level.

When the estimates of mean change time and mean access time are poor, it may do more harm than good to place an object in a bin that might be incorrect. When these parameters are uncertain, ideal divisions of the access/change rate space along crisp grid lines are blurred by the estimation errors that may have occurred. The net result is a partition in which many objects are in the wrong bin and are not sampled correctly.

One lesson to be learned is that the local density of partitions of the object space should coincide with the quality of the estimates used to place objects in those bins. That is, if it will generally be the case that only two or three samples are available, there is no sense in dividing the collection along twenty grid lines. There is no assurance that it will be possible to accurately place objects in those bins; the variance in the estimator might cause an object to land two or three bins away from where it should be. Alternatively, if it is relatively easy to obtain an accurate estimate for τ

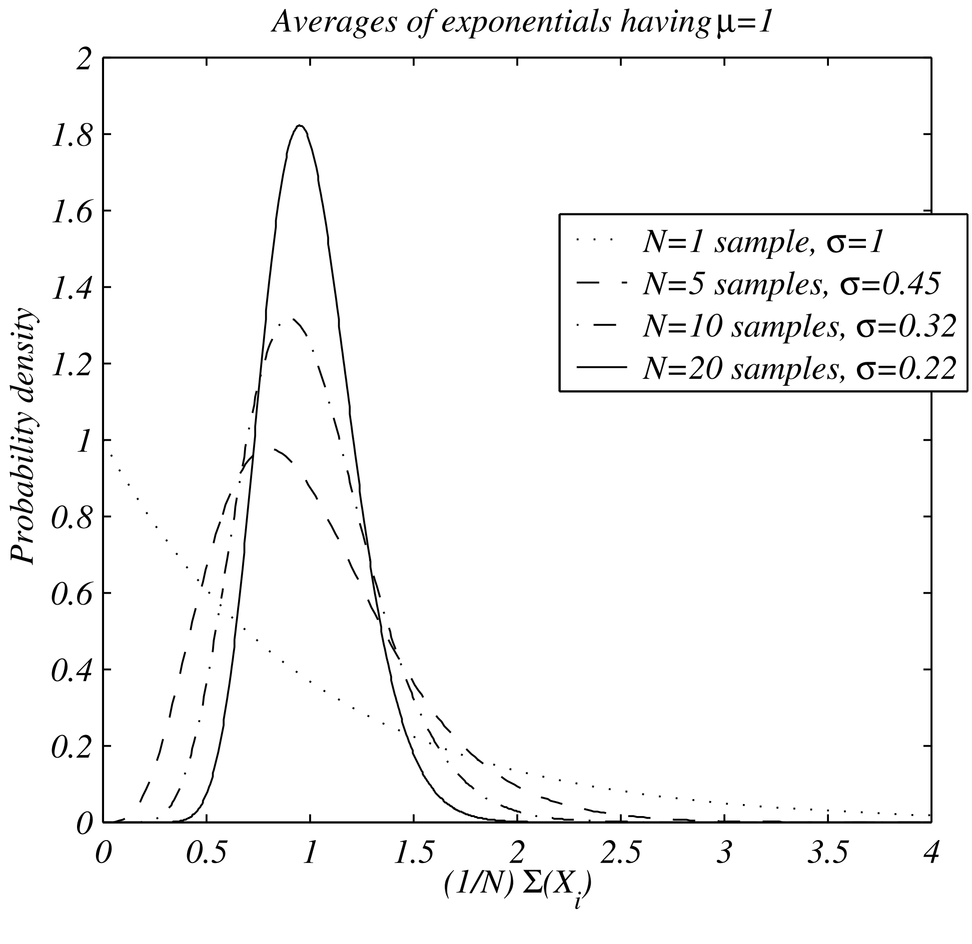


Figure 4.13: **Average of N exponential random variables** The average of N exponential random variables having mean value $\tau_0 = 1$ is shown here for different values of N. Since the standard deviation (listed in the legend as σ) only falls off as $\tau_0/\sqrt{N}$, it can take a large number of samples to be confident in the estimate of τ. If only a few samples are available, it may be unwise to "over-partition", since an inaccurate estimate will land an object in the wrong bin and it will thus be sampled incorrectly.

or t, there is no problem with carving up the space very finely since most bin placements will be accurate.

For search engines, the variance of the estimators for τ and t should translate into a more fine-grained partition (larger NV) of the mean time between access, and a comparatively coarse partition (smaller NT) of the mean time between changes. For both axes the density of partition lines can be much larger near the origin, where more samples will be available to make estimates for bin placement.

4.3.4 Interleaving observations

In this subsection we address two issues. First, how can observations be successfully interleaved in the presence of variable latency? Second, can an observation system attain near periodic time between observations? Towards answering these questions, we propose a two-tiered architecture in which the observation bins described in the optimization sections are logically distinct from the observation process. The key idea is that a one-to-one mapping between partitions and observation queues will not permit periodic observations, but a two-tiered system will.

For the discrete case, we have seen that the precise interleaving of many observations makes solutions complex, perhaps unnecessarily so. This has been partially alleviated by partitioning the object space and creating a fixed ordering of those observations within a bin. This is augmented by a randomized interleaving of the observations taken from those bins.

A real-world web monitoring system generally has fairly consistent total input bandwidth, but highly variable latency in how fast a single observation can be made. To permit an observation system to process at a fixed total (average) rate, as hypothesized in our discussion on optimization, it is necessary to address the variation in download rate. This is especially important if we claim that a periodic observation schedule is possible, as is assumed in much of our development of theory. Clearly, when observation time is exponentially distributed, observing one object after another in sequence will behave much like (4.30) and will not be periodic, but more Gaussian.

Intuitively, a downloading system must adapt such that when links are clogged and latency is high, more downloads are run in parallel. Likewise, when latency is low, fewer downloads are run in parallel. In this way, overhead is kept at a minimum while maintaining a more constant throughput. The schematic for such a system is shown in Figure 4.14. The key to understanding this system is the same as that used earlier in (4.18). Namely, the sum of the rates of the downloaders (each marked $DL\#n$ in Figure 4.14) must be greater than or equal to the aggregate request rate. Equality of the rates causes problems in real observation systems, since this will cause the queue length for any downloader to exceed any finite value. Rather than explicitly measuring the processing rate and determining the number of downloaders from the observed rate, we propose an adaptive scheme that routes observations to downloaders. This also allows the system to ensure that the processing rate is kept higher than the download rate. In a computerized setting, think of each downloader as a single process or thread, although not necessarily resident on the same machine. In fact, the downloaders might not even on the same local network. This is the basis for a whole branch of future work, in which download sites are selected for their proximity on the network to pages of interest. For either the single-site case or the distributed case, the multiplexer unit in the center of the figure routes observations to downloaders such that:

- If all downloaders $DL\#n$ are not finished with their previously allocated download and processing task, the multiplexer will create a new downloader and route an incoming observation to it.
- The multiplexer will route an incoming observation to a downloader that is done with its previous task.
- When a downloader has been idle for some predetermined waiting time, it can be taken off-line. Strictly speaking, this is not necessary–the system would function if all downloaders were up and running whether idle or not. Taking idle ones off-line will reduce overhead during the times when latency is smaller.

In this way, downloaders are created and destroyed in response to variable download time under a fixed request load. So long as the overhead of managing many concurrent downloads is tolerable, this method will allow a fixed processing throughput. We emphasize that this is not the only means for achieving near-periodic observations and tolerating variable latency, other systems could address

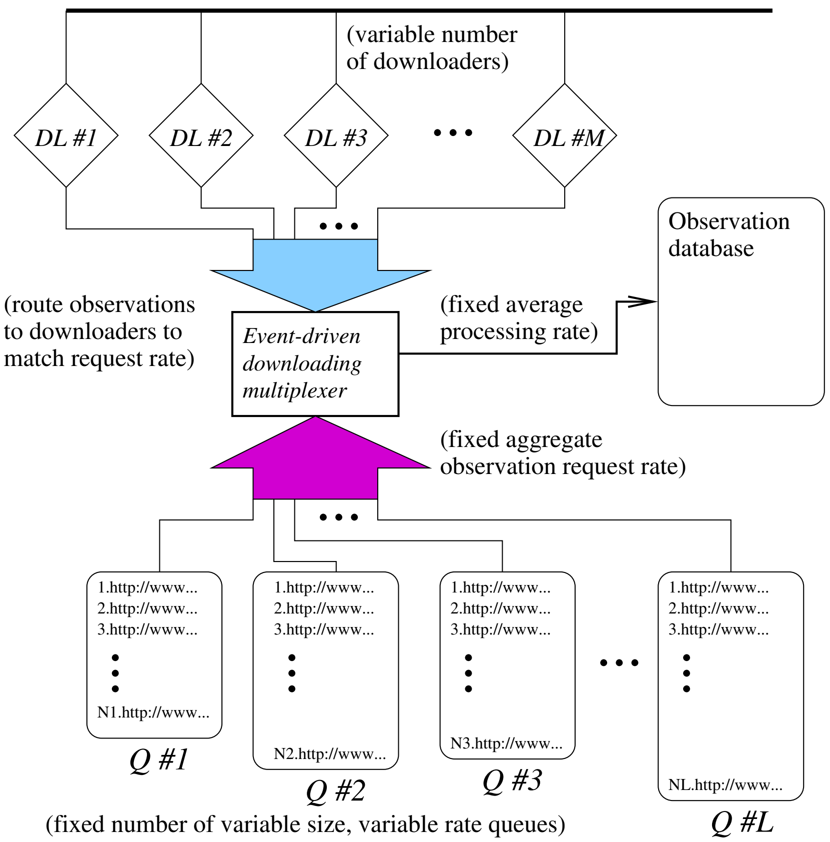


Figure 4.14: **Multiplexing fixed-period observation requests** To operate an observation module with a fixed number of variable-speed queues having fixed bandwidth and variable latency, a system like the one above can be employed. Download or observation modules, marked $DL\ \#n$ above, are created and destroyed as necessary in response to observation requests. A central control unit assigns observation requests to observers as they become available. These requests are pulled in from observation queues, marked $Q\ \#m$. If all observers are busy when a request is issued, another observer is started. Likewise, when there are idle observers that do not receive a request within a short time span after completing an observation, they are taken off-line. Think of each downloader as a process or thread, not necessarily all resident on a single machine. By starting downloads in response to requests, the processing rate is kept just above the request rate such that average throughput to the database is maintained. We emphasize that this strategy for maintaining a fixed observation period will eventually be made impossible by the overhead of managing so many download processes.

these issues as well.

4.4 Summary of optimizing observation

This chapter has dealt with a wide variety of topics. We began with discrete-time methods for optimizing the total number of expected objects out of date in an index. This metric could also be viewed as a probability that a randomly selected object was up to date. Applying the concept of (α, β)-currency, we then assumed memoryless page changes and sought a different kind of solution. Namely, we devised a means for finding a partition of the monitored object space and a corresponding rate allocation that optimized the probability α that an index was current to within β time units. Our examples demonstrate that such an optimization is much more helpful to the observer that is short on available bandwidth, as opposed to the one having an ample supply. As expected, the performance increase always came at the expense of less frequent monitoring of fast-changing but unpopular objects.

Chapter 5

Conclusion and future work

This thesis has focused on many aspects of the problem of monitoring a set of changing information sources. To begin, we developed discrete and continuous time models for finding the probability of a change to an information source, based upon the time since that source last changed. We then introduced the idea of (α, β)-currency to allow for a probabilistic and temporal relaxation of the meaning of "up-to-date". In addition, we presented a variety of information on the frequency and nature of changes to web pages, estimating that the expected mean time between changes for a web page within our collection is 138 days. The associated distribution of these change rates was then applied to the problem of a search engine monitoring those web pages. We presented discrete and continuous time algorithms for scheduling observations. Even from our simplified models, it is clear that search engines can increase their probability of returning 1-day current information by between 5 and 10%, and to a much greater extent when bandwidth is limited. Furthermore, the methods we applied are not necessarily limited to search engines; other application domains can implement these routines as well.

While much has been studied, it is easy to be overwhelmed by the number of possibilities considered but left undone. As might be expected, each subproblem has turned up many avenues for future work, which we categorize into three areas. First, our theory should be expanded. Second, as we apply this theory to the search engine's problem of observing the web, we find that there are many subtleties within that domain. We expect that these same subtleties will also arise in the third area, the application and further development of the theory for use in other domains.

5.1 Additional theory needed

Our method for determining (α, β)-currency rests upon the assumption that we can determine a rate of change and an importance measure for any object under observation. It may be that determining these is much more difficult than actually using that information. As mentioned in Chapter 4, there are errors in this determination as in any estimation process. It is possible to account for this inaccuracy, but we have not yet done so. A second theoretical issue, one which is quite easy to appreciate, is that we have assumed independence of change probability between monitored objects. Many environments will definitely have such correlations, whether this means that all web pages within a site change at or near the same time, or that a change in one stock's value will preclude a similar change in another's. A proper accounting for such correlations will dramatically increase the complexity of the observation problem, but the benefits may scale as well.

With the additional complexity, it is tempting to ask whether distributed processing and control can be of use. For observable objects that can self-monitor (having some programmability or built-in capability), there may be a means for notifying a central index of relevant changes rather than waiting to be observed. In many cases, objects "know" when they need to be observed again by a central index that is kept current. Since interaction effects are limited, a central server's workload is greatly reduced by giving objects the task of requesting their own re-observation. This is very much a push vs. pull technology contrast; a formal look into the relative efficiency of and methodology behind the two methods would make an interesting study.

5.2 Web observation and change assessment

Objects on the web are relatively near the goal of enabling remote observation. Each web server's filesystem, for example, already monitors each page for changes at the byte level; servers differ in the degree to which they are willing to divulge this information. The diverse efforts towards making caches more effective also permit some of the re-indexing effort to rest with web site administrators. Short of having remote processing power, perhaps a few extra cache control fields within HTTP is all the sophistication that can be expected.

Remote observation with a sensitivity to change metrics like the term vector model described in Chapter 3 might be implemented by mobile agents [Gra97b]. Like a waiter filling water glasses in a restaurant, it may be more effective for a "mobile" observer to examine many pages at a site while in their proximity, rather than waiting for them to change and then viewing them at higher cost from a more distant location. Just as the waiter buys time before having to revisit the table, an observation agent buys time before having to make more observations of a set of monitored objects at a server. Provided that web servers might allow their processor to be shared by an indexing agent (or agents), this might provide a means of encapsulating distributed processing and control. Obviously such a system could have much greater efficiency than a centralized system, and web server administrators could track or exclude the agents much more easily than if they use the same path as an ordinary user. A simple agent server might even be built in to the web server. Obvious security issues aside, these possibilities could solidify the oft-abused system of adding URLs to be indexed by search engines, which some sites exploit as a means to self-promote.

In the case of centralized change modeling and observation control, our continuous time models now operate on the assumption that web pages change according to exponential lifetime distributions. An analysis of how good (or bad) an assumption this may be would greatly contribute to a more definite understanding of the potential impact of this kind of scheduling for the web. Other models of change probability such as periodic (or near periodic) models would be useful additions to the theory of (α, β)-currency. We have demonstrated that there is at least some periodic component to change probability, as the probability of any given web page changing will rise and fall over the course of a day and on a weekly time scale as well. Generating such models will certainly require more complex mathematics but this is often the price of more useful modeling.

It is also not known whether a desired processing throughput can be maintained even when there is wide variation in observation time between documents (slow servers are intuitively less attractive for observation than fast servers) and over time. A further challenge is the probable correlation between how popular a resource is and how quickly it can be observed, so that the two-dimensional view of resources (popularity and change rate) must be expanded to include the added dimension of download time. There is a wide range of possibilities for maintaining processing throughput even in the context of highly variable latency. It is not clear how big an obstacle the control overhead will be.

Time between downloads is not necessarily periodic, as was assumed in the development of relations for finding β-currency with respect to a collection of memoryless sources. More realistic inter-sample time distributions are needed in order to estimate observer performance under these circumstances. Unlike the two-tiered architecture we proposed in Chapter 4, some observation systems that cycle through a large set of monitored objects, not starting a new observation until the previous one has finished, have near-Gaussian distributed time between observations (if the individual observations are independent random variables).

Experience with running the optimization routine seems to indicate it could be made more efficient. The relative sloppiness of the simulated annealing approach was chosen intentionally, trading slower convergence for a broader search of the solution space. Improvements could certainly be made in the manner by which the grid lines are placed, or even in whether grid lines are used at all—the low-bandwidth examples showed that the best bin choices would be irregularly shaped polygons that followed the edge beyond which objects were ignored. That entire region could probably be made into one oddly shaped bin. A method that permitted odd bin shapes would be helpful.

Additional modeling is also needed to determine user change tolerance in terms of a joint distribution of grace period times β over change rate and access rate. At most, one might speculate that users have an expectation of currency that scales along with the change rate of the topic of a web page or other object. It is not known how this grace period scales with change rate, or why.

Regardless of the statistics of change, actually detecting which changes are important (and which are not) presents a wide variety of challenges. A change to a web page is best defined not just by alteration in content or style alone, but by changes in content defined within the context of users' interest. This forces an attempt to divide page content into "content-related" and "non-content-related" subspaces. The ideas presented in Chapter 3 for such a division, regarding use of query keywords as a means of characterizing the user interest space, suggest just one possibility. Accurate projections of change onto user needs will probably only become more important as the content of web pages becomes more dynamic. It is quite likely that in the future no two documents obtained from the same URL will be entirely "alike." That is, content will be customized to such a degree that many documents will not remain the same for more than one viewing. This is already true for most of the popular sites on the web, epitomized by search engine and portal sites such as Excite, AltaVista, Yahoo, Go.com, and others.

Indeed, how does an index (or any list of links) deteriorate over time if not maintained? "Link rot", as it is called, could be formally studied using our data; prediction of which links are no longer usable could be used by search engines such that links that are likely to be non-functional are not followed. We have a year's worth of information on how the words in web pages change; the time series of observations and their difference from the first in the sequence should give an idea about how well a single index entry approximates those that follow. This would be useful in estimating the "damage" done to an index entry by an unobserved change, especially with respect to user needs.

5.3 Studying observation in other contexts

Some application domains will be much simpler than the web, having many objects of equal value and an easily determined change rate. One example of this would be a large signal processing system that pulls in sensor data on a variety of channels through a multiplexer. Other domains will have an easily determined access distribution and a more complicated rate of change determination. For example, how often should computers on a network be inspected for intrusion or compromise by system crackers? The many computers in an organization may have a very well-defined importance, but guessing a probability of attack is somewhat harder. The web and mail servers are clearly of great importance, but what is the probability of attack there? The operating system is probably more up-to-date, but an attacker might stand to gain more by successfully breaking in. Other computers on the network may be of very low-importance but much more open to attack (such as obsolete, all-but-decommissioned machines that just sit idle). The same principles we have developed for the web may apply there.

The theory we have developed can be applied to domains other than computerized observation. Just as an automated observer can be optimized for the detection of important changes, so can human observers. This may seem far-fetched, but one can imagine designing an instrument panel in an aircraft, or a display of vital signs for a surgeon, in which central visual placement is given to items of greater importance or having greater change rate. In this case we are not modifying the observation so much as we are placing things within a pre-existing observation strategy. The human observer already "knows" how to observe, it is simply a matter of placing an observable in the right place. There is a fascinating interplay between planning for easier observation and gaining an understanding of the filters that surround human sensory input. The human application domain may be the most challenging, but even a human observer can be measured for determining his or her (α, β)-currency with respect to an instrument.

Whether inspections are automated or done by humans, a determination must be made as to how often this should be done. Clearly, any such determination regardless of environment rests upon the same basic methodology as does our optimization of web observation: characterize the population by access rate or importance and change rate (even if on a time-varying basis), and then observe rapidly enough to catch the important changes. And regardless of the strategy employed, the measure of (α, β)-currency provides a meaningful measure of how up to date an observer is with respect to a set of changing information sources.

Appendix: Power-law distributions on the web

Zipf's law [Zip49], or the observation that rank-ordered lists of words tend to follow power-law distributions (a line on log-log scale plots), occurs frequently in our data on web pages. This is not the only part of the web characterized by power law scaling. This relationship has appeared in traffic analysis of page popularity within single domains [Nie97b], rank ordered external page references [Nie97a], and many other areas detailed in two summaries, [Pit98] and [Li00].

For a list of words ordered by their frequency within a corpus, Zipf's law states that

$$f \propto r^{-p} \tag{1}$$

where f is frequency, r is the rank within the list, and $p > 0$. For Zipf's law it is generally supposed that $p \approx 1$; some debate surrounds just how "magical" this constant is. On a log-log plot, the slope of (1) is just the exponent $-p$, as

$$\log f = \log k - p \log r. \tag{2}$$

As might be expected, our data shows that words used within web pages as well as words used by users running queries on search engines (for the Informant service) tend to follow these distributions. We performed the same analysis for the links within the documents in the same corpus, as well as the hostnames appearing within these links. It was a bit surprising to see power law behavior in the list of hostnames. The list of hyperlinks did not show a nice linear trend, though this may be due to the "small" sample size–with nearly a billion web pages, two million links is perhaps too small a sample to see this behavior. Graphs showing all these relations appear in Figures 1 through 3.

Inasmuch as hostnames and links represent something of a language unto themselves, one might expect that they exhibit the same sort of properties found in lists of words. Unlike lists of words, though, hostnames and links are references to large bodies of possibly dynamic text, images, and formatting, rather than to small, simple concepts. As observed by Mandelbrot [Man82], generalized power-law distributions (of which Zipf's is a special case) can be explained as a side effect of efficient communication, or more importantly, as a fractal phenomenon. Perhaps the deviations from linearity in the `HREF` data are indications of inefficient communication rather than an effect of sample size. This is very much open to discussion; a study of the web as a quickly-evolving communication mechanism would be quite fascinating.

We also see power law distributions as a result of slightly different derivation. When we examine a collection of items and frequencies, it is possible to estimate a probability density of frequencies. This too appears to follow a power-law scaling, analogous to Figure 3.15. Examples of this are shown in Figures 4 and 5. These differ from the first set of figures in that the *lowest* probability events (such as the event of having the highest frequency within a collection of words) are to the far right on the abscissa, while in relations like (1), the rank corresponding to this low-probability event is on the left end of the abscissa (having the highest rank). As above, much more could be done with this data: why do these plots have the slope that they do? What is their relation to the rank-ordered lists, and why? It will be interesting to see how the study of the web's self-similarity develops in the coming years as the web emerges from its youth.

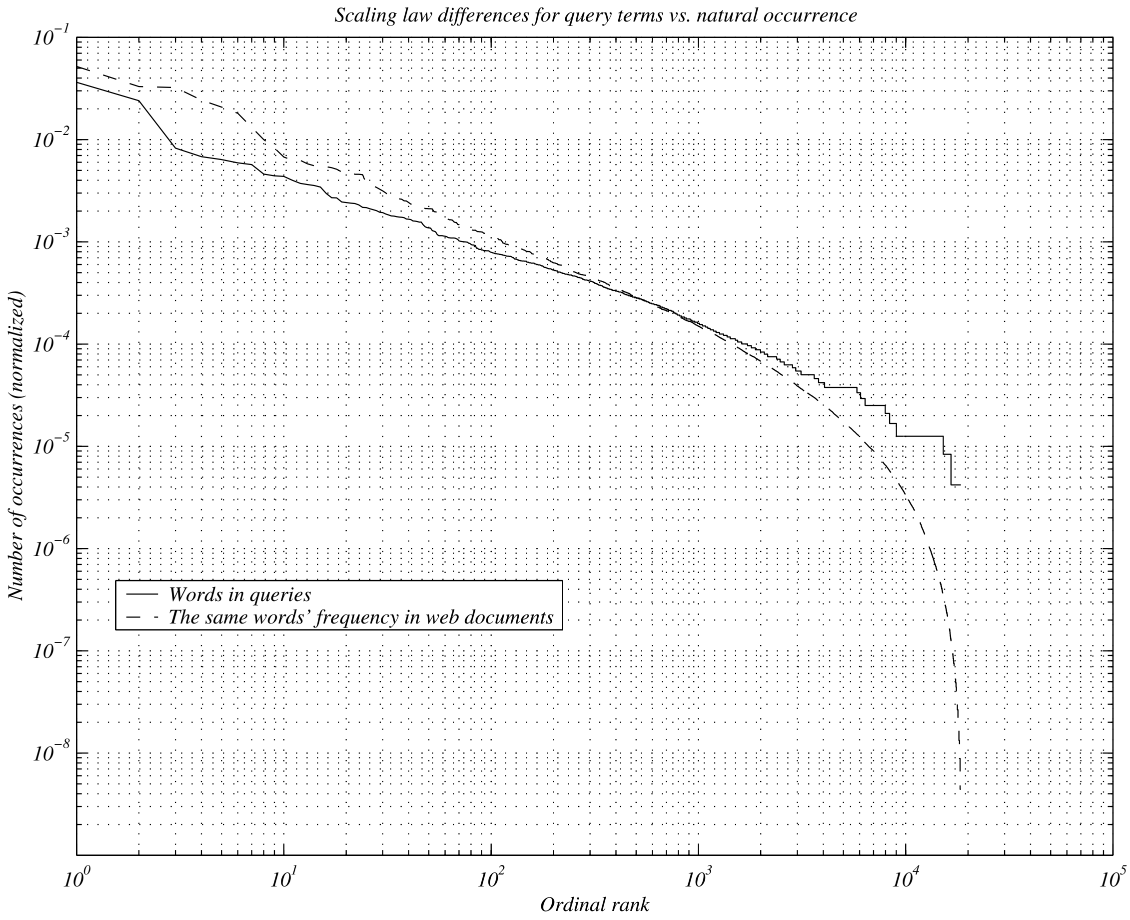


Figure 1: **Rank-ordered document words and query terms** ($N = 18,458$) As explained in (1), we plot the rank-ordered frequency of words from two different sets. First, we show the plot of the rank-ordered frequencies for terms appearing in user queries. Second, we plot the rank-ordered frequency of the same words within web documents. We emphasize that the ordering of the two lists is different, since each list is sorted by frequency. Thus the same word will correspond to a different horizontal position (rank) within each list. Notice that both plots are near linear for the higher ranks and have a similar slope, but this slope is between -0.7 and -0.9, not -1 as suggested by Zipf. This suggests that query vocabularies follow different rules than standard language. The central portion of each, where the slope estimates are more reliable, seem to have different slopes.

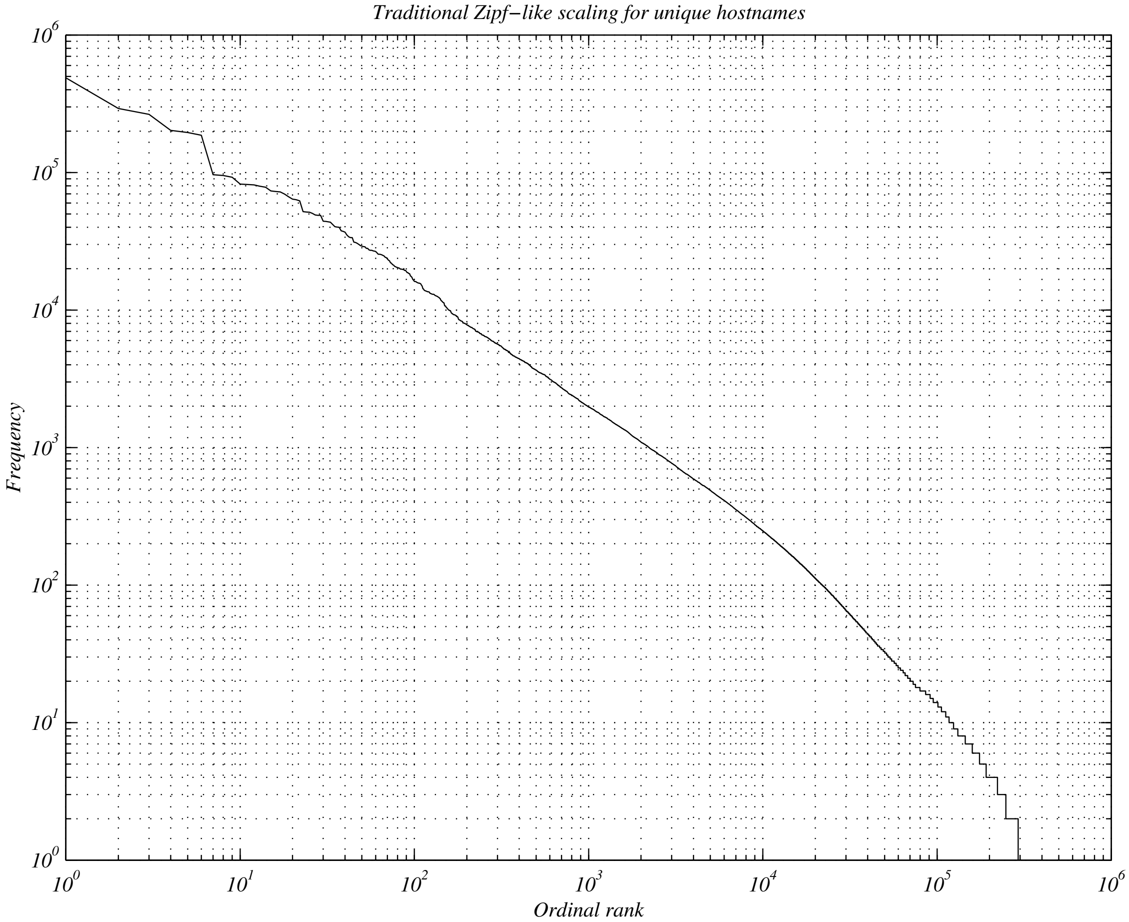


Figure 2: **Rank-ordered unique web hosts** As with the rank-ordered query terms, the linear behavior is strongest in the middle ranks of this curve. The slope is approximately -0.85. We counted the $360,398$ unique hostnames occurring within $3,314,876$ hyperlinks.

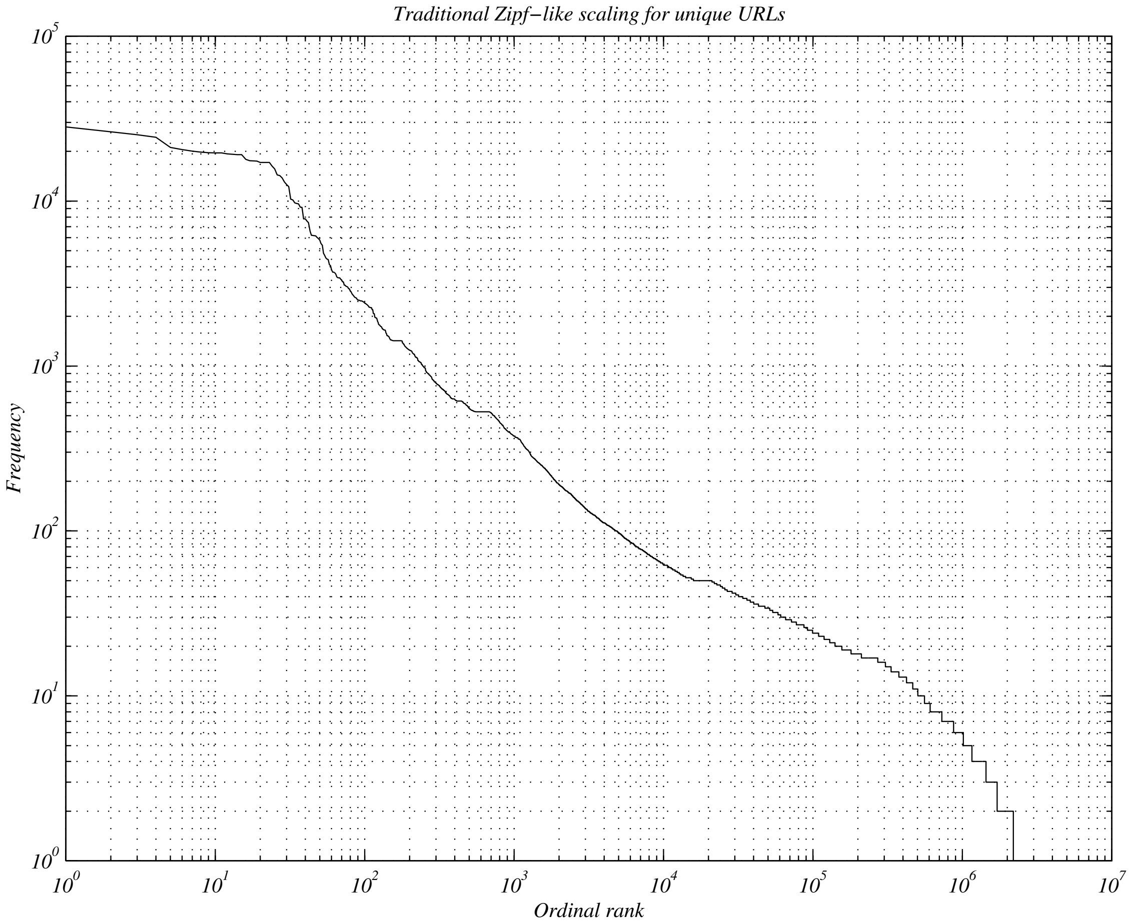


Figure 3: **Rank-ordered unique `HREF`s (hyperlinks)** This is not an especially linear fit, even by very loose standards. This is either due to too small a sample (of perhaps one billion possible `HREF`s we have sampled 3.3 million), or a governing law other than (1). The slope of a line through the data is roughly -0.69.

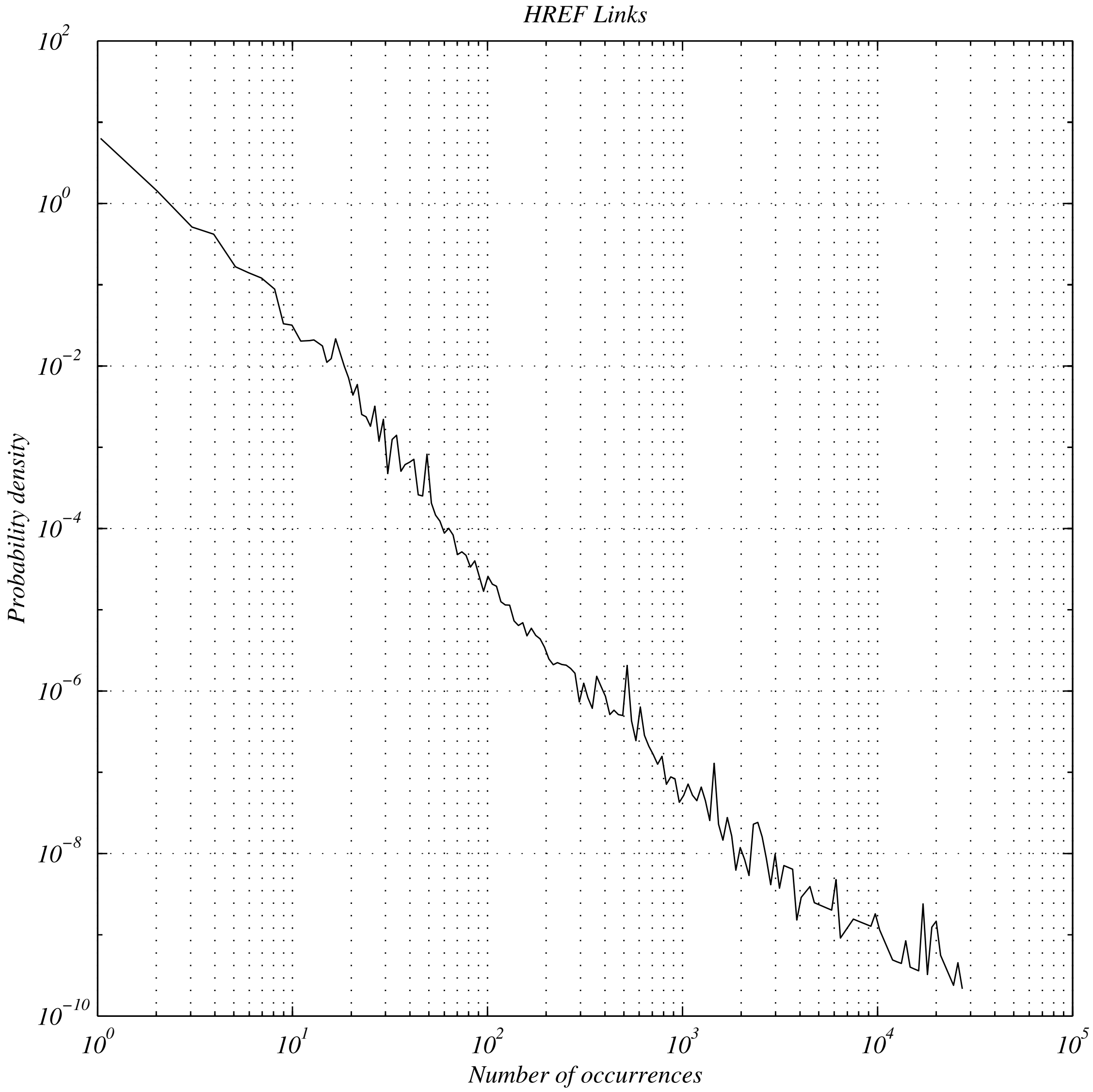


Figure 4: **PDF of frequency of unique `HREF`s (hyperlinks)** This figure shows much more linear behavior than Figure 3; as discussed in the text this is a somewhat different statistic. In this plot (and in Figure 5), the x-coordinate is the number of occurrences of a hyperlink, and the y-coordinate is the estimated probability density for each such count. Obviously this distribution scales with the sample size; as before this involves 3.2 million links. The slope of the line is approximately -2.6.

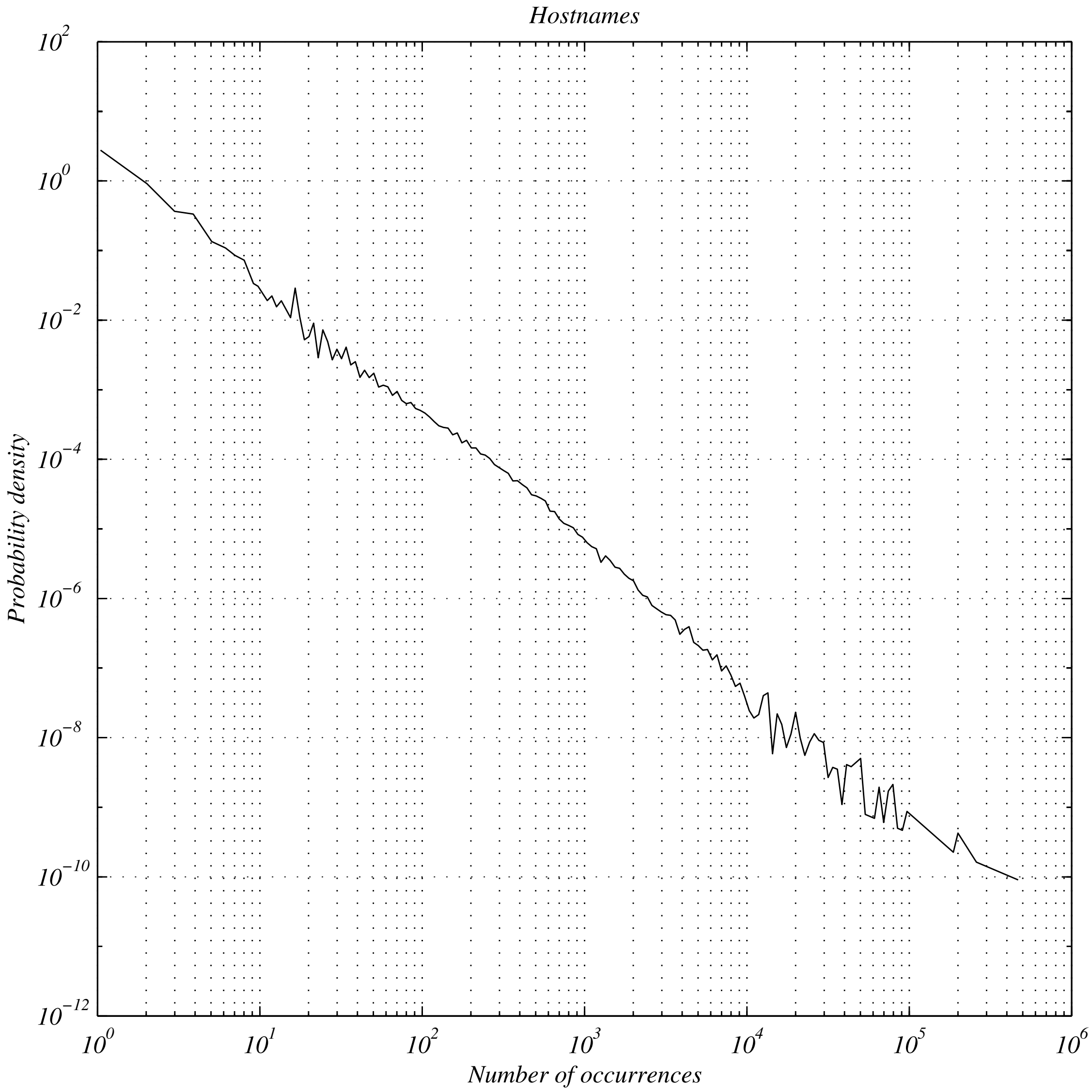


Figure 5: **PDF of frequency of unique web hosts** The hostname data shows an extremely good linear fit for the PDF of frequency. The slope, which is approximately -1.9, is very close to the slope for the same plot for unique words (which is plotted in the main text as Figure 3.15).

Bibliography

[BDH$^+$94] C. Mic Bowman, Peter B. Danzig, Darren R. Hardy, Udi Manber, and Michael F. Schwartz. The Harvest information discovery and access system. In *Proceedings of the Second International World Wide Web Conference*, pages 763–771, October 1994. Available from ftp://ftp.cs.colorado.edu/pub/cs/techreports/schwartz/Harvest.Conf.ps.Z.

[Ber87] D. M. Bertsekas. *Dynamic Programming.* Prentice Hall, 1987.

[Ber98] D. M. Bertsekas. *Network Optimization: Continuous and Discrete Models.* Athena Scientific, 1998.

[BKM$^+$00] 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 Ninth International World Wide Web Conference*, Amsterdam, The Netherlands, May 2000. Available from http://www9.org/.

[BP98] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the Seventh Annual World Wide Web Conference*, April 1998. Available from http://www7.scu.edu.au/programme/fullpapers/1921/com1921.htm.

[Bre98] Brian Brewington. Ph.D thesis proposal: Optimal observation with WWW applications. http://agent.cs.dartmouth.edu/papers/brewington:observe.ps.Z, 1998.

[Cat92] Vincent Cate. Alex – a global filesystem. In *Proceedings of the USENIX File Systems Workshop*, pages 1–11, May 1992.

[CBB+96] George V. Cybenko, Aditya Bhasin, Brian Brewington, Robert Gray, Katsuhiro Moizumi, and Kartik Raghavan. The Shannon Machine: A system for networked communication and computation. In *Proceedings of the Eighth Yale Workshop on Adaptive and Learning Systems*, Yale University, New Haven, Conn., 1996. Center for Systems Science, Dunham Laboratory. Available by request from http://www.dartmouth.edu/~gvc/.

[CDN+96] Anawat Chankhunthod, Peter Danzig, Chuck Neerdaels, Michael Schwartz, and Kert Worrell. A hierarchical internet object cache. In *Proceedings of the USENIX 1996 Annual Technical Conference*, January 1996. Available from http://www.usenix.org.

[CLW97] E. G. Coffman, Z. Liu, and R. R. Weber. Optimal robot scheduling for web search engines. *Journal of Scheduling*, 1997. Available at http://www.inria.fr/mistral/personnel/Zhen.Liu/Papers/.

[Coo72] Robert B. Cooper. *Introduction to Queueing Theory.* The Macmillan Company, New York, 1972.

[Cou93] Leon W. Couch. *Digital and analog communication systems.* Macmillan, New York, fourth edition, 1993.

[CvdBD99] Soumen Chakrabarti, Martin van den Berg, and Byron Dom. Focused crawling: A new approach to topic-specific web resource discovery. In *Proceedings of The Eighth International World Wide Web Conference*, Toronto, Canada, May 1999. Available from http://www8.org/w8-papers/5a-search-query/crawling/.

[DB96] Fred Douglis and Thomas Ball. Tracking and viewing changes on the web. In *USENIX Technical Conference*, January 1996. Available from http://www.research.att.com/~douglis/aide/.

[DFKM97] Fred Douglis, Anja Feldmann, Balachander Krishnamurthy, and Jeffrey Mogul. Rate of change and other metrics: A live study of the World Wide Web. In *Proceedings of* *the USENIX Symposium on Internetworking Technologies and Systems*, December 1997. Available from http://www.research.att.com/~anja/feldmann/papers.html.

[EW61] H. P. Edmonson and R. E. Wyllys. Automatic abstracting and indexing survey and recommendations. *Communications of the ACM*, 4:226–234, 1961.

[Fel68] William Feller. *An Introduction to Probability Theory and Its Applications*, volume 1. Wiley, 3rd edition, 1968.

[Fel71] William Feller. *An Introduction to Probability Theory and Its Applications*, volume 2. Wiley, 2nd edition, 1971.

[Goo99] Google. Google ranks first overall in third-party survey of search and portal users, December 1999. http://www.google.com/pressrel/pressrelease9.html.

[Gra97a] Matthew Gray. Internet growth summary. http://www.mit.edu/people/mkgray/net/internet-growth-raw-data.html, 1997.

[Gra97b] Robert Gray. *Agent Tcl: A flexible and secure mobile-agent system*. PhD thesis, Dept. of Computer Science, Dartmouth College, June 1997. Available as Dartmouth Computer Science Technical Report TR98-327.

[Hay94] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Macmillan College Publishing Company, 1994.

[HN99] Allan Heydon and Marc A. Najork. A scalable, extensible web crawler. *World Wide Web*, 2(4):219–229, December 1999. Available from http://www.research.digital.com/SRC/mercator/papers/www/paper.html.

[How60] R. Howard. *Dynamic Programming and Markov Processes*. Jointly published by MIT Technology Press and Wiley, 1960.

[HS98] J. P. Hardwick and Q. F. Stout. Flexible algorithms for creating and analyzing adaptive sampling procedures, 1998. To appear. Also see http://www.eecs.umich.edu/~qstout/abs/Seattle97.html.

[Inf95] Informant, 1995. http://informant.dartmouth.edu.

[Inf99] Infoseek. *Ultraseek Server 3.0 Administrator's Guide*, 1999. available from http://software.infoseek.com.

[ISC99] ISC, 1999. Internet Software Consortium; http://www.isc.org/dsview.cgi? ... domain-survey/index.html.

[JCP98] Hector Garcia-Molina Junghoo Cho and Lawrence Page. Efficient crawling through URL ordering. In *Proceedings of the Seventh Annual World Wide Web Conference*, April 1998. Available from http://www7.scu.edu.au/programme/fullpapers/1919/com1919.htm.

[Kah97] Brewster Kahle. Preserving the internet. *Scientific American*, March 1997. Available from http://www.sciam.com/0397issue/0397kahle.html.

[LG98] S. Lawrence and C. L. Giles. Searching the World Wide Web. *Science*, 28:98–100, April 1998. Available by request at http://www.neci.nj.nec.com/homepages/lawrence/.

[LG99] S. Lawrence and C. L. Giles. Accessibility of information on the web. *Nature*, 400:107–9, July 1999.

[Li00] W. Li. References on Zipf's law, 2000. Available from http://linkage.rockefeller.edu/wli/zipf/.

[Lue84] D. G. Luenberger. *Linear and Nonlinear Programming*. Addison Wesley, 2nd edition, 1984.

[Man82] Benoit B. Mandelbrot. *The Fractal Geometry of Nature*. WH Freeman, 1982.

[MDFK97] J. Mogul, F. Douglis, A. Feldmann, and B. Krishnamurthy. Potential benefits of delta-encoding and data compression for HTTP. In *Proceedings of ACM SIGCOMM'97 Conference*, pages 181–194, September 1997. Available from http://www.research.digital.com/wrl/techreports/abstracts/97.4.html.

[Moo99] General Thomas S. Moorman. The ultimate high ground: Space and our national security, June 1999. Notes from 1998 Wernher von Braun lecture at the National Air and Space Museum; http://www.nasm.edu/nasm/dsh/WERNHER/braun98.htm.

[MR94] Douglas C. Montgomery and George C. Runger. *Applied Statistics and Probability for Engineers.* John Wiley and Sons, Inc., 1994.

[Nic97] Scott Nicholson. Indexing and abstracting on the World Wide Web: An examination of six web databases. *Information Technology and Libraries*, June 1997. Available from http://www.askscott.com/iapaper.html.

[Nie97a] Jakob Nielsen. Traffic from referring sites, April 1997. Available from http://www.useit.com/alertbox/zipf_referring.html.

[Nie97b] Jakob Nielsen. Zipf curves and website popularity, April 1997. Available from http://www.useit.com/alertbox/zipf.html.

[Not99] Mark Nottingham. Caching tutorial for web authors and webmasters, June 1999. Web Developers Virtual Library; http://wdvl.com/Internet/Cache/.

[Pap84] Athanasios Papoulis. *Probability, Random Variables and Stochastic Processes.* McGraw-Hill, 2nd edition, 1984.

[PFTV92] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in C.* Cambridge Press, second edition, 1992.

[Pit98] James E. Pitkow. Summary of WWW characterizations. In *Proceedings of The Secenth International World Wide Web Conference*, Brisbane, Australia, April 1998. Available from http://www7.scu.edu.au/programme/fullpapers/1877/com1877.htm.

[vR79] C. J. van Rijsbergen. *Information Retrieval.* Butterworths, London, second edition, 1979. http://www.dcs.gla.ac.uk/Keith/Chapter.2/Ch.2.html.

[WMB94] Ian Witten, Alistair Moffat, and Timothy Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images.* Van Nostrand Reinhold, 1994.

[Zip49] George Kingsley Zipf. *Human Behaviour and the Principle of Least Effort.* Addison-Wesley, Cambridge, Massachusetts, 1949.