Temporal-Informatics of the WWW

Eytan Adar

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

University of Washington

2009

Program Authorized to Offer Degree: Computer Science and Engineering

University of Washington
Graduate School

This is to certify that I have examined this copy of a doctoral dissertation by

Eytan Adar

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Chair of the Supervisory Committee:

_____

Daniel S. Weld

Reading Committee:

_____

Daniel S. Weld

_____

James A. Fogarty

_____

Susan T. Dumais

Date: _____

University of Washington

Abstract

Temporal-Informatics of the WWW

Eytan Adar

Chair of the Supervisory Committee:
Professor Daniel S. Weld
Computer Science and Engineering

The World Wide Web is generally experienced as a single snapshot: the *Now Web*. The Now Web perspective means that only very recent states of the Web are being observed through browsers or through the crawling and index structures of search engines. Such a view ignores the utility of historical data to system designers and end-users alike. This dissertation characterizes content and behavior within the *Dynamic Web*. This work further illustrates the power of Dynamic Web data by utilizing measures of *content change* and behavioral measures of *use* to identify likely targets of user revisitation on the page level. Additionally, two systems are offered for manipulating temporal Web data

- *DTWExplorer*, a tool for analyzing time-series representations of content and behavior to identify correlations in different types of temporal Web data.

- *Zoetrope*, an end-user tool for querying, aggregating, and visualizing historical Web data from the context of the Now Web.

This work capitalizes on the richness of the Dynamic Web and demonstrates how moving away from *temporally insensitive* models and tools can enhance existing applications (e.g., crawlers and search engines) and lead to novel applications and systems.

Table of Contents

List of Figures

Acknowledgments

There are quite a number of people that made this dissertation possible but at the top of the list is my advisor, Dan Weld. My research style can best be described as research ADD with many projects going on at the same time and ranging from a little crazy to completely insane. Dan had the unenviable task of trying to manage this and guide me towards finishing. I learned a tremendous deal from our research collaborations, brainstorming sessions, and random chats. I feel extremely lucky and grateful that I worked with Dan.

I would also like to thank the rest of my committee: Susan Dumais, James Fogarty, Efthi Efthimiadis, and Magda Balazinska. In addition to being on my committee, Susan was truly great mentor. I credit many of the best results I got over the past 4 years to the time I spent at Microsoft with Susan and Jaime Teevan. Susan and Jaime both had a rare blend of interests and skills that complemented my own in many ways and made for amazing collaborations.

I would also like to thank my collaborators and co-authors, without whom many of the projects and studies described in this dissertation would not have been what they are today (if they would have been at all). In addition to Dan, Susan, and Jaime, I would like to especially thank Mira Dontcheva, James Fogarty, Jon Elsas, Brian Bershad, and Steve Gribble. Some of the data and infrastructure used in this dissertation were produced with help from Dan Liebling, Ronnie Chaiken, Bill Ramsey, and Dennis Fetterly.

I would like to thank my family. They are my biggest cheerleaders and supporters. Sara, my parents, Udi, and more recently, Maya and Avi, have kept me motivated, grounded and

perhaps most importantly, loved.

In a way I was lucky to have worked as a researcher for a number of years before going back to graduate school. The people who mentored and advised me early in my research career should all receive thanks (or blame) for the researcher I've become: Jerry Saltzer, David Karger, Lynn Andrea Stein, and Bernardo Huberman.

After working for so many years, the decision of where I should go to graduate school was a huge decision for me but I can unequivocally say I made the right one. Although, one generally complains about how the PhD process seems endless, the department and my colleagues and friends made it so that I never noticed.

My time at the University of Washington gave me an opportunity to work with many amazing faculty, staff, and students. In particular, I'd like to express my appreciation to Chris Ré, Michael Skinner, Miryung Kim, Raphael Hoffman, Fei Wu, Kayur Patel, Saleema Amershi, Mike Cafarella, Oren Etzioni, Yoshi Kohno, YongChul Kwon, Julie Letchner, Lindsay Michimoto, David Notkin, and Atri Rudra.

A number of organizations helped fund my research through fellowships including the University of Washington, the NSF and ARCS. Additionally, I was extremely lucky to be selected to receive data and support from Microsoft through the Accelerating Search in Academic Research program.

Dedication

to my wife, Sara

Chapter 1

Introduction

The World Wide Web, as understood by system designers and subsequently experienced by most people, is as a single snapshot in time–the *Now Web*. This view is largely due to a history of limited resources (e.g., the *time* to gather the data and the *storage* and *processor* power needed to analyze it once it was collected), the effect of which has been a lack of research on the nature of the Dynamic Web. Such limitations have slowed innovations in existing systems and stunted the creation of new technologies that leverage the *Dynamic Web (or the Temporal Web)* and address the needs of end-users and system designers. By failing to capitalize on the dynamics of information and user behavior on the Web, developers and designers are missing an opportunity to provide better services. For example, search engines are limiting their ability to detect and keep up with important changing content; browser designers still treat all bookmarks and pages the same–making revisits inefficient and content ephemeral; and advertisers are missing out on cues that might indicate when users are most attentive to new content and advertisements. These are but a few examples of situations in which the understanding and use of Dynamic Web content and behavior can lead to improvements. However, such innovation is difficult because of the many gaps in the research in this area. This dissertation seeks to answer a number of prerequisite questions and to build models and systems that allow for the transition from the static to the dynamic.

This work is particularly timely due to the rising demand for trending analysis sys-

tems [70, 28], real-time search [154, 129], dynamic social media applications [109], and applications that extend beyond the snapshot to collect information from multiple states of the Web [132]. As designers and system builders come to recognize the value of Dynamic Web data, they are faced with the strange reality that while Dynamic Web data is *incomplete* and difficult to collect (i.e., it is hard or impossible to observe every state change on the Web), it is simultaneously so large that working with the collected data is nearly an overwhelming challenge.

While users, engineers, and designers have in many ways learned to deal with a single snapshot of the Web, they have also run into resource limits in keeping up with the changing Web [53] and making use of the collected historical data. These problems will only become more pronounced as the creation of new Web content accelerates. For example, large scale studies over the past dozen years have identified broad types of change (e.g., 35% of existing pages change over 11 weeks [59] and new content equal to 50% of the Web is created every year [126]). This rate necessitates a compromise in the amount of content collected and retained. Of course, in selecting this subset of the Web, "valuable" content would be more useful than a random subset. It is here that behavioral data becomes important as it can be used to detect important or interesting trends and detect high-value content changes.

The examples above seek to fix or help many Now Web applications. But what of providing direct access to the Dynamic Web? Unfortunately, even if Dynamic Web data can be collected it is still unclear what we can do with it. The lack of support for handling temporal Web data, in part due to limited resources and in part to the present architecture of Web services, means the current user experience is focused on that single instance of the Web visible through the browser window or a search engine's view of the Now (i.e., the last crawled snapshot). Such interfaces are *temporally insensitive*, limiting the questions that individuals can ask and answer, and at best making certain tasks more difficult (and at worst impossible). Even when they are able to access historical snapshots of the Web, through tools such as the *Internet Archive's Wayback Machine*, end-users do not currently have the means to manipulate or analyze this data easily (e.g., how does one answer the question: how has a particular book price varied over time?).

This dissertation provides novel models of the Dynamic Web, and implement new solu-

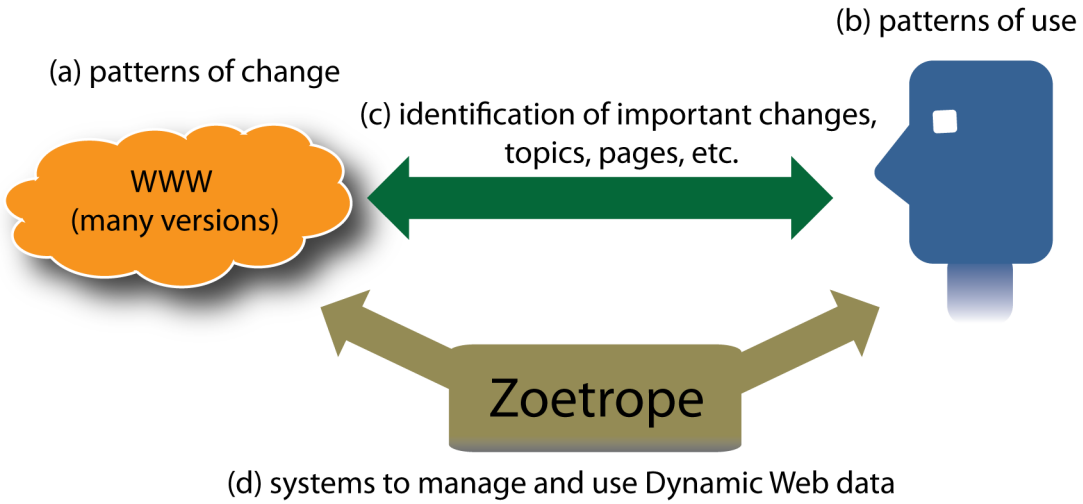


Figure 1.1: Components of the Dynamic Web research in this dissertation: (a) identification of properties and models of changing Web data, (b) patterns of use over time, (c) inference of interesting changes, topics, etc. and (d) better tools to support management and use of Dynamic Web data.

tions on it, combining an understanding of both content and the behaviors used to create and consume that content. By integrating and expanding notions of content change with actual behavioral patterns, this dissertation seeks to address the following high level questions:

1. What is the relationship between changing Web content and the ways this data is consumed?

2. How can temporal content produced by different communities, and in different mediums, be connected?

3. What systems can manage Dynamic Web content and provide new interaction techniques to consume this information?

Answering these high level questions requires a number of novel contributions in different aspects of the Dynamic Web, ranging from collecting massive datasets, designing new systems and algorithms to process this data at previously unattempted scales, building novel

models for both content and behavior, and creating new interaction techniques for end-user manipulation. Graphically, these can be represented as in Figure 1.1. First, identifying the relationship between content and consumption requires (a) new models for the evolution of the Web and (b) patterns of use such as revisitation (the way individuals return to the same page over and over) and refinding (repeated searches on the same topic over time). Taken in combination (c), content and behavior can be used to automatically identify which changing information is "interesting" and to provide anything from better crawling strategies to better bookmarking systems. Content and use can also be grouped and partitioned by different groups to better predict and improve the behavior and produced content in one group using the behavior and content of another. Finally, providing end-user technologies (d) to deal with the massive influx of data. That is, it is insufficient to simply save many copies of the Web. New tools must be created to let end-users actually work with this data.

1.1 State of the Art

Although Chapter 8 is a complete survey of related work, it is worth briefly acknowledging key research in the space and the limitations that are addressed in this dissertation.

While applications of the Dynamic Web are limited, understanding the growth and decay of the Web over time has been critical to a number of systems that require accurate snapshots of the Now Web (e.g., search engines). Work in this area has primarily focused on systems issues, such as crawl rates and indexing, and have ranged in size from 150 million pages over 11 weeks [59] (that identified that 65% of pages remained the same) to studies on smaller page sets over longer periods of time (e.g., 360 pages over 5 years in [103]). Although useful, none of the previous studies have utilized behavior for sampling pages– instead focusing on random samples of the Web. One might anticipate that pages that are actually used on a regular basis have a very different pattern of evolution and that evolution happens on time-scales faster than over single days (the fastest of the crawls).

In contrast to some of the large scale evolution studies that capture a significant portion of the Web, work on individual behavior over time (e.g., revisitation patterns) has been much more limited. Work in this area (e.g., [51, 94, 127]) has traditionally been limited

to a small user population willing to install a browser plugin or use a proxy that would track Web page access. While this work has been useful in identifying the prevalence of revisitation (50-80% of all clicks are to previously visited pages), very little could be said about revisitation patterns to *specific* Web pages. That is, the chance that two users in a small study population would independently revisit all but the most popular pages is extremely unlikely.

Without fine-grained content datasets for the visited Web and without the corresponding visitation patterns to those pages, it was previously impossible to identify the relationship between change and revisitation in any significant way (though some work suggestively correlated increasing revisitation rates to increased change rates [127]).

A similar issue of scale has impacted work on correlating different groups, topics and media over time. Although there has been growing recognition that such correlations are useful (e.g., Flu detection [69] or economic indicators [49] using search keywords, or book popularity using blog posts [76]), work in the area has focused on "point solutions" to specific problems. There is an opportunity then, to provide a more general infrastructure by which many different sources of dynamic data can be simultaneously correlated. Doing so requires mechanisms for converting content and behavioral data into time-series and providing better mechanisms for comparing and correlating time-series.

Finally, despite the tremendous amount of dynamic Web data that is constantly being created and deleted, very little has change since the advent of the Internet Archive's Wayback Machine (*www.archive.org*). Recognizing the fact that Web users are frequently interested in monitoring for changes (e.g., [94]), systems have been developed to compare two versions of pages to detect differences (e.g., [57, 153] among others). Others have created methods to track specific sub-pieces of pages [73]. However, comparing two versions or being able to access all versions are two extremes that do not satisfy a number of use cases where a user would like to *extract* temporal information from pages. Such cases require a more complex "query" language for asking and answering questions with data that exists over many different instances of a Web page[1].

[1]Though notably, some pages, including blogs and Twitter pages may contain a mix of historical and current information so the notion of "instance" is slightly more complex.

Querying Web pages has largely been the domain of End User Programming (EUP) environments (e.g., [111, 31]). Unfortunately, these systems greatly suffer from the Dynamic nature of the Web, as "programs" written for one version of a page will potentially break when the page changes. Furthermore, such systems are designed for working on single snapshots of the Web–usually the most current. Thus, while a program might be able to extract a specific stock price or weather statistics from the page, such programs are not applied to the historical versions of the page.

Dealing with these limitations, and answering the specific questions described above, has required a number of novel models, new studies, and new systems.

1.2 Facets of the Dynamic Web

The evolution of the Web is traditionally viewed from the perspectives of *structure* (new link formation and the appearance and disappearance of pages) and *content* (changes in content on existing pages). A third facet, human *behavior*, has frequently been ignored in the context of the Dynamic Web. Behavioral data corresponds to the time-varying behaviors of individuals and groups as they *produce* and *consume* Web data. While a number of studies have targeted structural and content changes on the Web, behavior has been a missing ingredient in this work–leading to gaps in understanding of drivers of Web change and our ability to leverage change data effectively. In the absence of behavioral information and direct support for tasks, a number of important signals that would help in identifying and utilize content of value are lost.

Although the structural facet of the Dynamic Web is only one of the important aspects of the Dynamic Web, it is here that most prior work has been applied almost exclusively. Instead, this dissertation concentrates on the time-varying *content* and the human *behavior* that has acted to produce or consume that content.

Dynamic Web content, or more precisely the *time-annotated* digital content that is published on the Web, can take many forms. The data itself might be anything from news stories, weather data, blog posts, traffic delay maps, tweets, etc. The time-annotation reflects the time when the data appeared, disappeared, or changed. Ideally, such a temporal-

annotation would also indicate the validity period or the exact time when something became true. For example, the *publication time* of a news story about Barack Obama's visit to California may not be the same as when he arrived in California and may not overlap the "valid" time when he actually was in California. The notion of time and validity will be touched upon later in this work. In addition to the perspective of time, there is also the perspective of action. Depending on this perspective, Web content might be treated from the point of view of the producer or from the view of the consumer. The particular population that is being studied (e.g., people who post on blogs versus people who visit CNN.com to read the latest news) will also define if the data is being produced or consumed (or both).

The idea of production and consumption can be extended beyond a simple relationship of person to artifact (i.e., the content). A second, potentially richer, view encapsulates the *act* of creating or consuming Web content. Inherently temporal data sets such as behavioral traces (e.g., search and click logs), observations, and interviews provide additional insights into the Dynamic Web. An understanding of the *production* of content and conversely the *consumption* of content are vital in creating robust models of the Dynamic Web, but such studies are rarely executed.

To summarize, there are two main temporal data streams (i.e., time-series) that are considered for Dynamic Web understanding: *behavior* and *content*. Behavior relates to the act of producing or consuming information. Behavioral logs always include the *when* (the timestamp), and optionally the *how* (e.g., through a browser, through a blog, etc.), and *who* (e.g., demographic information or unique ID). Depending on the perspective, streams in this class include *Content Production* (i.e., the activity of creating content over time) or *Content Consumption* streams (i.e., the activity of consuming that content over time). Content streams speak to the artifact (i.e., the content) and include the when but also the *what*. With luck, both behavior and content will be collected, but frequently inferences must be made to convert one to the other. This distinction will become more important in scenarios where only partial views of behavior or content are collected (e.g., determining the content of interest on a Web page based on visits, or predicting behavior based on the appearance of certain content). The study of behavior and content streams, independently and together, will allow for a rich exploration and manipulation of the Dynamic Web.

1.3 Putting it Together

This dissertation seeks to address a number of gaps in the understanding and manipulation of Dynamic Web data. Specifically, the work here describes a number of studies on properties of content and use over time as well as the relationship between the two. This is done both from the perspective of single pages and sites but also the topics that receive different levels of attention over time from different communities. Having established some patterns of change and use, this dissertation also offers Zoetrope, a system for manipulation and querying of this Dynamic, and frequently ephemeral, content.

1.3.1 The Production/Consumption Relationship

Despite the significant (independent) interest in the way the Web is changing and how people access it over time, there has been little work to connect the two. In large part, this has been due to the lack of real data identifying (1) fine-grained patterns of change over content that is actually visited, and (2) the corresponding visitation patterns of individuals accessing those pages. The consequence of this is that we are not aware of a) what are the patterns by which people (re)visit content, b) how fast this *visited* Web is changing, and c) what is the correlation between change and behavior.

The Changing Web

As described above, previous research on the Web's evolution tends to forsake the perspective of *use* for the *perspective of scale*: how to best crawl and index dynamic Web-sized collections and provide maximum coverage for live content. This is reasonable in designing algorithms, such as PageRank, that benefit from pages regardless of whether they are used or not. However, because such research concentrates on random samples of the Web, the results are content- and behavior- agnostic. Because of this fact, and because Web evolution studies collect data slowly (at most once a day), the derived models tend to misestimate the dynamics of the portion of the Web that is actually used. Not only does this make the synchronization hypothesis difficult to prove, but poor estimates can impact crawlers, search engines, and other applications that monitor the Web. To address this, the studies

described in this dissertation (see Chapter 3) utilize 55k pages that were selected based on different types of consumption patterns and are crawled hourly over a period of months, creating a unique dataset of the finest granularity crawls ever collected.

Previous work had alluded to the fact that the visited Web might change more frequently than a random sample. However, because a page collection based on use was never built, this could not be quantified or even verified conclusively. Whereas earlier studies indicated that 65% of the Web is unchanged after 11 weeks [59], the work presented here concludes that most pages on the visited Web change much more often. This result might be dismaying to the crawl engineer, who already feels overwhelmed by the amount of new content. However, by expanding on notions of change from a simple binary decision (did it change or not) this dissertation describes a number of techniques for analyzing the changes in a more meaningful way that would allow search systems to optimize crawling and provide refined ranking and display of results. By utilizing a temporal language model of a page (i.e., the probabilities of certain words appearing and "lasting" over time) and creating novel mechanisms to track structural changes in documents (a classically difficult tree isomorphism problem) it is possible to more clearly ascertain how information changed on a page.

Chapter 3 also describes a characterization of page evolution based on the way information changes in terms of a textual similarity to some original fixed point. Almost all observed pages display a behavior that can be fit by a two-segment linear model parametrized by a single knot point (see Figure 1.2). This model describes the rapid decay of content (e.g., as blog entries slide off the page) with an eventual stabilization, as every subsequent page is equally (dis)similar to the start point. The second (flat) segment of this model indicates when differences have stabilized. Text that has survived this far is likely to be navigation information or part of the core language model. Knowing that users were interested in the rapidly changing content (i.e., before the knot point), would imply that crawling rates need to be fast enough to collect data before it falls off. However, if the changing information (e.g. advertisements) were irrelevant, a crawler could adapt its behavior by slowing down and consuming fewer resources. The question remains: what portions of the page most visitors are interested in?

Chapter 3 describes the construction of these models and reports on different properties

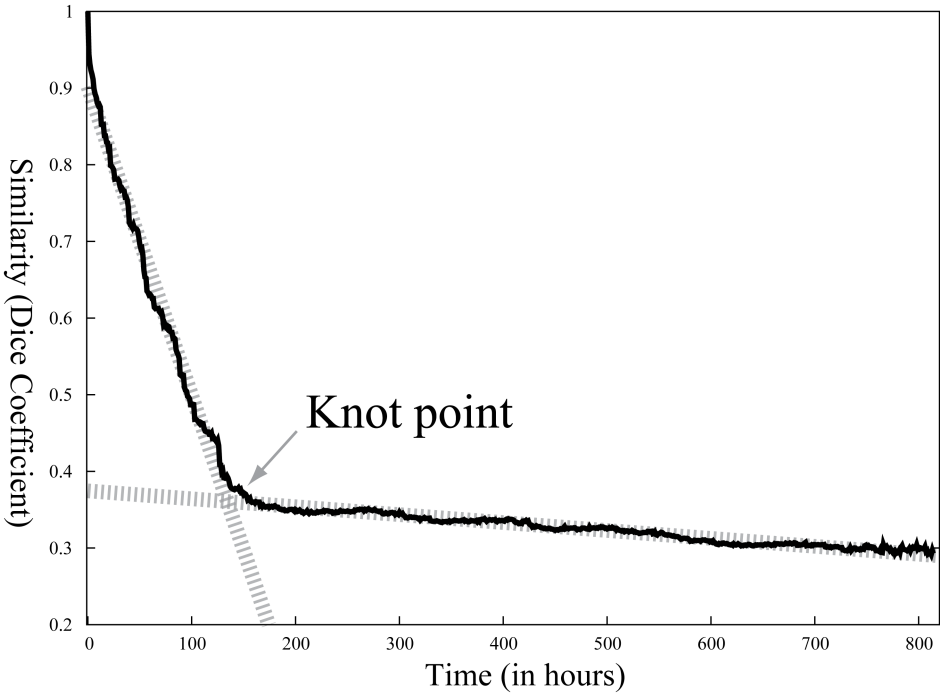


Figure 1.2: Sample change curve plot describing the (dis)similarity of a document to itself over time with time on the x axis and the Dice coefficient on the y (higher Dice indicates higher similarity).

of the Dynamic Web which have never before been observed.

Patterns of Use

The source of the consumption patterns described above was a search toolbar installed by millions of users, a subset of which (600k in one study, 2.3M in another) were used for sampling the Web pages. As the toolbar captured all Web page visits, not just those related to searches, the logs from these users also formed the foundation for the behavioral analysis in the study described in Chapters 4 and 5.

Specifically, the logs were analyzed for revisitation (i.e., re-access by a single user to a given URL). Revisitation is an important behavior, representing 50-80% of page visits [150, 51]. Such statistics are possible to calculate by averaging across a small sample of users, but by leveraging the enormous population of users in the toolbar dataset, the study described

in Chapter 4 characterizes page (and site) level revisitation.

The patterns of revisitation identified in this study may potentially lead to improvements in how search engines, browsers, and Websites treat revisitation. By identifying different types of revisitation and refinding intents, a search engine, for example, can treat queries in a personalized manner, maintaining stability when necessary and interjecting new, improved results when they can make a positive difference.

Resonance

Both content and behavior independently offer interesting insights into different aspects of the Dynamic Web. However, it is in their relationship that interesting new features emerge. Recall that research in the Dynamic Web is riddled with missing data. In part, missing data is the result of the insufficient resources and insufficient instrumentation. Thus, for example, it is extremely difficult to infer what changing content on a page is actually interesting to users. While a page may change on every crawl instance, most individuals may only care about slow-changing content (such as navigation information) or the medium changing content (such as weekly news). Without some behavioral information, it is difficult to go from the raw models of change to "interesting" changes. For this, we require behavioral data. In the past such data may have come from click logs or eye tracking studies. Unfortunately, eye-tracking studies, while informative, are difficult and expensive to deploy at a large scale. On the other hand, click logs frequently do not capture all interesting events. For example, a user of a stock price page may simply look at the price without clicking on anything.

Chapter 4 offers a study that makes use of behavioral data that we *do* have–revisitation. As individuals would "speak with their money," this study makes use of the fact that people will "speak with their visits," when it comes to interesting content. Note that we do not need any information about what is clicked on a page, but simply the timings for visitation. Content of value, both changing and static, will attract repeat visits from users. In this way, the *combination* of change and usage data can be used to achieve what each independently could not.

Having content and behavioral data, one might hypothesize that a connection exists

Figure 1.3: The Woot page filtered by revisitation behavior.

between the way Web content is changing and the rate at which that information is consumed. For example, one might expect that pages that are more popular will change more quickly or that pages that change more frequently are revisited more often. A more complex hypothesis is that different pieces of the page are changing at different rates and that people will synchronize their revisitation patterns to those changes they care about. This synchronization, or *resonance*, might tell us which *parts* of the page are actually of interest.

A number of these hypotheses are borne out in the study. For example, one might anticipate that different *types* of Web pages might have very different revisitation properties based on the targeted content. Take for example three very different pages: the Woot, New York Times, and Costco homepages. On average, people revisit Woot (*woot.com*), an online shopping site, once every 24 to 25 hours. This corresponds precisely to the frequency when Woot introduces a new offer (daily, every 24 hours). As importantly, not every page displays this synchronization. A person will rapidly revisit a page, such as the New York Times homepage, in order to capture stories before they fall off the page (with peak revisitation earlier than the knot point). On the other hand, people may revisit another page, such as the superstore Costco's homepage, long after the knot point, indicating an interest in the stable content on the page (e.g., searching the catalog, finding store hours). The relative relationship between revisitation and change provides us a key hint about the intent in why people return to a given page.

Expanding on the notion of resonance, Chapter 5 explores the idea that calculating the rate of change of content blocks on a Web page and filtering only those that are slightly above and below the average revisitation rate leaves only the content that is the likely target of most users (see Figure 1.3). This notion is quite powerful as it is generally very difficult to determine what piece of a page most people are looking at (Navigation? Search? Some key piece of content?). By using revisitation rates, it is possible to infer the target whether it is navigation content, a specific image, the search box, or any other page element. Though not a complete solution, the technique allows mobile browsers to display content of interest on small screens; a search engine to create search summaries biased towards the users' probable goals; and crawlers to concentrate on keeping up with important changes.

1.3.2 Community and Topic Responses

Having established that there is a relationship between consumption behavior and content over time for *one* Web page, it becomes interesting to consider this relationship for both subsets and super-sets of these time-series. Specifically, one can begin to study the behavioral reaction of different communities to different kinds of information (potentially held on *many* pages). Furthermore, it is possible to begin to understand the driving forces that motivate certain kinds of temporal behavior.

For example, a Web-based news outlet (e.g., CNN) which makes money from advertising has an incentive to keep up with the latest happenings and report those in a higher quality way to the public. In contrast, a casual blogger may have different incentives and social pressures for creating and reporting on different types of content. This behavior may again be very different than what is observed in the more "average" Web population who are simply information searchers and consumers and rarely produce content. These can create differentials in the way different populations produce and consume information about the *same* "event." An event, of course, can mean some specific happening (e.g., an election result, a sports event) but can also be a continuous stimulus (e.g., the number of flu cases, the weather). Detecting how Web-based populations react becomes interesting when the reactions are a) high-correlation proxies for hard-to-detect events, or are b) leading indica-

tors for tracked events. Conversely, we would like to determine which external events and stimuli are leading indicators for Web behaviors (e.g., increased search or blogging volume). In the past few years this type of real-time trend behavior has become highly popular for everything ranging from online meme finding (e.g., [158]), to determining movie performance ([76]), to detecting flu incidence ([69]).

One of the contributions of this dissertation is the analysis of temporal reactions in different populations and Web-based mediums. The aggregation and comparison of behavioral patterns on the Web represent a tremendous opportunity for understanding past behaviors and predicting future behaviors. Through a large scale study correlating the behaviors of Internet users on multiple systems ranging in size from 27 million queries to 14 million blog posts to 20,000 news articles, differentials and leading indicators can be identified. Chapter 6 describes the results of this study and an interface for analyzing the datasets (the DTWExplorer), which includes a novel visual artifact for summarizing the differences between time series, the DTWRadar (see Figure 1.4). Using the analysis tool it is possible to identify a number of behavioral properties that allow us to understand the predictive power of patterns of use. For example, the analysis highlighted the way in which "bottom of the fold" news events (i.e., things that were less likely to be front page news) were correlated with increased blogging behavior, likely due to the pressure for bloggers to offer information that is not likely to have been seen elsewhere.

Differential interest can be measured both in terms of content *and* behavior. Content-based analysis tells us how much information is being created in response to some external stimulus, but this is one step removed from the actual behavior that creates the content and in many situations behavior does not explicitly create content. Behavior-based analysis, on the other hand, tells us much more about what information people are producing and consuming. Unfortunately, such data is hard to get and while we would like to concentrate on the behavioral data streams, in many situations we do not have access to such data and require inferring behavioral information from content. One of the contributions of this work is creating the transformations necessary for comparing true behavioral streams (e.g., query logs) and content streams (e.g., blog posts). By transforming both streams into time series, it is possible to compare behavior and content simultaneously.

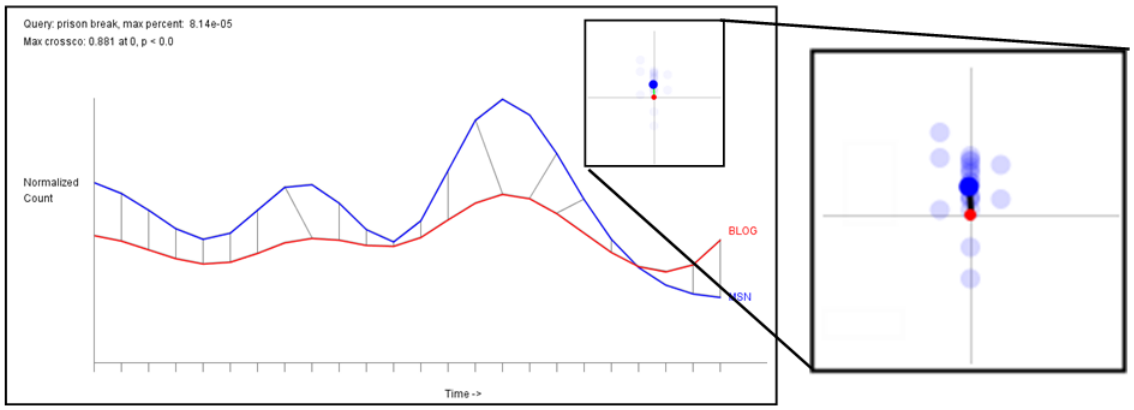


Figure 1.4: Daily interest levels for the show "prison break" in blog and search communities and an overlaid DTWRadar (on right).

1.3.3 Zoetrope: The Dynamic in the Context of the Static

The DTWExplorer system is a sophisticated tool for dealing with very large and very complex behavioral and content streams. Although providing the more technical user with a set of functions to ask and answer questions on Dynamic Web data, there is a fundamental assumption to the tool that this data is somewhat easy to come by. In reality, temporal data (and not just large scale logs) is difficult to obtain. The reality is that data on a Web page will generally vanish when the page changes. For example, the Amazon sales rank for a book will vanish every time the rank is recalculated. Even more "public" data, such as the travel times between two neighboring cities displayed on a government server, will disappear. At best, this data may be stored in an archive somewhere. At worst, it will simply be gone. Unfortunately, even if the data were available somewhere, it may be in a form that requires a great deal of work for an end user to deal with.

One revelation of the work on temporal behavior patterns described above was that it revealed the large amount of *temporal sensitivity* people displayed and simultaneously the lack of any significant tools to support this. That is, people frequently needed to know the temporal context of the information they were accessing: is it new? is it current? has it changed? how has it changed? Unfortunately, the tools of the Now Web, standard browsers

Figure 1.5: Zoetrope in use: the user has found past headline articles about the Ukraine (indicated by the histogram above the lens).

and search engines, are limited to a static perspective of the world. The pages viewed through the browser or the pages indexed by servers are a snapshot which does not allow users to easily ask for or understand data within its true temporal context. The Internet Archive and related efforts seek to capture and archive multiple versions of a page. However, while these systems may store the data that a user might need, they do nothing to help users actually make sense of it.

In response to this problem, the Zoetrope system (described in Chapter 7) was designed to allow users to interact with historical content. So that a user will not need to sift through many pages or leave the familiar Now Web, Zoetrope is designed to work from within the context of the (temporally) current page. For example, the user may go to today's Seattle weather page, which only contains the current temperature, and through Zoetrope, access and visualize past values of the data. This is achieved by drawing a simple rectangular lens around the temperature–thereby visually constructing a query. Through this query, the user specifies the selection criteria to Zoetrope and what information they would like to track over time. A simple move of the timeline slider above the lens brings the extracted content from the past into the "present" of the page (see Figure 1.5). By pulling the information for historical data from its collection, Zoetrope transforms a static data point (e.g., the current

Figure 1.6: A visualization was automatically constructed for the number of Harry Potter books pre-ordered over time (the Muggle Counter)

temperature) into a dynamically constructed time series (see Figure 1.6).

Zoetrope is architected as a complete system including crawlers, various databases and indices, and a front end. Unifying these components is an architecture for describing queries on the historical data, which utilizes a small set of operators that can be combined in many ways to act on a stream of temporal tuples (temporally annotated data). Lenses, which are used to define queries visually, allow users to select the information they want to track through time whether that information is stable visually (always rendered in the same spot), structurally (stable in the page structure), or semantically (stable in textual content). A broad set of transformation and filter operators identify and extract specific historical instances (e.g., "traffic congestion at 6pm every Monday" or "the photograph next to articles that mentions Iraq"). Finally, a set of render operators are able to combine the many historical snapshots into timelines, movies, time series, and other visualizations directly within the interface.

The Zoetrope system seeks to address the end-user problem of dealing with the inevitable fire-hose of data that is the result of moving from the Static to the Dynamic Web.

1.4 Summary of Contributions and Roadmap

Addressing the various facets of the Dynamic Web requires building up new datasets to represent various temporal datasets (content and behavior), creating algorithms to analyze these in a scalable fashion, and building models of how they function independently and in concert.

In seeking to address a number of central questions of the Dynamic Web, this dissertation makes significant progress in this nascent field. I briefly summarize these contributions and the path taken in this dissertation to describe them.

What is the relationship between changing Web content and the ways this data is consumed?

Chapter 3 describes the properties of Dynamic Web content, not just for "randomly" crawled content with low-temporal resolution, but for pages that are actually used (i.e., those that are regularly accessed by users and are not "dormant"). This analysis leads to the development a number of novel models of changing content (i.e., *change curves*) and scalable structural analysis of Web pages. Chapter 4 describes the revisitation behavior of individuals accessing this dynamic content, concentrating on the distribution of revisits (i.e., the *revisitation curve*). Finally, Chapter 5 connects the change and revisitation information to demonstrate how the resonance between the two can automatically identify the *type* of content that is of likely interest (e.g., navigation, search, ads, or news content) as well as *specific* portions of the page without resorting to expensive monitoring techniques.

How can temporal content produced by different communities, and in different mediums, be connected?

Chapter 6 illustrates a mechanism for comparing content and behavioral streams from many different sources to identify leading indicators. The chapter contributes a mechanism for transforming textual and behavioral streams into time-series data and visualization techniques to allow analysts to compare these time-series. The study includes a number of findings based on qualitative analysis of this data that explain behaviors in various Internet communities.

How can we build a system to manage Dynamic Web content and provide new interaction

techniques to consume this information?

Finally, Chapter 7 works through the construction of Zoetrope, an end-user tool that provides individuals with direct access to the ephemeral Dynamic Web from within the context of the static Now Web. Individuals can query historical data within a Web page, answering questions with data that was previously impossible to obtain.

Chapter 8 relates, in more detail, the work described in this dissertation to existing literature. Chapter 9 concludes with a discussion of potential future work in *temporal-informatics*, and specifically in Dynamic Web research.

Chapter 2

Content and Behavioral Streams on the Web

To understand the Dynamic Web and how it might be modeled, it is important to briefly consider the what drives Web change and the concept of "Now" within the context of the dynamics of the Web.

2.1 The Event, the Observation, the Production, and the Consumption

Consider one of the most basic possible pages: one that was created by a single person, accessed once, and never modified or visited again. To start, this page will have a single static fact: the birthplace of former President Bill Clinton (i.e., Hope, Arkansas)[1]. This single fact was recorded by the page's *Author* in HTML and is accessible through a fixed URL–in its static unchanging state, is the simplest part of the Now Web. Such a page will always contain the same information regardless of whether it is accessed now, 5 minutes ago (assuming it existed 5 minutes ago), or 5 minutes from now. A slightly more complex page is one that, like many pages on the Web, records some changing information. That is, the Author–potentially an automated process–may modify some content based on some

[1]Hopefully, an uncontested presidential birth place.

changing observation.

To model both the static and dynamic models of the Web, it is useful to have a model with sufficient granularity that describes the mechanisms that drive production and consumption (while unread pages do exist, they are not particularly useful). One way to achieve this is to think about the creation/consumption for a page. Such a process (graphically represented in Figure 2.1) might look like this:

1. Some *event* happens (T_{event}). An event can be a concrete "real world" action (e.g., Obama takes office) or a state change (e.g., the temperature changed to 89°F). The event may also be some more abstract process, thought, or design process completion (e.g., a new navigation bar is designed for a Web page)[2]. Notably, the "event" might be a Web page being created or modified. For the purposes of this dissertation, an event is defined quite broadly as any "interesting" change of state in any system (real or digital).

2. The event is *observed* by a human or automated process ($T_{observation}$). The observation, in the case of a Web-based events, is simply a page visit (i.e., the user "observes" a Web page). In this case, the observed page (or some part of it) is also the *consumed content*.

3. Finally, the observation may then be (optionally) *produced* in some digital format (i.e., the *produced content*), stored on a server, and accessible by some URL ($T_{production}$). The act of production is in itself an event which, can then be observed, re-produced, and so on (see Figure 2.2).

Note that in all instances we are actually keeping track of *timestamped* information, so in reality we have *tuples* of time and some "extra" data. That extra data may be anything from details about the event, some notation about the observer, the content of the produced

[2]There is still some ambiguity for future events. For an event or action that will happen in the future, we might consider only the time at which some decision, guess, or commitment that this will happen as (i.e., T_{event} is really $T_{decision\_to\_believe\_that\_some\_future\_event\_will\_happen}$. Determining when something has actually happened is somewhat outside the scope of this work.

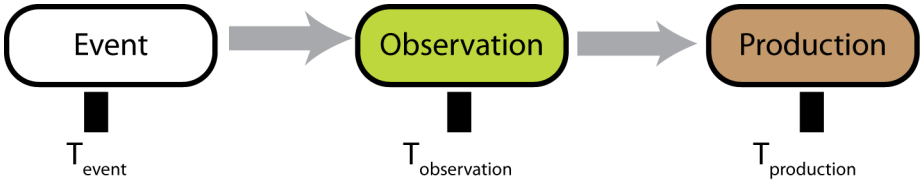


Figure 2.1: Basic page model. An event occurs, an observation is made, and a page is created.

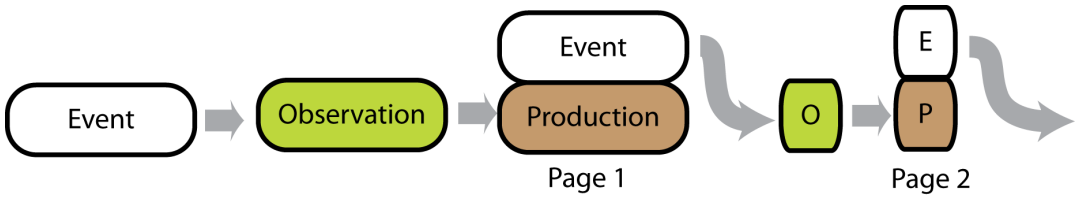


Figure 2.2: A chain of events/observations/productions. Some "real world" event is observed by the author of the first page. The creation/modification of the first page represents an event which is then observed by a second author who may create another page and so on.

page, the demographics of the visitor, or the search term used to find the page. The notation we use for tuples is $\langle T_i, C_j \rangle$ indicating that some data, C_j (potentially describing an event) existed at time T_i. When the content is unimportant, we will simplify this by talking about $\langle T_i \rangle$, or simply T_i, to indicate the tuple at time T_i. Given this model for the Dynamic Web– one based on different states–the Now Web is simply the most recent produced document.

In the case of a dynamic (i.e., changing) page, over time each of the event/observation/ production "modules" in this model (see Figure 2.1) will "output" new information: new events will happen (i.e., Clinton moves), new observations will occur, the page containing the address will be modified, the modified content will be observed as individuals visit or re-visit the page, and so on. Thus, each block produces a sequence of timestamps corresponding to the times when these things happened. All events (such as Clinton moving to a new address) corresponds to the tuple stream: $\{T^0_{event}, T^1_{event}, \ldots, T^n_{event}\}$. Similarly, the observations made of these events form a tuple sequence: $\{T^0_{observation}, T^1_{observation}, \ldots, T^m_{observation}\}$ which

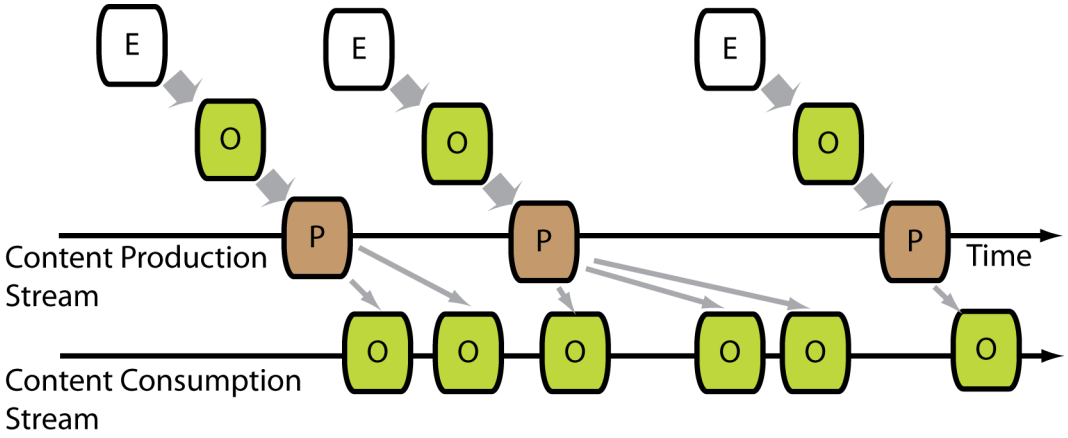


Figure 2.3: A timeline of events around a *single* page. Events are observed at different times and modifications are made to the page resulting in a content production stream (a produced content stream when considering the actual content). A content consumption stream is created when the page is observed, potentially by multiple users, over time.

lead to page updates $\{T^0_{production}, T^1_{production}, \ldots, T^r_{production}\}$. The behavior sequence of page productions is the *Content Production* stream, whereas the *actual* artifacts produced (i.e., timestamped content) is the *Produced Content Stream*. When the observations correspond to page visits, we refer to this sequence as the *Content Consumption* stream.

What we have considered so far has been focused almost exclusively on a single page (see Figure 2.3)–where we have built a model of the Dynamic *Page* rather than the Dynamic *Web*. Although we will return to this, it is worth briefly considering what this page model looks like in the context of the larger Web. First, we note that there might be many related events, many related observations, many related Web pages, and many related visits. In order to leverage this in analysis, we can utilize the notion of a *grouping criteria* which lump timestamped content and events. A grouping criteria might be all Web pages about Bill Clinton, or all blogs about last week's baseball game, or all visitors that access pages about XBox games. Such grouping will become useful when seeking to understand production and consumption patterns around *types of users*, *topics*, or *events*.

2.1.1 Ambiguity, Noisy Signals, and Lost Data

Where the model as a whole becomes interesting is in recognizing the amount of missing data and ambiguity in it. The key sources of ambiguity are that a) there is not a one-to-one, causal relationship between event, observation, and production, b) we do not know definitively when events, observations, or productions will happen (or for that matter have happened), c) we are unable to instrument every possible person and service to capture every change. These problems exist in some sense for the individuals creating and modifying the page, the visitors accessing the information, and the monitoring researcher/service.

These ambiguities, frequently caused by missing data, motivate the research and contributions presented here–driving analysis, inference techniques, and end-user tools. The creation of systems to deal with this data, and resolving the ambiguities in it, corresponds to the central dissertation questions presented in the Chapter 1. Below, we provide a brief description of ambiguity, noise, and lost data that emerge when the reality of the Web meets the simple model described here.

The Ambiguity of Time

There are two important temporal variables we would like to resolve in our model: when something did happen and when will it happen. Determining when something did happen is critical for correctly assigning a temporal label to the event, observation, or page production. While this would seem like an easy problem–just look at the clock–in reality it is challenging. Ideally, for any event we would have $T_{event} = T_{observation} = T_{production}$ (i.e., the page is modified every time an event happens at exactly the time this happens) where T_{event} is known. It is impossible to ask each Website owner to report when and why a page was changed or to "push" notifications of change. Because of this limitation, individuals or automated services are forced to "pull" or "query" for new or interesting events. The only thing we can (abstractly) say is that recording follows observations follows event, or:

$$T^i_{event} \leq T^j_{observation} \leq T^k_{production}$$

Because we are unable to instrument every page and every person, the best we can do is estimate. This problem, again, is inherent in the "pull" nature of the Web. Ideally we would

know exactly when an event happened (e.g., when did Clinton move? when did a new article about Google appear on the NY Times homepage?). Note that this is desirable as "time" (or some range of times) is ideally the foreign key in our dataset (i.e., just determining the *ordering* of events, observations, and production is usually insufficient). In order to create a lower bound on these times we must generally repeat observations until a change is witnessed. At worst, this means guessing that $T^{n-1}_{observation} \leq T_{event} \leq T^{n}_{observation}$. That is, the observer has to guess that the event happened sometime between the two most recent observations. At best, some correction might be possible so that one could guess a T'_{event} that is close to the real T_{event}.

In the work presented here, the most important times to identify are the timestamps that we associate with behavior and content streams. That is, when an action (i.e., observation) occurred and when a piece of content was modified or created. One strategy that we apply in Chapter 3 to determine $T_{production}$ is to try and capture a page as near in time to when it changed. One way to determine this crawling interval in regularly changing content is a *backoff protocol* where fast crawls (many retrievals) are slowed down if no change is detected and sped up when many changes are observed.

Determining the exact time something *has* happened is certainly of value for Dynamic Web analysis when we are mining existing data, but is it possible to determine when something *will* happen? Doing so would have two benefits. First, we would be able to refine our crawling strategy to capture the change or event as near to the time at which it actually happened. Second, we would be able to create services that are reactive–adapting quickly to predicted changes in behavior or content (e.g., changing indexing strategies, advertising).

In situations when page changes are regular we can estimate when a page will change and only around that time crawl to test for changes. This would reduce the number of page retrievals. This estimate can be derived using the backoff protocol described above or by re-using the idea of resonance. When re-visitation rates can be determined, these might form an initial input into the crawling system (i.e., the "seed" re-crawl interval). Unfortunately, this strategy is not as effective in situations where there is no regularity to modifications (e.g., a news spike around some event).

Chapter 6 introduces the notion that it is possible to *anticipate* reactions. For example, it is possible to determine when bloggers will start creating new content in response to some news event or TV show or when the population of visitors will begin looking for information about some news event. While this may be difficult or impossible to calculate for a specific page or individual user in reaction to some random event, we are able to anticipate reactions in groups. That is, some set of data may contain leading indicators that allows us to predict, or assign some distributions, to the times in the model.

It is worth considering that better estimates for T_{event} might be achieved by utilizing the temporal information in the content itself to infer the event and observation times. This is difficult, however, because the information might be about something in the past (e.g., a page created in 2008 talking about Lincoln), a potential future event (e.g., a page written on August 12, 2009 about a trip on September 20, 2009), or something with validity "bounds" (e.g., a fact that is only valid between certain dates). Interpreting the information on the page in this way introduces a level of complexity that is out of the scope of this work.

Incomplete Observation and Production

The other side effect of being unable to track *all* events, page changes, and other behavioral data is that the content streams are frequently incomplete. For example, the content stream[3] that is collected by an external process (i.e., crawler), P', may only be a subset of the ideal content stream. Thus, if the true event stream, E, has m events, the ideal content stream P would capture all m events, or:

$$E = \{T^0_{event}, T^1_{event}, \ldots, T^m_{event}\}$$
$$P = \{T^0_{production}, T^1_{production}, \ldots, T^m_{production}\}$$

Ideally, with the times of production and the event being similar (i.e., $T^i_{event} = T^i_{production}$). However, because not every event is captured we may only have:

$$P' \subset P$$

[3]The same applies to event, observation, production and visit streams which may only be recorded partially

One approach to dealing with this issue is trying to identify proxies for various streams. Thus, while one particular dataset may be difficult to obtain or analyze, another with similar properties may approximate the missing stream. A consequence of the "anticipation" work described in Chapter 6 is that it is possible to identify streams with sufficiently high correlation that they might be used as proxies (e.g., a stream of blog posts about a topic may model a stream of searches about that topic).

Having sufficient resources to capture the complete streams is an issue not just for us, as scientists trying to understand the Dynamic Web, but also the individuals producing and consuming data on the Web. For example, individuals and communities *creating* Web pages would like to keep up with all changing information but are frequently unable to do so. One potential solution is that the we might find two pages, $page_1$ and $page_2$, that are modified in response to the same tracked event (say Clinton's biographical information). Unfortunately, the editor of $page_2$ doesn't keep as up to date, so the produced content stream, P_2' for this page tends to be very sparse with many missing observations. The author of $page_1$ has many more resources so P_1' tends to have tuples for most events. The problem then is that visitors (e.g., Web surfers) looking at $page_2$ may not get the most accurate reflection of reality.

The idea of is then to a) find a mapping in co-evolving content streams (namely P_1' and P_2') and b) automatically adjust the "lagging" content stream. In this way it is also possible to produce a *combined* content stream, composed of both P_1' and P_2', that has a better overlap with ideal stream. This technique has been successfully applied in static Web contexts ([3]), and may work well for the Dynamic Web as well.

Composite pages

The best scenario for a researcher studying data on the Web is that one event (e.g., Clinton's address changed), is observed by one person, and recorded on one page with no additional information. There would be no question *what* the page author was interested in recording (Clinton's address), and *why* the author made a change (Clinton's address changed) and what particular information a visitor was interested in (Clinton's address). Unfortunately,

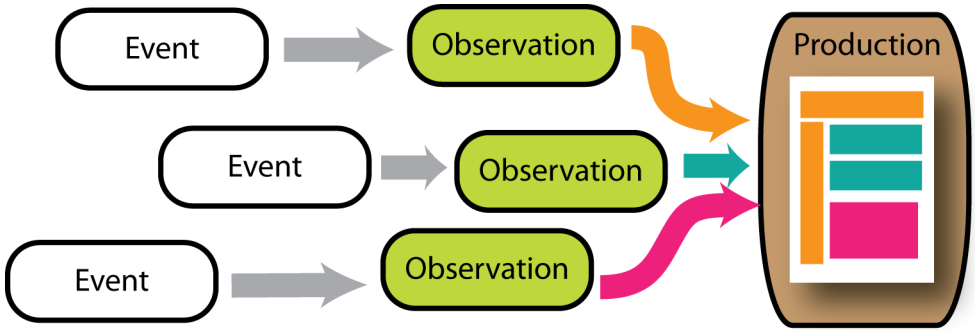


Figure 2.4: A page is composed of multiple pieces of data. Multiple observations, potentially made at different times, lead to modifications on different parts of the page.

the world is not so simple.

Our Bill Clinton page likely contains many independently changing blocks of content, which capture updated stories, biographical information, navigation elements, advertising, and other embedded content (e.g., images). Thus, a page (see Figure 2.4) might be a result of different events, different observations, and different modifications. Each modified piece on the page may change at a different rate. Although the author's motivations may still be fixed (information about Clinton), the page may now contain information unrelated to Clinton (e.g., advertisements, structural elements, navigation and search bars, visitor counts). This composite page presents a challenge for a researcher or automated system attempting to automatically identify "interesting" events and changes. Furthermore, studying *visitors* to this composite content is hard as there is no way to know what on the page is the target of the consumption behavior, thus making the building of refined consumption behavior streams a difficult task.

A solution to both problems, however, is possible when we take advantage of temporality in changes and visitor behavior. Recall, the idea of *resonance*–that in an attempt to capture important changes and events individuals will revisit a page at the rate of change of whatever they find interesting or important. Validating and utilizing this idea requires some way to analyze content streams to find different "types" of content on the page (i.e., static navigation elements versus quick changing news) as well as the specific blocks on the page

that are changing at different rates. Second, we require a more sophisticated analysis of the behavioral data (i.e. the content consumption stream) to identify different types of revisitation and transform that data into a form that can be used in concert with the content stream. The precise mechanism by which content streams are analyzed, blocks identified, and revisits analyzed is described in Chapter 5.

2.2 Filtering, Time-series, and Aggregation

In addition to the problems inherent in the data itself, there is also the issue of translating between "raw" content streams to something that can be used for analysis. That is, how does one get to a clean time-series representation of topics, or sessions, or translate from content to behavior (or vice versa) when one starts with raw data such as:

1. *Page Streams*–time annotated versions of the *same* page (i.e., the page accessed through the same URL.

2. *Click logs*–time-annotated behavioral streams capturing the surfing behavior of a user (identified generally through a user ID or session ID)

3. The related *Search logs*–a subset of click logs– capturing time-annotated search engine queries and associated URLs that were clicked on (again identified through a user ID or session ID).

Between the two extremes of raw content and behavioral streams and the "refined" analysis and system goals described above, there are intermediate forms of these content streams that more directly serve our purposes. Though such intermediate forms will be defined as necessary throughout this work, a number of common representations and operations exist that transform the raw data, such as timestamped search and click logs and complete Web pages, into discrete components and numerical time-series. This is generally done through: aggregation, filtering, and time-series conversion.

2.2.1 Aggregation and Filtering

Raw data streams from a single page or a single user are frequently too sparse to be usable for inference tasks. A solution to this is to filter and combine different streams. For example:

- all news articles published by CNN, MSNBC, and Fox News might be combined into one "news" content stream.

- all click logs for individuals accessing the New York Times homepage might be aggregated into a behavioral stream.

- all searches for "Britney Spears" can be combined into a Britney Spears search stream.

The form of aggregation described above functions across multiple streams. It is also possible to perform aggregation operations on single content streams in order to normalize or generate histograms. This type of transformation will be useful when, for example, different page revisitation patterns need to be compared. In situations where data is sparse, aggregation can also be used to bin different events, observations, or pages that occurred near each other (e.g., the number of news articles in each 10 minute interval starting at some time).

2.2.2 Time-Series

In general, the data we start with is *not* organized in a form that is useful for mathematical operations. For example, a Web page is largely unstructured textual information. However, we would still like to be able to answer questions about how the data has changed and how it relates to other temporal data. Thus, it is generally much easier to work with temporal data when it is in some numerical form such as a time series (i.e., pairs of timestamped numerical values).

There are a number of ways we implement such transformations. In the case of text, we will frequently convert the textual representation into some vector space form [18]. A time-series representation of such a stream might be the amount of change between one version

and the last. A different representation might be the number of different news articles or blog posts that mention a specific term over time.

By transforming the raw data into different time-series it becomes possible to a) compare different types of raw behavior and content data, and b) apply standard quantitative analytical techniques.

2.3 Summary

The model described above considers the way in which information is produced and consumed, describing it in terms of the event, the observation, and the production of digital content. This model provides a common framing for the data analyzed in later chapters.

The resulting content and behavioral streams that are described above represent a way of thinking about Dynamic Web data that will be used throughout this dissertation. The stream of data that is produced at each step–generally in the form of raw data–is processed, cleaned, and transformed into something that can be used to answer concrete questions about the Web.

The problems that affect this model include different forms of ambiguity and data loss. A number of the contributions are a direct result of addressing these limitations. Thus, both the framing of the data in these terms, and of the issues surrounding data use, unify much of the next few chapters.

Chapter 3

Content Change

Before resolving any of the ambiguities or problems inherent in Dynamic Web data, it is worth considering exactly what dynamic content looks like. That is, what is the nature of the produced content stream (Figure 3.1)? How often are changes made? How much is changed? What stays the same? Answering these questions is critical to understanding the Dynamic Web and building systems around it.

The fact that the Web is highly dynamic has been acknowledged (in a very limited way) from its early days. Realizing that content is created, changes, and vanishes led to the development of services such as the Internet Archives. Search engines have also had to respond to changing content by modifying crawler behavior and indexing strategies [48, 53]. From the perspective of the end-user, *monitoring* changing content is a common activity. Many monitored changes are quite subtle, and many "differencing" tools have been developed to help (e.g., [57, 153]).

Unfortunately, much of the modeling work on the Dynamic Web is based on studies and data that does not reflect the way the Web is actually *used*. Because of the power-law like distribution of visitors to pages (see Section 3.1.2), any page sampled at random from the Web is likely to never be visited (i.e., it comes from the tail of the visitor distribution). Furthermore, the general strategy for pages requires crawling at somewhat slow rates (i.e., once a day at best). Recall, that this leads to a sparsity in the content stream data discussed in Chapter 2.

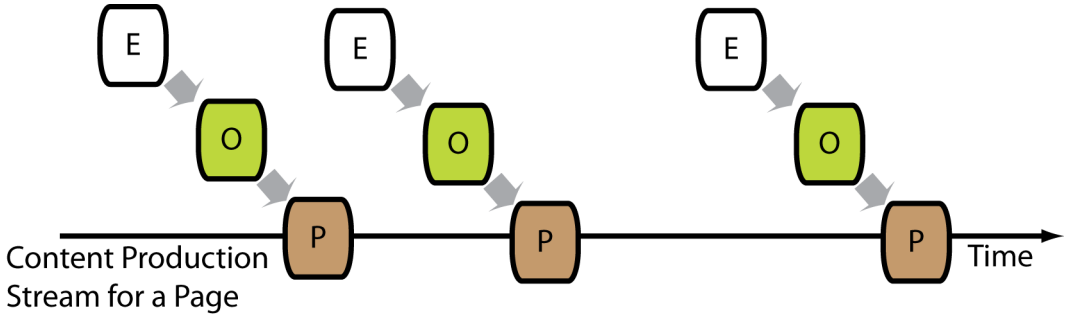


Figure 3.1: A timeline of events around a *single* page. Events are observed at different times and modifications are made to the page resulting in a content production stream (a produced content stream when considering the actual content).

This chapter and the next describes two of the raw streams: content and click behavior. The two streams are unique in their reliance on real user behavior from millions of individuals. This chapter describes the collection and analysis of a Dynamic Web dataset that attempts to more closely model the changes in the pages that actually inspire use and reuse. The data differs from previous work (e.g., [59, 103, 126, 128] but see Section 8.1 for a complete discussion) in the pages selected for crawling (sampled from pages that we know have been visited), the granularity of the crawl (hourly and sub-hourly crawls), and in the level of detail with which changes in the content and internal structure of Web pages are analyzed. Through analysis of this unique dataset, this chapter describes various characteristics of changing Web data and new new algorithms for exploring and utilizing changes in a number of applications. Many pages change in a way that can be represented using a two-segment, piece-wise linear model that would be unobservable in a coarser crawl.

The main dataset utilized for this analysis is a multi-week Web crawl of 55,000 pages. The pages in the crawl were selected to represent a range of visitation patterns, and are unique in that it reflects how the observed Web is changing. Understanding this segment of the Web is important for crawlers and search engines that must keep pace with changing content as well as applications intended to help people interact with dynamic Web content. Pages in this dataset were collected through hourly, sub-hourly, and concurrent crawls (i.e., crawls executed from two machines simultaneously) and studied at different levels of

granularity:

1. *Gross textual changes* (Section 3.2)–In looking at the page as a whole, one obvious question is *how much* has the page changed between versions? Furthermore, how has that page evolved over time? "Gross" textual changes measure the differences between page versions by utilizing metrics that ignore any fine-grained textual or structural features of the page.

2. *Term/language model changes* (Section 3.3)–Expanding beyond gross textual changes, it is valuable to ask *what* is changing on the page? Through an analysis of the language model for the page, it is possible to identify term-level features of the page that survive or vanish (i.e., terms with high or low *staying power* respectively).

3. *Structural changes* (Section 3.4)–The survivability of various DOM elements is critical for many Web clipping and template extraction applications which rely on the structure to function [2, 31, 54]. The study explore the dynamics of Web page structure through analysis of DOM-level changes.

The chapter begins with a description of the dataset utilized in this study and continues with an exploration of both content and structural change in greater detail. The study presented is the result of collaboration with Jaime Teevan, Susan Dumais and Jon Elsas ([6]).

3.1 Methodology and Data

3.1.1 User Selection

The base dataset for this study is page visitation logs collected from opt-in users of the Windows Live Toolbar. The toolbar provides augmented search features and reports anonymized usage behavior to a central server. The study makes use of data from a sample of 612,000 users for a five week period starting August 1, 2006. The five week period is sufficient to

capture a wide variety of revisitation patterns, although we may seasonal or yearly patterns. However, as part of a different study of data from May and June of 2007 (described in Chapter 5), the same general patterns of use were observed.

Users were identified by an anonymized ID associated with each toolbar. As is the case with large-scale log analyses, if more than one person uses the same computer, they will have the same ID. However, it is likely that with the exception of some very popular URLs, revisits will be user specific. To simplify the discussion, a toolbar instance will be referred to as a "user."

For simplicity, the sample was restricted to to data gathered from English speaking users in the United States. Additionally, only users for which there was data for over 7 days were considered. In order to eliminate the "long-tail" of non-representative instances, users activities were characterized on a number of dimensions including: total number of URLs visited, unique URLs visited, number of URLs revisited, and number of URLs visited per day. The lowest ten percent of users in any of these dimensions were removed from the study population. Finally, users in the top one percent were also excluded (to eliminate robots and other badly behaving clients). The result of this filtering, for the time period of the study is over 612,000 valid users.

3.1.2 Page Selection

The goal in selecting Web pages for this study was to sample pages with a diversity of revisitation patterns. This is a significant departure from other studies that have traced the evolution of subsets of Web pages (typically gathered by breadth-first crawls from seed pages) regardless of use. Instead, this study captures a sample representing those pages which are actually revisited and consumed by users in different ways.

Beyond sampling pages with different popularity, pages were sampled based on revisitation patterns. The patterns were defined by a number of attributes for each page which were then used to systematically sample pages for further analysis. Specifically, sampling in this work considered the number of unique visitors to a page (unique-visitors), the average and median inter-arrival (i.e. revisit) times for a page (inter-arrival time), and the average

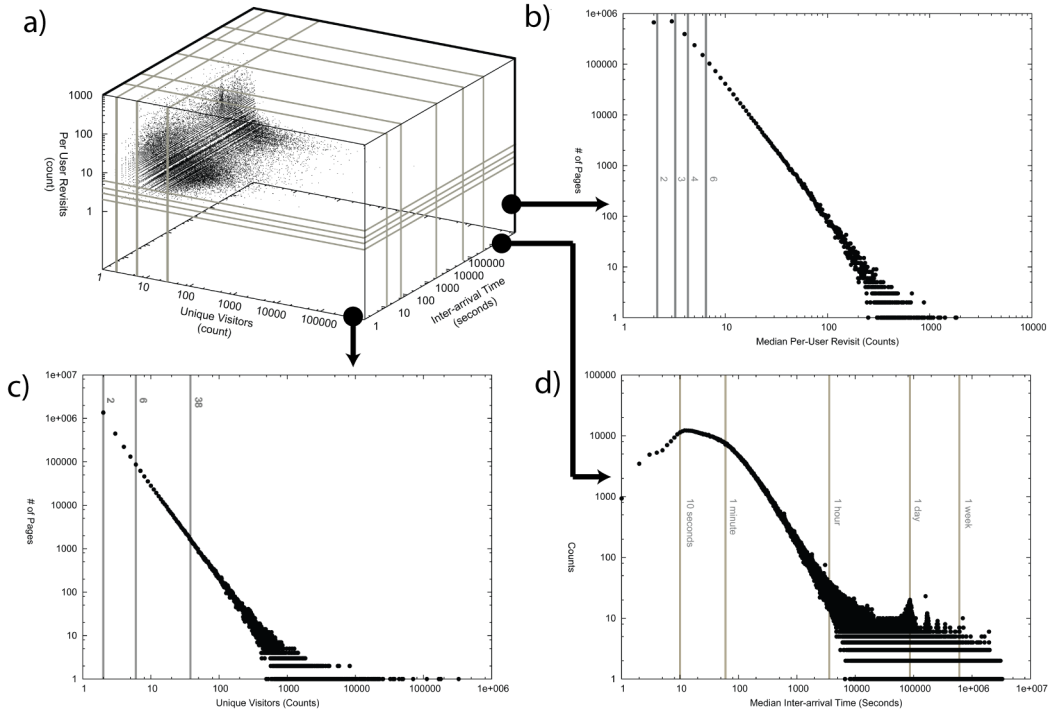


Figure 3.2: A (log-scaled) scatter plot of pages (each point represents a single page) along the three dimensions of interest. 1b-d (also Figures 3.3-3.5) are the histograms for per-user visit, unique visitors, and inter-arrival time respectively (each point is a set of pages). The 55k pages were sampled from these distributions.

and median number of revisits per user for a page (per-user revisits). Per-user revisits for a page represents the median number of times a user will revisit that page.

Figure 3.2a shows a scatter plot of pages along these three dimensions, as well as histograms for each of the three criteria individually (log scaled). Figure 3.2c (and Figure 3.4) highlights that most pages were visited by very few people overall (the top leftmost point indicating that over one million pages only had one unique visitor), with a small number of revisits per person (Figure 3.2b and Figure 3.3), and at low (fast) inter-arrival times (Figure 3.2d and Figure 3.5). To sample broadly from this space without over-selecting from this "long-tail" we applied a pseudo-exponential binning to find thresholds for the unique-visitor (with a minimum of two) and per-user revisit criteria. The specific thresholds are noted as vertical lines in Figure 3.2b-d. For the inter-arrival time criteria we opted to use thresholds

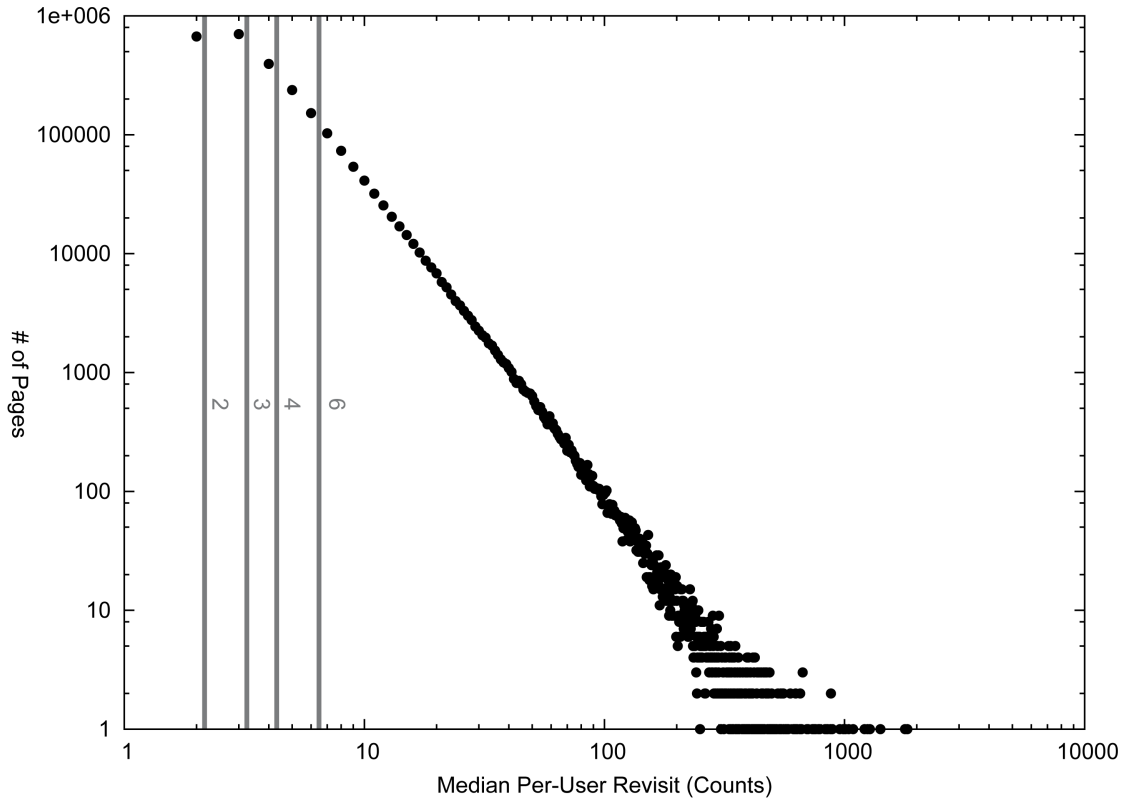


Figure 3.3: A histogram representing the number of pages (y-axis) with the given median per-user revisits (x-axis), log-log scale).

that correspond to well understood time periods (e.g. 10 seconds, 1 minute, 1 hour, 1 day, 1 week).

As illustrated in Figure 3.2a, four bins were defined for the unique-visitor criteria, five for the per-user revisit criteria, and six for the inter-arrival time criteria (for a combination of $4 \times 5 \times 6 = 120$ possible bins). Each URL was annotated with these three bin numbers, and sampled from this space. Some oversampling of popular pages was added by explicitly including the top 5000 most visited pages. Pages were crawled to ensure that they were still publicly available (in conformance with the robots.txt protocol), and removed from further analysis those that were not. The final sample included 54,788 URLs with an average of 468.3 (median 650) URLs in each of the 120 bins.

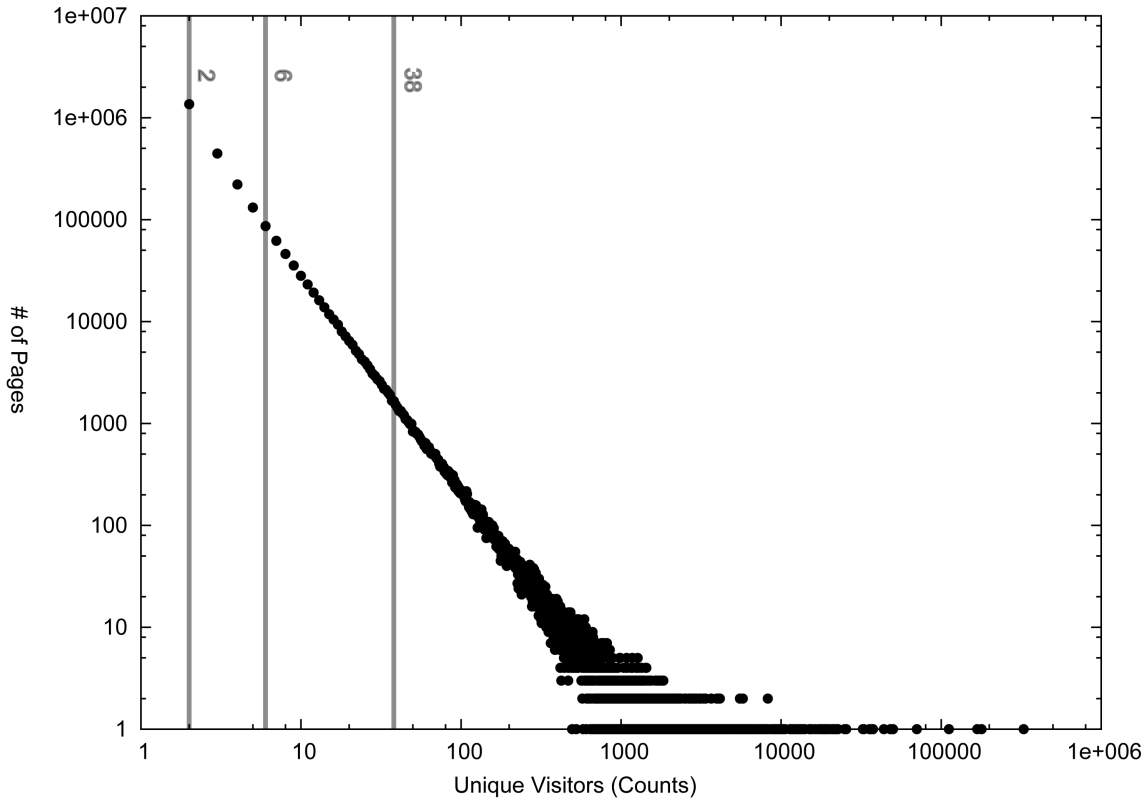


Figure 3.4: A histogram representing the number of pages (y-axis) with the given number of unique visitors (x-axis), log-log scale.

Each page was categorized into topical categories using standard text classification techniques [143]. The topical categories used are similar to those in the first two levels of the Open Directory [136]. A second classifier, used by Live Search, identified various genres (e.g., children, mail, pornography, etc.). Additionally, pages were grouped by the number of unique visitors (as described above), the URL depth (how deep the page is within the site), and top level domain.

3.1.3 Crawling Strategy

To understand how the content of these Web pages change over time, we crawled each URL hourly, starting May 24, 2007. Because of some randomization in the crawler, the time

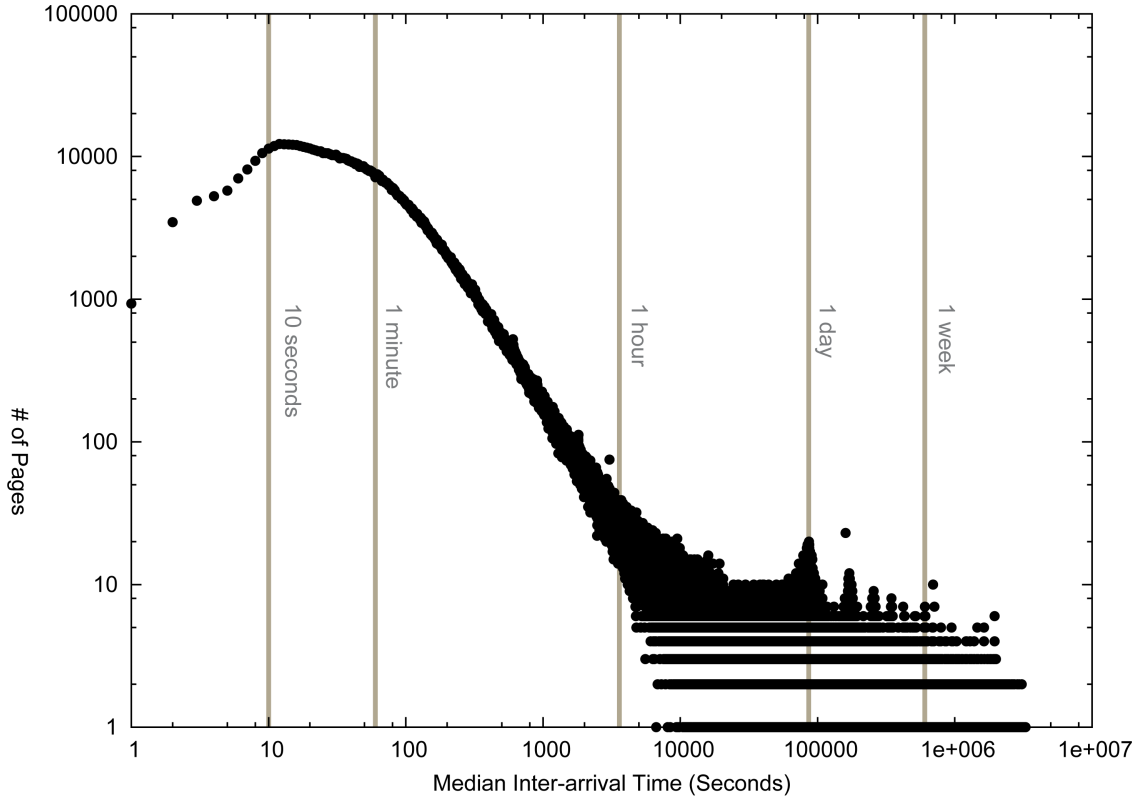


Figure 3.5: A histogram representing the number of pages (y-axis) with the given median per-user revisit *interval* (x-axis).

between subsequent re-crawls may be slightly more or less than an hour. For those pages that displayed a substantial number of hourly changes a secondary crawl was initiated over the course of four days to capture sub-hour changes, as described in Section 3.2.3. Finally, for pages that displayed many changes at the two-minute interval, a third crawl was constructed to collect the page twice at the same time from two synchronized machines. All three crawls capture changes at a much finer grain than previous work. The full HTML text of the page was collected and stored for each retrieved version.

The bulk of the analysis in this study is on a five week period starting on May 24, 2007. However, the crawl was continued beyond this period and a somewhat larger sample of 6 months (from June to December of 2007) was utilized to study text content changes in

greater detail.

3.2 Content Change

3.2.1 Overview

There are a number of alternative measures of "difference" or "change" between two or more versions of the same document. An extremely coarse measurement of change can make use of document checksums that detect any difference. In a way this is unsatisfying as both large and small changes, as well as content and structural changes, are treated the same.

Previous work has focused on measuring textual differences by, for example, calculating the differences between blocks of text [59] or word frequencies [126]. The study here begins with a similar approach, based on the Dice coefficient, to represent the amount of textual change. The Dice coefficient, which measures the overlap in text between various document versions, allows us to develop a high level model of page change over time (i.e., $Dice(W_i, W_k) = 2 * |W_i \cap W_k| / (|W_i| + |W_k|)$, where W_i and W_k are sets of words for the document at time i and k respectively). A high Dice coefficient (i.e., 1) reflects high similarity, whereas a low Dice coefficient (i.e., 0) indicates no similarity.

One way to utilize the Dice coefficient is to determine the frequency and *amount* of change for each page individually. As an example, a page may not change at all for several days after it is first crawled, but change a lot when it finally does. The same page may then begin to experience small changes at frequent intervals. All of the changes that are made to a page over the period of this crawl can be represented as a scatter plot like the ones shown in Figure 3.6. Each point represents an instance where the page changes. The amount of time elapsed since the last change is represented along the x-axis, and the amount of change that occurs, measured using the Dice coefficient, is represented along the y-axis. For example, the plot for the New York Times homepage in Figure 3.6 shows most changes occur with every crawl (i.e., every 60 minutes), although some happen after two (red points). In contrast, Figure 3.6 also represents a page that provides horoscopes and shows larger changes occurring on a daily basis (blue points).

In the studied dataset, 18971 pages (34%), displayed no change during the study inter-

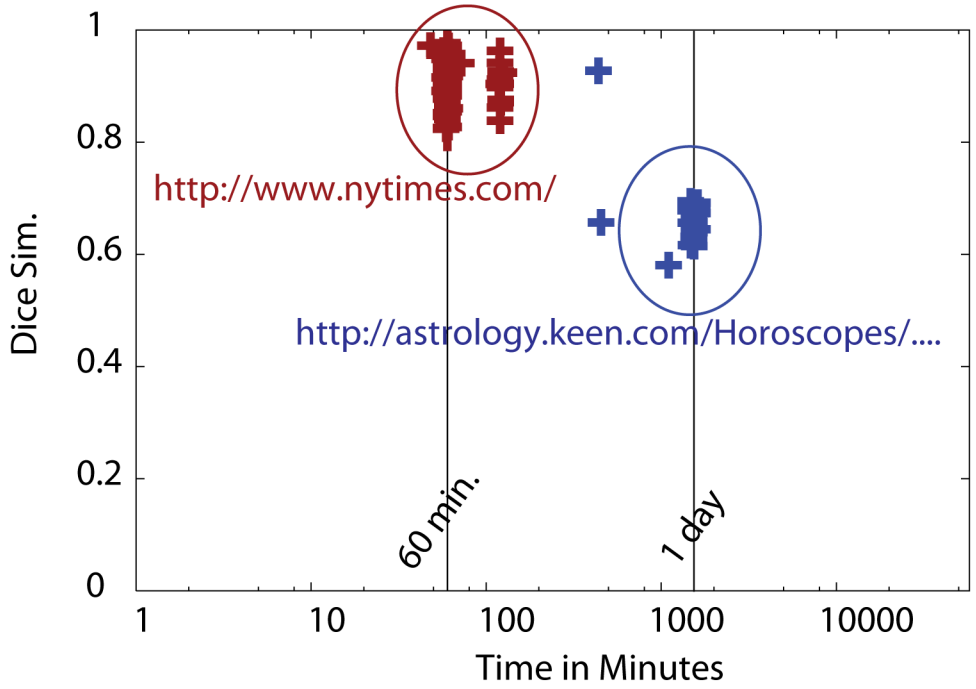


Figure 3.6: The interval between successive changes, plotted against the amount of change for two pages. Note that the NY Times page changes on every crawl (i.e., every 60 min.).

val. On average, documents that displayed some change (35,997 pages) did so every 123 hours and the average Dice coefficient for the change was 0.794. Figure 3.7a shows the distribution of average inter-change times, and Figure 3.7b shows the average inter-version Dice coefficients.

3.2.2 Change by Page Type

The mean time between changes, and the mean Dice coefficient for the changes that did occur, are shown in Table 3.1, broken down along several different lines including: the number of visitors, top-level domain, URL depth, and 6 most prevalent categories. For each page, we calculate the average time between changes (i.e., the inter-version time) as well as the amount of change (as reflected by the Dice similarity). This roughly corresponds to the centroid of the points for each page in Figure 3.6. The mean inter-version time (in hours)

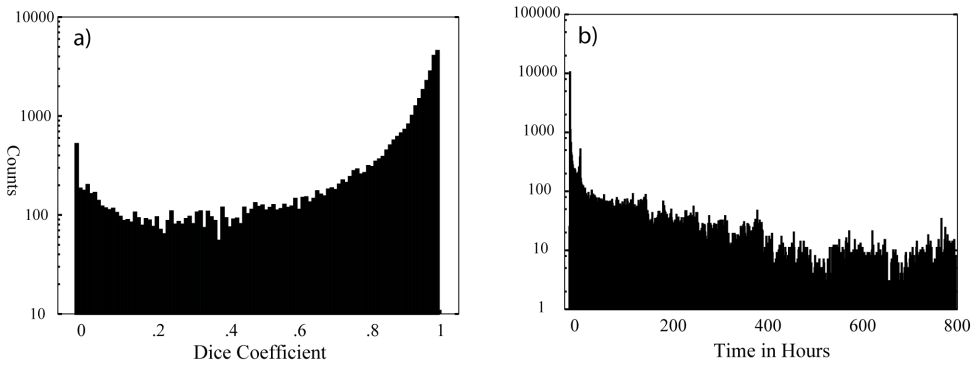


Figure 3.7: Histograms of a) the average amount the content changed, as measured by the Dice Coefficient and b) the average time between changes in our dataset for all sequential pairs of Web pages downloaded from the same URL.

and Dice coefficient are then calculated for pages of the same class (e.g., News pages or pages in the .edu domain). In this section we concentrate on the mean inter-arrival times and similarities, returning to the knot data in subsequent sections. All differences in the table are significant ($p<0.0001$) with the exception of differences in mean change interval that are less than 10 hours, and differences in the mean Dice coefficient that are less than 0.02. Additionally, the .org top-level domain Dice coefficient is only significantly different from .net.

When broken down by unique visitors, we observe that pages that are more popular tend to change more frequently than less popular pages. This most likely explains why the percentage of pages in this sample that displayed no change was nearly half of what has been observed in prior studies ([48, 59]). Pages that are revisited (as all pages in this sample are) change more often than pages collected using other sampling techniques. However, while the number of unique visitors is related to the rate of change, it does not appear to correspond to the amount of change.

Looking at change by top-level domain, it appears that educational and government domain addresses do not change as frequently or as much as pages in other domains do. This is consistent with prior research ([59, 98]), and may reflect the fact that sites in these domains typically provide richer and less transient content that only requires small,

Table 3.1: Pages change at different rates and change different amounts. This table shows the mean interval between changes, the mean amount of change to a page, and the mean knot point, broken down by page type (items with * are not significant).

		Inter-version means		**Knot Point Location**	
		Hours	*Dice*	*Hours*	*Dice*
	Total	123	0.794	145	0.7372
Visitors	2	138	0.8022	.146*	.7594*
	6-Mar	125	0.8268	143*	.7692*
	Jul-38	106	0.8252	145*	.7458*
	39+	102	0.8123	139*	.7621*
Domain	.gov	169	0.8358	153	0.8177
	.edu	161	0.8753	200	0.8109
	.com	126	0.7882	145	0.7408
	.net	125	0.7642	132	0.7195
	.org	95	0.8518	129	0.6743
URL depth	5+	199	0.6782	173	0.715
	4	176	0.7401	159	0.7413
	3	167	0.7363	160	0.7378
	2	127	0.7804	144	0.734
	1	104	0.82	137	0.7432
	0	80	0.8584	141	0.7334
Category	Industry/trade	218	0.6649	156	0.668
	Music	147	0.8013	166	0.7693
	Porn	137	0.7649	140	0.7365
	Personal pages	88	0.8288	124	0.7347
	Sports/recreation	66	0.8975	137	0.7138
	News/magazines	33	0.87	104	0.6415

infrequent updates.

Pages with deep URL structure change less frequently than pages at the root, but change more when they do. This may be because top level pages are used as jumping points into the site, with a high navigational to content ratio. Thus while new ads or short teaser links to new deep content are often added, these are fairly minor changes. Internal pages, which may have a low navigation to content ratio, will see large changes when they do change.

The final breakdown of change shows that pages from different topical categories change at different rates and by different amounts. Pages from categories like News, Sports and Personal pages change the most frequently as would be expected. The amount of change, however, is not generally as large as that seen for categories that change less frequently.

3.2.3 Change over Short Intervals

Because 22818 of the pages in the sample changed nearly every hour, two sub-hour crawls (fast-change and instant-change) were initiated to gain a better understanding of how these pages changes at intervals of less than an hour. Fast-change pages were collected by taking a single snapshot of the page, a second one 2 minutes after the first snapshot, a third 16 minutes after the first snapshot, and a fourth at 32 minutes. This was done repeatedly over a period of three days, with each sample (i.e., the set of four snapshots at 0/2/16/32 minutes) shifted by 4 hours in order to capture different conditions (e.g., during the day versus at night), for a total of 8 four-sample rounds. For those pages that displayed changes during the initial 2 minute delay (10,616 pages), instant-change data was collected by simultaneously requesting the page on two synchronized machines. Although this data does not necessarily specify rates of change, it does indicate if pages change through some automated process on every request rather than changing as a result of some human-driven process.

Table 3.2 presents high level statistics for this analysis, showing the number of documents that changed at least once during sampling as well as the number of pages that change in all samples. The range between the two approximates the lower and upper bound on the true number of pages that change at a given rate. In all cases, the Dice similarity is high, reflecting small amounts of change in the short time-scales. As the change interval gets

Table 3.2: Change properties for sub-hour crawls (60 minutes row statistics are from the original "slow" crawl).

Interval	**Docs. with change (% of 54816)**		**Mean / Median Dice coeff. (given change)**
	Change in any sample (upper bound)	*Change in all samples (lower bound)*	
Instant (0 min.)	6303 (11.5%)	3549 (6.5%)	.937 / .985
2 minutes	10616 (19.3%)	4952 (9%)	.937 / .982
16 minutes	12503 (22.8%)	6206 (11.3%)	.927 / .975
32 minutes	13126 (23.9%)	6445 (11.8%)	.920 / .969
60 minutes	35997 (65.7%)	22818 (41.6%)	.867 / .950

longer, the amount of change a page undergoes increases. Manual analysis of documents displaying instant change reveals many instances of randomly changing advertisements. Additionally, details about how fast a page was served (e.g., "Page served in 0.00252 secs,") to whom (e.g., "IP address is..") , and to how many (e.g., "There are 665 users online," or, "Visitor number 81434,") resulted in changes in the concurrent crawl. Because the content of these pages is driven by the visiting client, it is important to recognize that many of the changes detected by a crawler may be insubstantial as they are driven by the crawler itself. A simple pattern for these automated ("uninteresting") changes could be easily learned by automated analysis of the textual differences between two instantly-changing documents.

3.2.4 Summarizing a Pages Change over Time

Thus far the analysis has centered on changes between two *successive* versions of a page. Such analysis does not capture a pages change trajectory over a period a time or the more detailed content of that change. A page that has one restricted area of change that updates

every time it is crawled (e.g., an advertising block) will show very similar amounts of change between successive versions as a blog that adds a new entry every crawl. But, over time, the content of the page with the advertising block changes very little, while the content of the blog changes a lot.

3.2.5 Change Curves

A change curve represents the amount of textual change (as measured by the Dice coefficient) from a fixed point in the documents history. For each page, a random set of up to n starting points is selected ($n = 5$ in the analyses reported below, with a bias to the first week of samples). D_t is defined as the Web page content at time t, and D_r^1 to be the content at the first randomly selected time. Content, for this analysis, is defined to be the page stripped of markup. The value of the change curve at each time point, t, is calculated as the average Dice coefficient from each of the randomly selected starting points to the Web page content t time steps in future.

Change curves summarize a pages evolution over time. Several example curves can be seen in Figure 3.8. Notably, most documents change rapidly from the initial starting point as content shifts off the page or is changed during the initial hours. For example, in a news homepage, specific news stories may move off the page at a certain rate as new articles are added. This causes a rapid falloff in the Dice coefficient. At some point, the curve levels off, as the similarity of all subsequent versions to the initial version is approximately equal. This creates a change curve with a "hockey stick" form (see Figure 3.8).

Note that not all versions of the page after the curve levels off are the same. It is rather their similarity to the original starting point that is the same. In the news example, at the inflection point or *knot*, where the curve levels off, all stories that were present in the initial sample have moved off the page. Every page following the inflection point has different stories from the starting point, and this represents the constant Dice coefficient in the level portion. The part of the site that remains similar to the initial sample is the text that always resides on the page, including navigational content and terms that appear regularly in all stories. In general, the textual content past the inflection point that is similar to

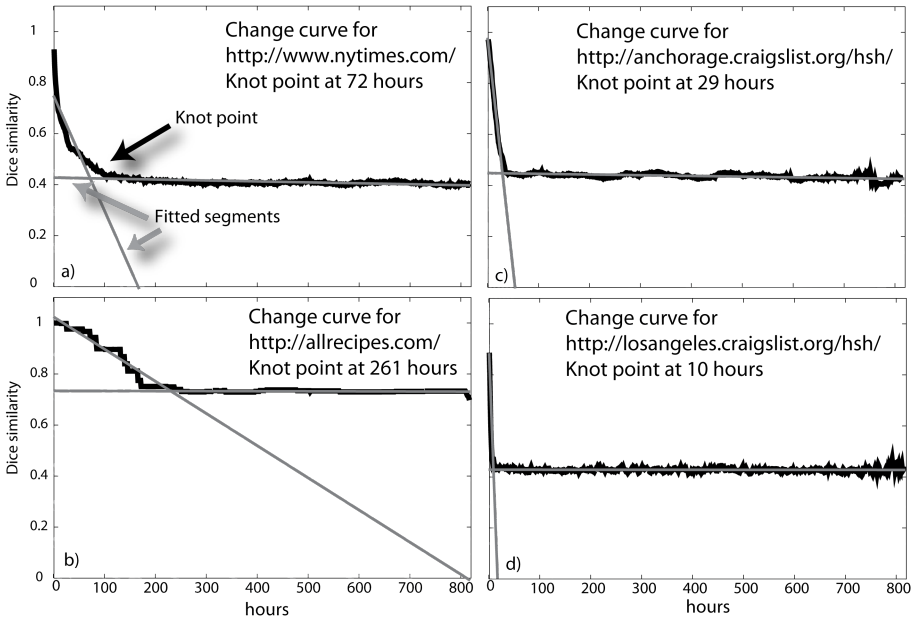


Figure 3.8: Several example change curves depicting Dice coefficients over time.

the starting point is template information and a rough representation of the underlying language model of the page.

Manual analysis of the change curves identified three main types: knotted (made up of two piecewise linear segments, the "hockey stick" shape), flat (one unchanging, zero slope line), and sloped (one sloped line with no obvious knot).

3.2.6 Identifying the Knot

To compare the change curves of different Web pages to each other, curves were modeled using the characteristic hockey stick shape. This is achieved by identifying the curves knot and fitting two linear regressions to the curve, one up to the knot, and the other following it. Piecewise linear regression is a well known problem which requires either knowing the knot location a priori or determining it through a grid search with a well defined objective function. To find the knot point efficiently, a practical heuristic scheme is used.

The algorithm works by first fitting one linear segment to the full change curve. The

leftmost intercept of the fit line to the change curve becomes the initial guess for the knot location. A segment is then fit to the portion of the change curve occurring before the knot. The knot point is incremented as long as the mean-squared error of the first segment does not increase. Figure 3.8 shows the fitted segments for different pages. In situations when no knot point appears within the sampled time period (i.e., a knot point at 0 hours or the two segments have nearly identical slopes), one segment is fitted which is either continuously sloped or flat (an unchanging page). Of course, sloped pages may eventually level off in a hockey stick pattern, but there is not sufficient evidence to conclude when, or even if, this might happen. This process is equivalent to a grid search where some initial guess is made for the knot point location and the knot is then moved iteratively in one direction while the fit improves.

The combination of knot point and characteristic regression information (slopes and intercepts) provides a representation of the data which can be used to automatically classify the curves as knotted, flat, or sloped. 200 curves were randomly selected from the dataset and then manually labeled. Curves were split into training and test sets (70/30 split). Using a classifier based on Additive Logistic Regression [65], 93% of the test set were correctly classified. The classifier is somewhat conservative in deciding on the knotted category, resulting in very few false positives but a number of false negatives which are categorized as sloped. In the end, the algorithm classified 819 pages (of 36,767) as flat (2%), 10462 as sloped (28%), and 25,486 as knotted (70%). For all pages with knot points, the mean time is 145 hours (median time 92 hours), and the mean Dice coefficient is 0.74 (median Dice 0.80).

Table 3.1 shows the average knot point for pages broken down in a number of different ways, including by top-level domain, URL depth, and category. The change time (in hours) reflects how long it takes to reach steady state content, and the Dice coefficient represents the amount of overlap at that time (the x/y coordinates of the knot point). All of the differences in change intervals that are greater than four hours are significant, except that .gov domain is only significantly different from .edu. All differences in Dice coefficient greater than 0.01 are significant.

There are a number of interesting properties of these change curves:

1. *Knot location*: Different pages (e.g., Figure 3.8b and 3.8d) level off at different Dice values. This level represents the eventual (dis)similarity of the page once content has "decayed" off. Allrecipes, with much more navigation and structure retains more content over time than the bare Craigslist page that primarily consists of new ads. The time of the knot point (i.e., its x-coordinate) reflects the rate at which the most quickly changing data, does change. Note the difference between Figures 3.8c and 3.8d that have the same eventual Dice value, but different knot locations in time. The Los Angeles Craigslist page (Figure 3.8d), with many more new home products being sold, has a knot point at 10 hours versus Anchorage's 29 (3.8c).

2. *Unique visitors*: Unlike inter-version means, there is no statistical difference in where the knot point falls as a function of unique visitors. This is consistent with the fact that while popular pages change more often, they change less when they do, and thus require the same amount of time to "stabilize" as less popular pages.

3. *URL Depth*: The deeper the page is in the page hierarchy the further the knot point, potentially indicating that content on pages deep within a site "decay" at a slower rate.

4. *Category*: Perhaps unsurprisingly, News and Sports pages have an earlier knot point as content in these pages is likely to be replaced quickly. Industry/trade pages, including corporate home pages, display a much more gradual rate of content decay before reaching the knot point.

3.3 Term-Level Change

The above analysis explores how page content changes across an entire Web page. This type of page evolution analysis determines the rate at which the page changes as a whole, but not specifically how the textual content is evolving. To understand page change at this finer resolution, it is worth briefly considering how content change at the term level.

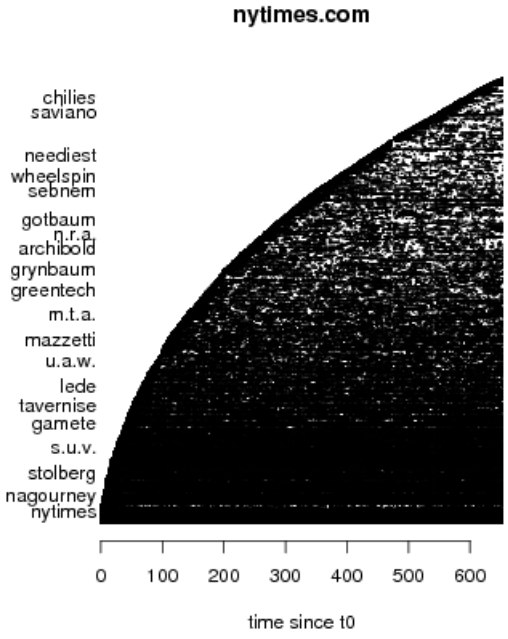


Figure 3.9: Longevity plot for the New York Times page. Time is shown along the x-axis, and terms are shown as horizontal lines on the y-axis (some terms are selected at random and displayed on the y-axis). A point is drawn when a term occurs in the document at a given time. Terms are ordered (from bottom to top) by their first occurrence in the page, and then by their longevity (amount of time on the page). Thus, terms on the bottom appear the earliest and survive the longest (though we note that they may leave and return).

Figures 3.9-3.12 shows several term longevity plots which visualize the dynamics of vocabulary change over time. Time is shown along the x-axis, and terms are shown as horizontal lines on the y-axis. A point is drawn when a term occurs in the document at a given time. Terms are ordered (from bottom to top) by their first occurrence in the page, and then by their longevity. The change curves in Figure 3.8 can be viewed as a summary of the term longevity plots in Figures 3.9-3.12. Change curves can be generated by selecting a reference vertical slice in a longevity plot and measuring the overlap of terms in that slice with those in subsequent slices.

There is a qualitative difference between the terms in the bottom of the longevity plot and those in the top. The terms at the bottom characterize aspects such as the central topic

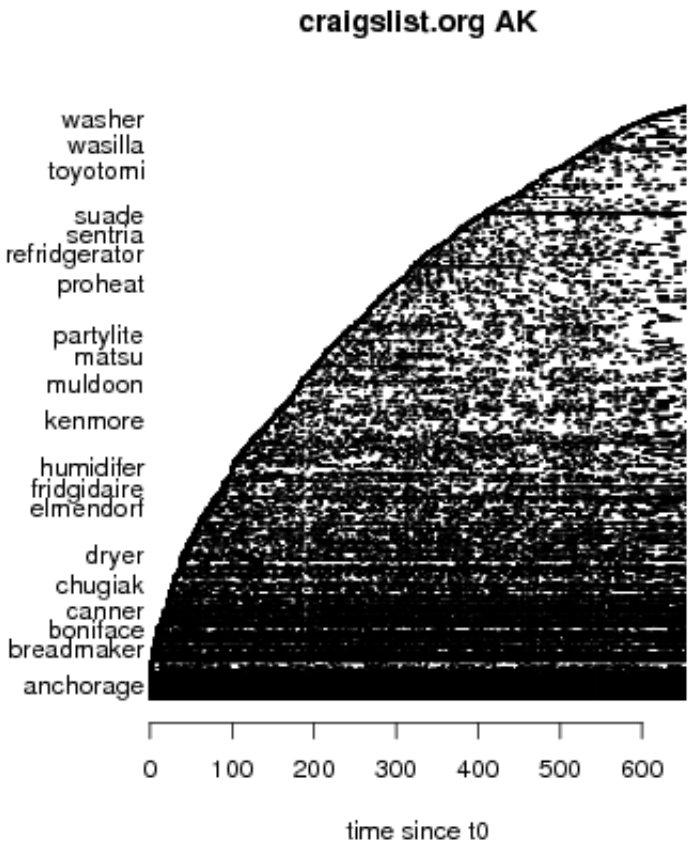


Figure 3.10: Longevity plot for the Craigslist Anchorage, Alaska home goods page

of the page, as well as navigational elements, while those towards the top may be seasonal or related to a small niche of the page's central topic.

A more quantitative characterization of these differences can be determined through the calculation of *divergence* and *staying power*. Staying power, $\sigma(w, D)$, is defined as the probability that a term, w, will remain on the Web page, D:

$$\sigma(w, D) \approx P(t)P(\alpha)P(w|D_t, D_{t+\alpha})$$

where α is the interval between times; $t \in [0 \ldots T]$ indexes timestamps; D_t is a document at time t. Stated another way, this is the likelihood of observing a word in document D at

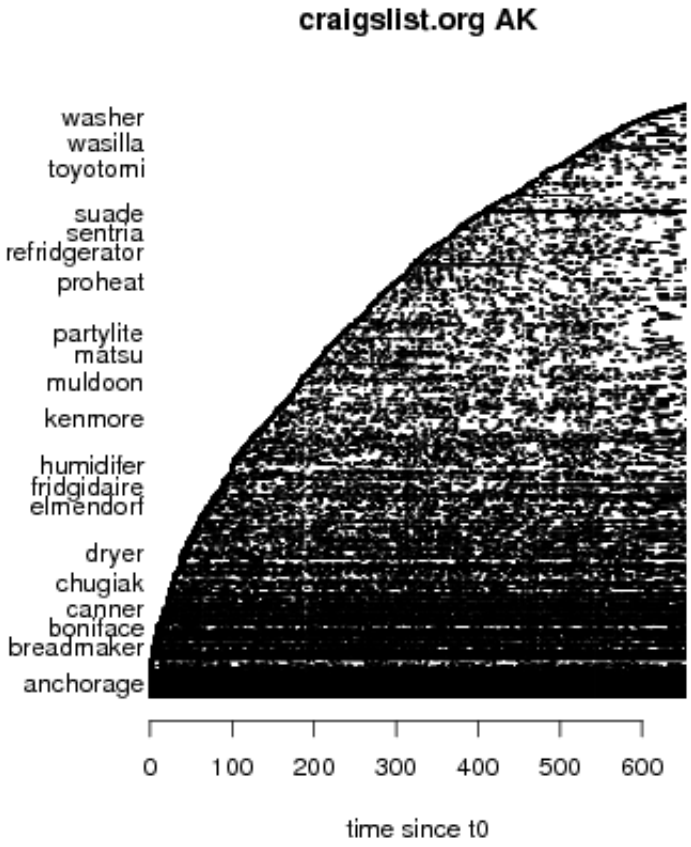


Figure 3.11: Longevity plot for the Craigslist Los Angeles, California home goods page

two different timestamps, t and $t + \alpha$, where $P(t)$ and $P(\alpha)$ are sampled uniformly. This may be calculated as:

$$P(wD) = \sum_{t=0}^{T-1} \sum_{\alpha=1}^{T-t} \frac{1}{T(T-\alpha)} I(w \in D_t \wedge \in D_{t+\alpha})$$

where $I(.)$ is an indicator function used in testing if the word exists in both times. Figure 3.13, shows the staying power distribution for several sample pages. Note the clear bi-modal distribution, with most of the vocabulary clustered at the ends. Terms with high staying power are likely to be in multiple crawls of the page, and those with low staying power are unlikely to occur in multiple crawls of the page.

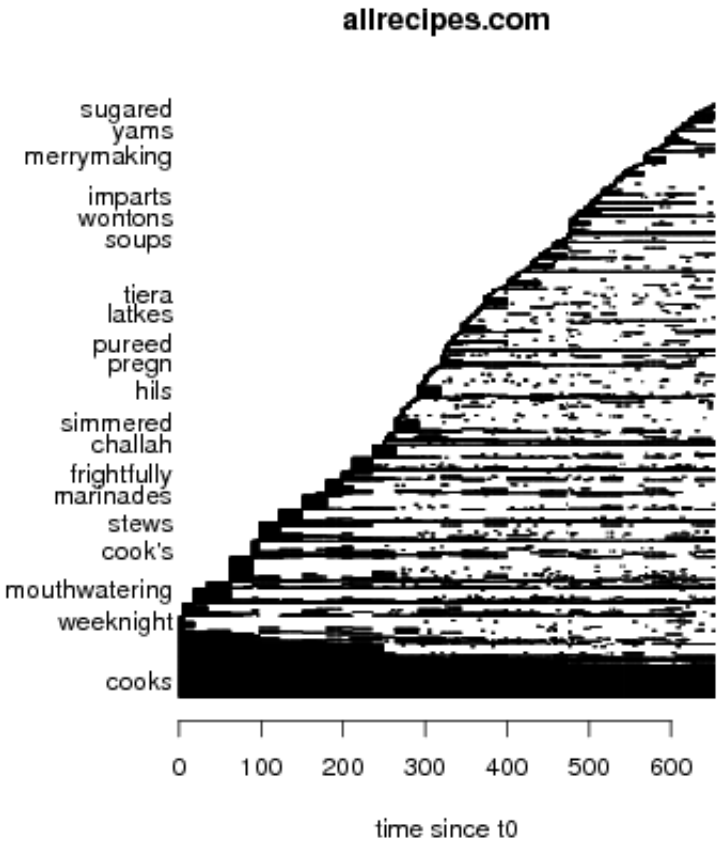


Figure 3.12: Longevity plot for the AllRecipes page

Unfortunately, staying power captures terms that may not actually be "important." For a page such as Allrecipes (see Table 3.3), terms with low staying power include *bbq*, *salads*, *pork* (recipe related words that vanished as the summer became fall), terms with high staying power include *cook*, *ingredient*, and *deserts*, but also *search* and *home*. High staying power terms contain terms that are unique to the Web page as well as terms that relate to navigation information and exist across a diverse collection. In order to better distinguish between these different types of terms a different type of measure is needed.

Divergence attempts to distinguish the set of terms that are more informative of the document's central topic. In non-temporal contexts, divergence has been used to measure

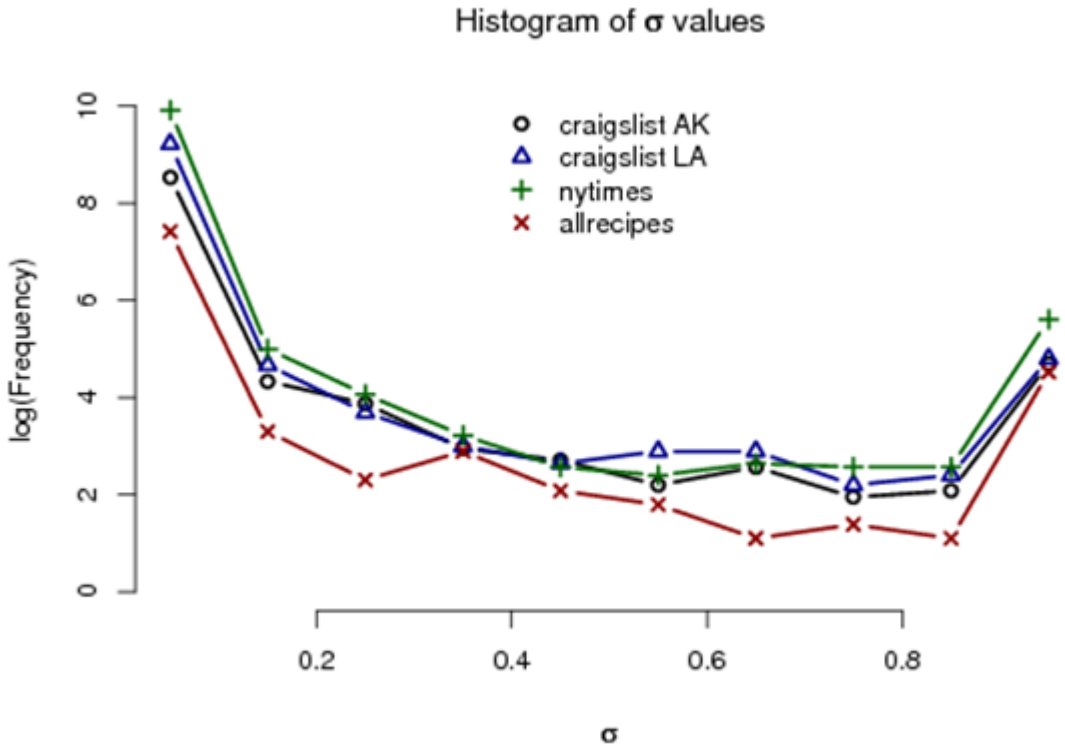


Figure 3.13: Histogram of staying power (σ) values for four web pages.

how strongly a subset of documents is distinguished from a collection, and has been applied to the task of query difficulty prediction [52]. In the traditional setting, the K-L divergence is used to as a measure of distance between two distributions. In this case, the divergence is used to measure the coherence between the set of retrieved documents and the collection as a whole (where documents are treated as a distribution of terms). An application of a similar measure in the context of a single document measures how terms in the document distinguish the language model of that document from the entire collection. This is calculated as:

$$Div(w, D) = P(wD)log\frac{P(w|D)}{P(w|C)}$$

Where w, D, C are the word, document, and collection respectively. The probabilities are computed as un-smoothed maximum likelihood estimates:

$$P(w|D) = \frac{1}{T}\sum_{t=0}^{T}\frac{tf_{w;D_T}}{|D_t|}$$

$$P(w|C) = \frac{1}{T}\sum_{t=0}^{T}\frac{ctf_{w;C_T}}{|C_t|}$$

Where T is the total length of time, tf and ctf refer to document and collection term frequency respectively, D_t and C_t are the document and collection at time t, and $|D|$ and $|C|$ refer to the length of the document and collection respectively. Divergence is analogous to a probabilistic form of TF-IDF weighting, favoring terms with a high likelihood of occurrence in the document and a low overall likelihood of occurrence in the collection. The divergence can be measured for the document over all crawled instances as shown in the equation above (divergence), or for a single crawled instance (divergence at t) by just taking a single term of the summations above.

Table 3.3: Sample terms from Allrecipes.com. Recall that a High D indicates terms that are unique to the page and High σ terms are those that persist for extended periods. Low D/Low σ terms are somewhat rare and generally uninformative.

High D/Low σ	**High D/High σ**	**Low D/High σ**
pork	cooks	home
salads	cookbooks	you
sandwiches	ingredient	search
bbq	desserts	free
salad	rachael	your

Table 3.3 shows several sets of terms for Allrecipes.com. In the left columns, terms with a high divergence at t value are shown, for $t = 0$, the first crawl in this collection. In bold are terms with a low staying power ($\sigma < 0.2$). These terms represent ephemeral vocabulary which are descriptive of the page at the time (here $t = 0$) but are not descriptive on an ongoing basis. In the right columns, we show terms with high staying power in the documents ($\sigma > 0.8$) broken down by whether they have high or low divergence from the collection model. In this example, the divergence measure can separate terms such as function words which are always present in the documents (low divergence) from those

that are more descriptive of the document's content (high divergence). Note that certain words with low staying power but high divergence includes *bbq* and *salad* and *cool*. These terms have to do with *seasonal* recipes and while they have low staying power *within* the study period (done in the summer), they likely emerge again every year. This argues that a third measure–one that captures periodicity–might be additionally useful. The analyses shown in Table 3.3 suggest that measures such as staying power and divergence are useful in identifying terms that are both characteristic and discriminating for the document as it evolves over time.

3.4 Structural Change

This far, Web pages have been evaluated as raw, unstructured text. The reality is that structural changes represent an important way a page can change. A page with exactly the same textual content as a previous version can still be very different because that content is organized and presented in a different way.

In considering structural change, it should be possible to compare the tree structure of each version of the Web page to one or many other instances. Various efforts have attempted to address this difficult problem (extensively surveyed by Grandi [71]). The tree alignment problem is complex to implement, expensive to calculate, and usually only treats pairs of trees. Ideally, a thousand or more instances of the same page (i.e., different versions) could be analyzed rapidly. One test for change is that of *DOM lifespan*: the length of time various DOM elements (described by XPath) exist, and potential changes within them.

Appendix A describes a simple algorithm that serializes an HTML document in such a way that different forms of lexicographic sorting and aggregation can answer these questions. A limitation of this approach is that it is unable to detect the amount of change in content directly. We return to a possible solution below. But despite its limitations, it has the significant advantage of being implementable as a basic map-reduce style problem and scaled.

3.4.1 DOM Element Lifespan

Using the mechanism described in Appendix A it is possible to calculate the lifespan of various DOM elements within the crawled pages over time.

Table 3.4: Percent of persistent DOM elements.

	2 hour	**1 day**	**1 week**	**2 weeks**	**4 weeks**	**5 weeks**
Mean	99.30%	97.40%	91.70%	87.90%	86.40%	84.30%

Table 3.4 presents the survival rates for DOM elements in pages that were successfully parsed (37,160 pages). The survivability of elements is quite high, even after 5 weeks (median survival of 99.8%). The large amount of survivable structure suggests that XPath-based extractions (e.g., [2]) over short periods would continue to function. The algorithm may also identify unstable content and indicate where XPath extractions are inappropriate.

3.5 Using Change Analyses

In Chapter 5, change and revisitation behavior will be coupled to determine *what* are *interesting* changes on a page. In addition to this application, there are a number of others that can make use of the models described in this study.

Because the sample of Web pages in this study were selected based on how people use the Web pages, the results also have implications beyond crawling strategies for tools that support human interaction with dynamic Web content, including Web services like search engines and client-side applications like Web browsers or toolbars.

Crawling. Previous studies of Web change have typically been used to inform search engine crawler policies, the idea being that pages that change a lot are more likely to get stale and should be crawled more often. One important observation for Web crawlers in the study above is that some types of change are more meaningful. By looking at what type of content is changing, crawlers can focus on meaningful change. For example, the fast crawl highlights the importance of ignoring change to advertising content or change that occurs

as a result of touching a page (e.g., changes to visit counters or information about page load time).

The analysis of the terms that change in a page suggest some of the static content may be more valuable for characterizing a page, and thus more valuable to the search engine index. Crawlers could be updated to take into account several archived versions of a page to identify the characteristic static component, and then not worry about crawling the page to capture every new, relatively unimportant change. Further identifying those pages for which the dynamic component is important (perhaps because the queries that lead to a page do so via matches with the dynamic component, or because the queries are similarly dynamic) would allow crawlers to focus on those pages in particular.

Ranking. A similar approach could be applied to improve Web search engines. Characteristics of Web pages, such as the knot point, that summarize page-level changes, over time could be used as features for improved ranking. In addition terms may warrant different weight for retrieval based on whether they appear in the static or dynamic portion of a Web page. The characterization of the page in the index could mirror the page's change patterns, and queries could be mapped to the appropriate aspect, either in general (e.g., static content is always weighted higher) or on a per-query basis (e.g., some queries bias towards dynamic content while others bias towards static content). Similar ideas could be used to improve selection of keywords for contextual ads.

3.6 Summary

This chapter has described the dynamic properties content on the Web through a unique dataset. A number of algorithms, analyses, and models generated characterizations of the evolution of Web content. This analysis gives insight into how Web content changes on a finer grain than previous research, both in terms of the time intervals studied (i.e., hourly and sub-hourly crawls instead of weekly or daily crawls) and the detail of change analyzed (in addition to looking at page-level changes, as well as DOM- and term-level changes). Change rates are higher in this behavior-based sample than has been found in previous work on randomly sampled pages, with a large portion of pages changing more than hourly.

Detailed content and structure analyses identify stable and dynamic content within each page.

Chapter 4

Revisitation

This chapter presents the flip side of Dynamic Web content production–content consumption. The most basic measure of content consumption is simply the visitation patterns of users accessing the page. Though such visitation logs do not necessarily identify why an individual is accessing (or re-accessing) a page, they are nonetheless informative about how individuals and groups behave *over time* in relation to specific pages (Figure 4.1).

Revisiting Web pages is common, but an individual's reasons for returning to different pages can be diverse. For example, a person may revisit a shopping site's homepage every couple of weeks to check for sales. That same person may also revisit the site's listing of fantasy books many times in just a few minutes while trying to find a new book.

This chapter describes an analysis of the revisitation patterns of the 612,000 users described in Chapter 3. Thanks to the large quantity of data, it was possible to explore the revisitation patterns for specific Web pages along a number of dimensions not previously feasible. The study revealed that even very similar pages can have very different revisitation patterns and suggests such differences can be due to a number of factors, including a person's intent, page content, and site structure.

Though the log data enables characterizing the differences between revisitation patterns, it is difficult to deduce revisitation intent without a more detailed survey or observational data. For example, someone may revisit a Web page frequently during a short interval of time because they are monitoring changing content contained in the page, or they may do

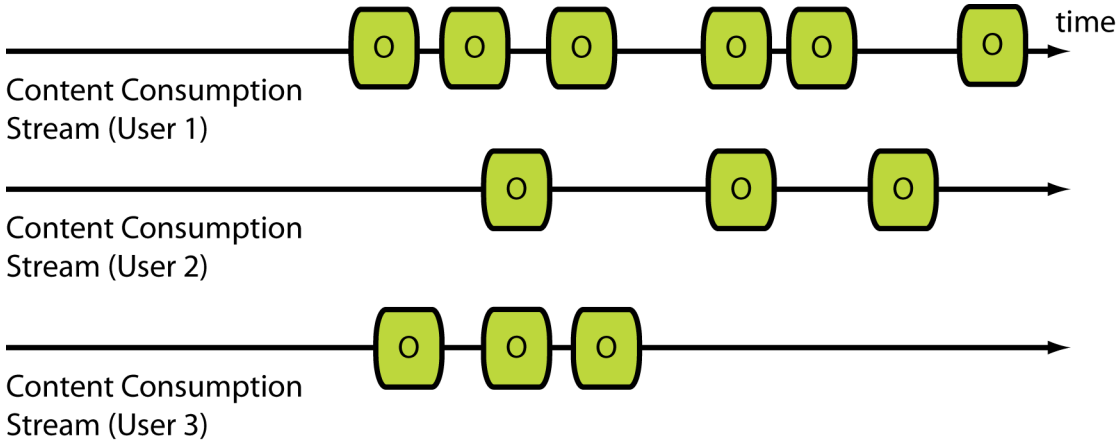


Figure 4.1: Different individuals revisiting the *same* Web page.

so because they are using the site as a hub to navigate to linked pages. For this reason, the original log data was supplemented with a survey in which participants were asked to describe their reasons for revisiting specific pages.

By combining revisitation patterns and survey data, the study revealed four main patterns of revisitation that are characterized by various factors including usage, content, structure, and intent. This analysis can enable Web sites, Web browsers, and search engines to better support revisitation (Section 4.4).

This study distinguishes itself from previous efforts (see Section 8.2) in a number of significant ways. First, to date, this is by far the largest study of Web revisitation behavior. While other studies have observed at most a few hundred participants in their population, the cohort in this work exceeds 612,000 people. Second, while other studies have pursued the longitudinal tracking of a small number of individuals in their complete, unbounded, interactions with the Web, this study attempts to capture the interaction of hundreds of thousands of people with a specific set of Web pages. By broadly sampling pages and people it is possible to understand the characteristics of a page that are associated with specific revisitation behaviors.

A novel aspect of this methodology is that it is possible to find diverse Web pages that are (re-)accessed in many different ways. With a small user population it is difficult to find Web

pages that are not considered popular but are nonetheless visited sufficiently for statistical analysis. That is, the likelihood of two people in a small population both visiting anything but very popular pages is low. With the large user population in this study, it is possible to find multiple access patterns for all but the most unpopular pages. By understanding the variety of revisitation patterns, we can offer design insights to better support the range of observed behaviors.

The study presented here is the result of collaboration with Jaime Teevan and Susan Dumais ([4]).

4.1 Methodology

The three main sources of data, usage information, self-reported intent, and Web page content, are summarized in Table 4.1. The principle dataset used in this study is described in Chapter 3. Recall that this included 612k users and 55k pages. Additionally, a secondary data set was collected through a monitoring application and survey in a smaller population.

4.1.1 Self-Reported Intent

Interaction logs allow researchers observe what people did, and in this particular study led to categorize revisitation in a way that emphasizes observable behaviors. However, the logs do not indicate *why* people behaved the observed way. For this reason, a complimentary user study was conducted to gather information about user intent. In addition to helping better understand revisitation patterns, the results of the user study also serve to relate the behavioral observations to previous taxonomies of revisitation that focus on intent (e.g., monitoring or communication [95]).

Twenty volunteers participated in the study (all daily WWW users; 7 were in their 20's, 9 in their 30's, and four were 40 or older; 19 were male). Participants were employees of Microsoft in a range of roles (developers, researchers, interns, program managers, etc.) Each participant installed software to log Web page visits, which they used for one to two months. The tool recorded visits for the 55,000 URLs in the log study as well as a random subset from the user's cache and Web history. At the end of the logging period, participants

Table 4.1: Summary of the data collected for analysis.

Class	Data type	Collection Method
Usage information		
Bin	# of unique visitors	Log analysis
	Time between visits	
	# of visits per visitor	
Patterns	Revisitation curve	Log analysis, clustering
Session	Previous URL visited	Log analysis of URLs visited prior to page
	Accessed via search?	
Self-reported intent		
Survey	Reason for revisitation	Survey, monitoring
Web page content		
URL	Length	Analysis of URL text
	Domain	
	Text substrings	
Content	Terms	Analysis of content
	ODP Category	SVM classifier
	Genre	Product classifier
Change	# of changes	Regular crawl
Structure	Outlinks	HTML parsing

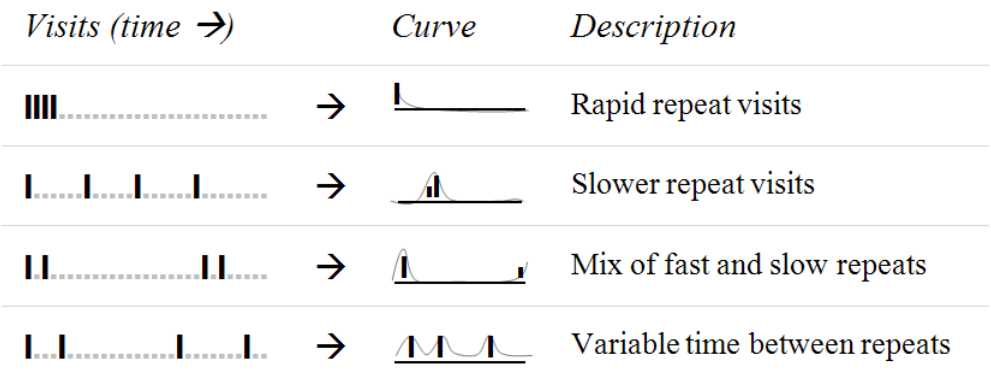


Figure 4.2: Relationship between visits and revisitation curves.

were asked to complete a survey to gather greater detail about ten of the Web pages they had revisited during the observation period. Of the survey responses, 49% (∼80 URLs) overlapped with the 55,000 pages in the log study. The remaining URLs included a number of intranet pages which were used for additional qualitative analysis.

For each Web page in the survey, participants were asked whether they remembered visiting and revisiting the page. If they remembered the page, they were asked to indicate their intent when visiting from a list of options (e.g., to check for new information, to purchase, to communicate, etc.). If they recalled visiting the page more than once, they were further asked to describe how often they visited the page, and whether they visited it at regular intervals.

4.2 Revisitation Curves

4.2.1 Construction

To compare and evaluate revisitation behavior for different URLs, log data was transformed into *revisitation curves*. A revisitation curve is a normalized histogram of inter-visit (i.e., revisit) times for all users visiting a specific Web page, and characterizes the page's revisitation pattern.

Figure 4.2 illustrates the relationship between page visits and revisitation curves. For each row, four page visits are represented as four tall bars along a time line. The revisitation curve is a histogram of the inter-visit times. The x-axis represents the inter-visit time interval, and the y-axis represents a count of the number of visits to the page separated by that interval. The specific density of visits determines the shape of the revisitation curve. For example, the page in the first row shows four visits in rapid succession, and none at longer intervals, so the revisitation curve shows a high number of revisitations in the smallest interval bin. In contrast, visits in the second row are spread out which shifts the peak of the revisitation to the right (corresponding to a higher inter-arrival time bin). Note that the number of visits in each example is the same. That is, the same number of visits per user can result in very different revisitation curves.

Revisitation curves are generated first by calculating all inter-arrival times between consecutive pairs of revisits. Deltas, in minutes, are computed for all users revisiting a specific URL ($\Delta_{minutes} = floor(\Delta_{seconds}/60)$). Because binning by minute results in very sparse data for slow revisits, 16 exponential bins were used to smooth the histograms and to provide a finer grained resolution than the six bins used in URL sampling. Some manual tuning of the bin boundaries was used to generate more descriptive timescales. The boundaries were 1, 5.5, 11, 32, 64 (∼hour), 98, 136, 212, 424, 848, 1696 (∼day), 3392, 6784, 13568 (∼week), 27136 (∼2 weeks), and 55000 minutes (∼month).

Because histograms are count based, URLs that had many more visitors or more revisits per visitor had higher counts. In order to compare revisitation patterns between URLs, each individual curve is normalized by the centroid (i.e. average) of *all* curves. This centroid is depicted in Figure 4.3 (which also illustrates the effects of binning and bin location). This bottom curve shows the normalized histogram using the raw (un-binned) data. The periodicity in the figure corresponds to a 24 hour cycle and is likely due to the daily patterns of Internet usage. If a person has a usual time for Web surfing, this kind of periodic behavior will be seen in revisitation curves (similar to periodicities observed in re-finding behavior [140]). The top portion of the figure shows the smoother centroid curve that is obtained by summing over all points in each of the 16 inter-visit bins, which are shown as vertical lines.

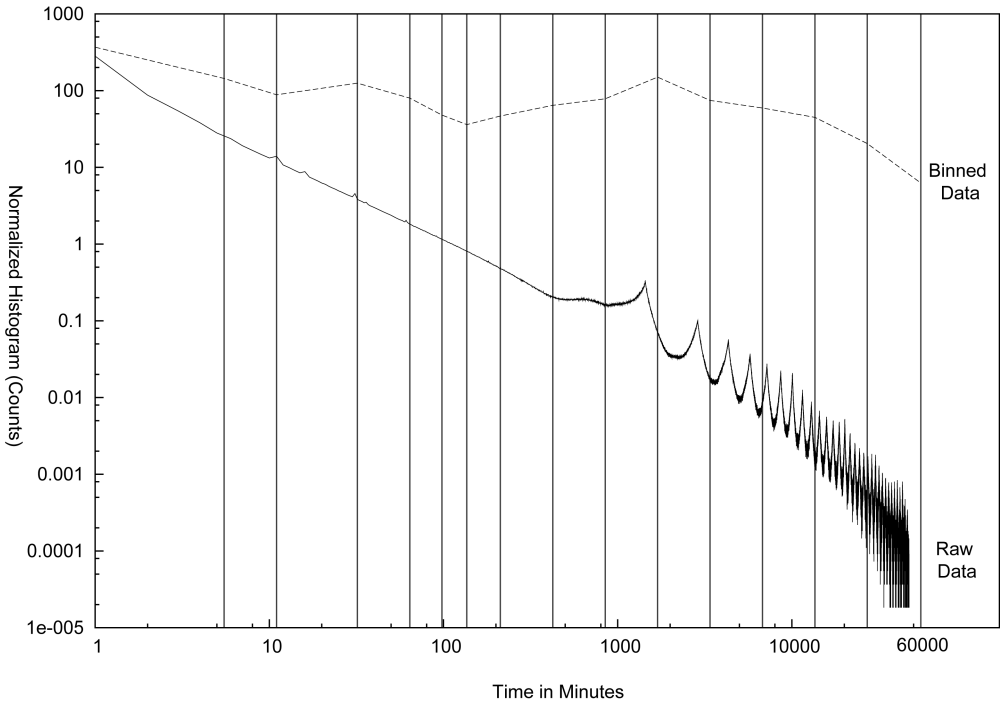


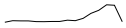
Figure 4.3: The centroid (average) revisitation curve for all pages in the 55,000 sample. The binned data only has one point that corresponds to the sum of points in the raw data (i.e., a point representing the area under the curve).

To complete the normalization: for each URL the un-normalized revisitation count in each bin was divided by the corresponding count in the centroid. For every bin, i:

$$(normalized)revisit - curve_{url}[i] = count_{url}[i]/centroid[i]$$

Roughly speaking, the normalized revisitation curve for each URL represents the percentage over, or under, revisits to that URL compared to the average revisit pattern. Though there are a number of other ways to normalize this type of data (normalize to 0-1 range, subtract out the centroid, etc.) this mechanism works well in practice and allowing simultaneous comparison and grouping of different behavior patterns.

Examples of revisitation curves for two specific URLs are:

-  http://www.amazon.com/ has a revisit curve that peaks towards the right,

indicating that most revisits occur after a longer time period (longer than a day).

- http://www.cnn.com/ displays a peak on the left, perhaps driven by automatic reloads, along with a higher middle region, perhaps due to users checking for latest news.

4.2.2 Grouping

Each revisitation curve is a signature of user behavior in accessing a given Web page. Given this representation of behavior it is natural to ask about the range of different curves. In order to group these curves, the 55k curves were clustered to identify curves that have similar shape and magnitude. A repeated-bisection clustering algorithm [93] was used with a cosine similarity metric and the ratio of intra- to extra- cluster similarity as the objective function. In practice, clusters were found to be fairly stable regardless of the specific clustering or similarity metric. By varying the number of clusters and testing within- and between-cluster similarity, the objective function levels off at around 12 clusters (graphically represented in Table 4.2 and Figure 4.4).

Clusters were grouped and manually labeled and named as shown in Table 4.2. The four groups were named based on general trends observed in each grouping *fast*, *medium*, *slow*, and *hybrid*. Many revisitation patterns fell at the extremes. Five clusters represented primarily fast revisitation patterns, where people revisited the member Web pages many times over a short interval but rarely revisited over longer intervals. On the other hand, four clusters represented slow revisitation patterns, with people revisiting the member pages mostly at intervals of a week or more. Between these two extremes are two groups of clusters. One is a hybrid combination of fast and slow revisitations, and displays a bimodal revisitation pattern. The other type consists of two medium clusters comprised of URLs that are revisited primarily at intervals between an hour and a day. The clusters in this group are less peaked and show more variability in revisitation intervals than the fast or slow groups.

The self-reported revisitation reinforces the selection of the grouping criteria as patterns from the surveys were fairly consistent. This was true not only with the participant's ob-

Table 4.2: Cluster groups and descriptions.

Cluster Group	**Name**	**Shape**	**Description**
Fast (< hour), 23611 pages	F1		Pornography & Spam, Hub & Spoke, Shopping & Reference Web sites, Auto refresh, Fast monitoring
	F2		
	F3		
	F4		
	F5		
Medium (hour to day), 9421 pages	M1		Popular homepages, Communication, .edu domain, Browser homepages
	M2		
Slow (> day), 18422 pages	S1		Entry pages, Weekend activity, Search engines used for revisitation, Child-oriented content, Software updates
	S2		
	S3		
	S4		
Hybrid (3334 pages)	H1		Popular but infrequently used, Entertainment & Hobbies, Combined Fast & Slow

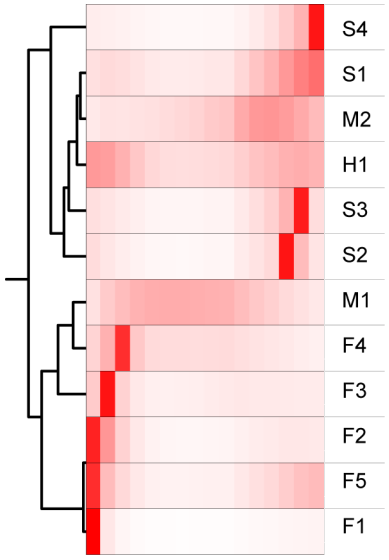


Figure 4.4: The hierarchical relationship of the different clusters plotted by Cluto [93]. Each row is a cluster, a column is a time bin, and darker colors represent more people revisiting in the time represented by the bin (imagine viewing a time series from above).

served page interactions, but with patterns in the aggregate log data. Participants tended to report hourly or daily visits to pages that were clustered as fast or medium-term revisitation, and weekly, monthly or longer revisits those pages with slow revisitation patterns. The self-reported regularity of access decreased as the visitation interval increased. Participants reported visiting early and medium pages at regular intervals, and slow pages at irregular intervals.

4.3 Analysis of Revisitation Patterns

Using the properties described in Table 4.1, different cluster groups were characterized by the pages in them. These findings were further supplemented by a manual examination of the Web pages closest to each cluster's centroid. A summary of the findings can be found in Tables 4.3 and 4.4. Unless noted otherwise, all results are significant as measured using ANOVA/Kruskal-Wallis with post hoc tests, or X^2-tests as appropriate. In most cases, a two-tailed significance level of $p < .0017$ was used. This represents a .01 experiment-wise

error rate using a Bonferroni correction for six dependent pair-wise tests. All differences in Tables 4.3 and 4.4 are significant except for four pair-wise comparisons–the medium and hybrid groups do not differ in the unique visitors or URL length (chars); the fast and hybrid groups do not differ in % domain .com; and the slow and hybrid groups do not differ in % domain .edu. For tests involving terms or topics, 2-tests were used and the significance levels of individual tests were corrected to preserve an experiment-wise error rate of .01.

4.3.1 Fast Group

Pages in the fast group tended to be revisited at intervals of less than an hour. Such sites elicit quick revisitation by forced reloads or shallow exploration and receive almost no long term revisitations. There are a number of reasons why Web pages may display a fast revisitation pattern.

Pornography & Spam: A main category in fast group, and in particular in cluster F_1 , were spam or porn pages. Thirty eight percent of the URLs in F_1 (4746 of 12474) were categorized as Porn (compared to the average cluster which contained 923 pages of porn representing 16.5% of pages in that cluster). Pages in the fast group were also likely to contain words like xxx, sex, photo, and free in the URLs and the text of the pages.

Hub & Spoke: Many of the pages in the fast group appeared to exhibit a hub-and-spoke (also known as "pogosticking") revisitation pattern. For example, a person may start at a list of all products, such as a table of blouses, visit an individual product description pages and then rapidly return to the original page to explore more options. As evidence of this shopping page behavior, fast pages were more likely than pages in other groups to contain the words, buy, catalog, or shop in the URL, and to belong to shopping-related categories. The two most popular categories for this group (seen in Table 4) are Home & Garden and Clothes & Accessories, both of which are comprised mostly of shopping pages.

A hypothesis that visits to hub-and-spoke pages (also called "pogo-sticking") would be particularly likely to be preceded by a visit to a page in the same domain was validated. Seventy seven percent of all revisits in the fast group were from the same domain. This is

Table 4.3: Properties of the clusters, based on analysis of the cluster URLs, content, usage, sessions, and changes (part 1).

	Property type	**Fast** (< **hourly**)	**Medium** (**daily**)	**Slow** (> **daily**)	**Hybrid** (<**hourly**, > **daily**)
Usage information					
Bins	Unique visitors	40.1	254.3	74.9	274.8
	Visits/visitor	5.7	7.4	4	4.9
Session	Previous URL is the same	28.60%	6.80%	7.30%	10.60%
	Previous URL same domain	77.00%	43.80%	56.50%	65.20%
	Accessed via a search	2.90%	4.00%	4.30%	3.40%
Self-reported intent					
Survey	Revisitation reason	Buy something, monitor live content (e.g., sports scores or stock)	Communicate, listen to music, view videos, search, play games	Interact with personal data, view previously viewed information	Visit new content or follow new links, buy something

Table 4.4: Properties of the clusters, based on analysis of the cluster URLs, content, usage, sessions, and changes (part 2).

	Property type	**Fast** (< **hourly**)	**Medium** (**daily**)	**Slow** (> **daily**)	**Hybrid** (<**hourly**, > **daily**)
Web page content					
URL	Length (chars)	48.5	38.3	43.2	38.8
	Length (pieces)	3.2	2.4	2.7	2.6
	Domain .com	77.90%	72.30%	75.50%	77.80%
	Domain .edu	1.10%	4.40%	1.70%	1.60%
	Characteristic substrings	buy, shop, photo	mail, bank	money, weather	game, music
Content	Characteristic terms	price, gallery, auction	news, search, home	pictures, movie, table	change, updated, songs
	Distinguishing topics	House & Garden	Finance & Investment	Business & Finance	Video Games
		Clothes & Accessories	World Cultures	Soccer	Music
		Porn	Astrology & Psychic	Movies	Internet and the Web
Change	Number of changes	222	315.2	283.3	344.6
Structure	Outlinks (total/unique)	74/58	86/69	85/69	112/92
	% Outlinks to same site	75.90%	67.10%	72.50%	71.20%

significantly more than the percent of visits from the same domain to pages in the medium (43.8%) or slow (56.5%) groups. Additionally, the fast group had the highest number of links on the page pointing back at pages in the same site (87%). The longer URLs for pages in this group is also consistent with pages reached by internal site navigation.

Shopping & Reference Web sites: Not only specific Web pages but entire Web sites may display an over-representation of URLs in the fast group. These include a number of shopping/classifieds Web sites (e.g. Craigslist and Kelly's Blue Book) as well as reference pages. For example, pages belonging to Wikipedia and the Internet Movie Database (IMDB) appeared more often in this group (though the homepages themselves were in the slow group). One may hypothesize that certain information needs, such as specific encyclopedia details, are localized temporally. User satisfying such needs may revisit frequently for the current information need, but do not generally return again for an extended period of time.

Auto refresh: Another reason for frequent fast revisits was that pages were auto-refreshing. For example, CNN's homepage causes a browser refreshes every 30 minutes, Fox New's homepage every 10, Bloomberg every 5, and Sports Illustrated refreshes game scores every 60 seconds. In particular, pages in clusters F_3  and F_4 , with peaks at five and ten minutes respectively, may be pages where much of the revisitation occurs from auto-refresh. Both clusters display an over-representation of the terms meta refresh and window.location corresponding to the HTML and Javascript reload mechanisms. Further supporting this hypothesis is the fact that the preceding pages from these clusters are exactly the same 29% of the time, compared with 12% of the time for pages in the medium group and 10% of the time for pages in the slow group.

Fast monitoring: The user survey revealed that monitoring applications in this domain tended to be for quickly changing information that required attention for a short period. For example, a page with local, up-to-the-minute traffic conditions was one such page.

4.3.2 Medium Group 

Pages that fell in the medium-term revisitation group were visited hourly (M_1 ) or daily (M_2 ). Pages revisited hourly (M_1) appeared to be portals, Web mail,

and chat, as evidenced by keywords in the URL and topic classification. Pages in M_2 were bank pages (bank and credit union URLs were very common), news pages, and sports pages.

Popular homepages: Pages in the medium group were among the most easily recognizable, with many being very popular. The URLs were significantly shorter at a mean of 38.3 characters (versus 48.5 and 43.2 for fast and slow respectively) and a mean directory depth of 2.4 (versus 3.2 and 2.7) indicating that they were more likely to be top level homepages. There were many more unique visitors to these Web sites than to others, with a mean number of 254.3 visitors, compared with 40 for the fast group and 75 for the slow group. Interestingly, however, the hybrid group displayed a similar number of unique visitors.

Consistent with the notion that popular sites appear in the medium group, this group includes pages from search engines (e.g. www.google.com and search.yahoo.com) as well as the homepages for BBC's news site (news.bbc.co.uk) and Google news (news.google.com). Additionally, medium rate revisits are observed for URLs on games Web sites such as Yahoo Games (games.yahoo.com) and Gamesville (www.gamesville.com).

Communication (Web mail and forums): Mail and forum pages are over-represented in this group. This is likely due to the timescale of human-to-human communication. That is, the turnaround in communication between two individuals through a Web mediated system appears to be more on the hourly or daily basis than significantly faster or slower. The width of the revisitation curve for this group indicates that there is considerable variability (both across and within individuals) in revisitation intervals. Survey respondents further indicated that communication was a reason to access a number of Web pages in this group.

.edu domain: The medium group also contained a significantly higher number of educational domain Web pages, with 35% (440 of 1245 .edu pages in the study) of the URLs coming from the .edu domain (versus 28%, 31%, and 4% for the fast, slow, and hybrid groups respectively). The high representation of educational pages may reflect the fact that student populations are particularly likely to re-access Web content at daily or weekly periods.

Browser homepages: According to the survey data, most browser homepages fell in this group, with none falling in the fast group, and few in the slow group. This is consistent with the fact that portals and search engines, both frequently used as browser homepages,

appear in the medium group.

4.3.3 Slow Group 

Pages with slow revisitation patterns were visited on average at intervals longer than a day. These pages most likely represented people's long term interests and infrequent information needs.

Entry pages: Many pages appeared to be product home pages or the entry pages for specialty search engines (e.g. for travel, jobs, or cars). These pages were not used in a hub-and-spoke manner like the pages in the fast group, but were perhaps used to get to such pages.

Weekend activity: Cluster S_2 exhibits a peak at a week. One hypothesis is that much of the activity in this cluster is related to weekend activity. For example, the page http://www.fetenet.com, which is close to the cluster's centroid, contains event listings. The Movies category is the most distinguishing category for this cluster, followed by the Religion category, both of which further suggest weekend activity.

Search-engines used for revisitation: One trend noted across groups was that as the revisitation period increased, the pages were more likely to be accessed from a search engine. The URLs are also longer in the slow group than the medium group, which suggests that direct access typing the URL was less likely for these pages.

Child-oriented content: Interestingly, many pages for children's Web sites are over-represented in the slow group. This includes pages for the Cartoon Network (www.cartoon-network.com), the Barbie Web site (barbie.everythinggirl.com), Nickelodeon (www.nick.com) and Disney (disney.go.com). This finding may be due to limited availability of Internet access for children who may only have access at certain times of the day or days of the week. Additionally, many Web sites are tied to TV programs which have weekly periodicities. Children's Web sites also display hybrid behavior which appears to be due to multiple functions of the Web site.

Software updates: The survey data also revealed pages in the slow group were used to download software updates. For examples, pages with update in the URL or content

included application update sites such as Microsoft Windows Update and McAfee Update.

4.3.4 Hybrid Group

URLs in the hybrid group bore similarities to several other groups, as can be seen in Table 4. Much of the navigation behavior was similar to what was observed for the fast group (e.g., hub-and-spoke), but the hybrid pages appeared to be of higher quality and receive more long-term revisitations. Like the medium group, hybrid pages had short URLs and a large number of unique visitors (274).

Popular, but infrequently used, site homepages: It is interesting that the most popular Web pages, in terms of unique visitors, fall into two different behavior patterns–medium and hybrid groups. Whereas the pages occurring in medium group indicate a fairly constant need, hybrid interactions reflect a rarer need that nonetheless requires many page revisits for the individual need to be met. For example, a page such as Automart's homepage (www.automart.com) provides a front end interface to search for new sales listings, in this case for cars. A user may visit the page, search for updated or new listings and check many listings in any given visit. The homepages for various classifieds listing service, such as local Craigslist homepages, fall into this group as well. Both hybrid and medium group's pages display a higher than expected number of "login[s]" in the text indicating both types are transactional or personalized in some way.

According to the survey participants, the amount of meaningful change they expected to the page decreased as the revisitation interval increased. Visits to hybrid pages were particularly likely to be to find new information (on a page that had previously been visited). Hybrid pages also displayed the greatest number of changes in content. A hybrid behavior might be a mechanism for catching-up with changed content. Pages in the medium group displayed the second highest number of changes and indicate a potentially more regular monitoring activity.

Entertainment & Hobbies: Hybrid behavior is likely observed in situations when an activity requires hub-and-spoke movement around the page, but the activity itself is somewhat infrequent. For example certain shopping related pages, such as shopping carts, exist

in this group. High-level auction pages are also in this group (e.g. computers.ebay.com, home.ebay.com, etc.). The difference between auction related pages, such as eBay, and other shopping destinations might be the requirement of monitoring ever-changing auctions, successive bids, and long-term interest in collectibles.

Another over-represented category within the hybrid group are games pages (as indicated by the page text and by the page categories). It is interesting–though perhaps unsurprising–that games are played rarely but repeatedly.

Combined Fast & Slow: Unlike the medium group, the hybrid group received significantly fewer mean visits per visitor (4.9 v. 7.4). This is consistent with the observation that most URLs in the medium group are attached to constant, more generic, information needs (e.g. communication, news, etc.) whereas page in the hybrid group represent more easily satisfied needs (e.g. games, music, or product purchase) and are more likely a combination of slow (rare information need) and fast (high session revisits) .

4.3.5 Web Sites Across Groups

Because of large scale sampling of URL data it is possible to also look at how pages within a Web site are distributed across groups. This analysis generates confirmation of the findings described above.

As mentioned earlier, there is an over-representation of URLs for shopping and reference Web sites in the fast group (e.g., Craigslist's pages from Seattle and New York, seattle.craigslist.com and newyork.craigslist.com, Ikea at www.ikea.com, Kelly's Blue Book, for pricing used cars, www.kbb.com, and Wikipedia at www.wikipedia.org). An interesting characteristic of such sites is that while a vast majority of the pages deep inside the website were in the fast group, their top-level homepages (e.g., http://www.ikea.com/ and http://www.wikipedia.org/) fell into the slow group. The homepage visits probably represent the occasional need people have for the type of information (Ikea furniture, encyclopedic references, or used car pricing), while the deeper pages that fall in the fast group probably represent individual's behavior when satisfying the need. This suggests that while some Web pages themselves may fall into fast or slow groups, the Web sites may display hybrid

or slow behavior.

In general, Web sites that serve a single consistent role tend to be present in only one group. For example, BlinkYou, a predominantly fast site at www.blinkyou.com, provides widgets to embed in MySpace homepages and MSN Weather, a generally slow site, serves weather at weather.msn.com. Those that serve multiple roles have URLs that fall more equally into all groups. For example, EsMas, www.esmas.com, a Spanish language portal and media site that provides shopping, classifieds, news, galleries, and many other features has a fairly even split across groups (with no significant over-representation in any group).

4.4 Design Implications

4.4.1 Web Browser Design

Previous studies of the revisitation patterns of individual users have led to numerous browser enhancements [17, 92, 94, 120, 147]. In one study, Cockburn et al. [50] explored presenting a user's history as a series of Web page thumbnails enhanced with information about the user's page visit frequency, and allowed the user to group thumbnails to visualize hub-and-spoke revisitation patterns. Given the results presented here, it is likely that there may be value in providing awareness of, and grouping by, a broader range of revisitation patterns. For example, users may want to quickly sort previously visited pages into groups corresponding to a working stack (recently accessed fast pages), a frequent stack (medium and hybrid pages), and a searchable stack (slow pages).

Several history mechanisms have tried to predict whether people will access a Web page based on overall measures of last access time and frequency of access. It may be possible to better predict the future utility of a URL by taking into account the page's revisitation pattern. For example, if a page is visited weekly and the user has not visited it in almost seven days, it is very likely to be revisited soon. On the other hand, a page the user visits daily but that was just visited is unlikely to be visited in the immediate future.

More accurate predictions of the future utility of a Web page can be used in many ways. The color of the page's URL could change to indicate increased potential interest, or the address bar's completion drop-down list could favor URLs that are likely to be accessed.

Alternatively, such pages could be proactively recommended and the content pre-fetched [25] for efficiency in browsing or indexing.

In order for a Web browser to take advantage of revisitation patterns for a page or site, aggregate visitation data could be made available by the site itself. If this is not feasible, it may be possible to predict the page's revisitation pattern by classifying it using non-behavioral features like text and link structure. Initial explorations indicate this approach is promising. Personalized revisitation patterns could further be combined with aggregate observations or predictions.

4.4.2 Search Engine Implications

Revisitation analysis also has a number of implications for search engine design and in particular to the related issue of re-finding. Prior research has demonstrated re-finding behavior is prevalent [140, 152] in search engine use. This analysis, which further demonstrates a relationship between search and specific kinds of revisitation behavior, suggests several ways search engines can support re-finding.

For repeat searches, there is a tension between displaying new results and preserving previously viewed results [151]. It may be particularly important for search engines to provide consistent results for queries that return many slow and hybrid pages, since these searches are likely to be used to re-find pages found a long time ago. Though consistency can be achieved by keeping the results displayed static, it may be better achieved through personalization where the results an individual is likely to revisit are emphasized.

The revisitation patterns of pages returned for a query can tell the search engine something about the searcher's information need. Depending on the particular task or distribution of results, the search engine could include pages that had consistent or diverse revisitation patterns.

The types of pages a person is interested in may also suggest how receptive that person is to new information (e.g. suggestions of related content or advertisements). For example, if a query returns results that are generally in the fast group, this could indicate that the user is looking for something new and may be particularly responsive to the suggestion of

relevant content. On the other hand, if the results are primarily in the medium or slow revisit groups, the user may be more likely to have a specific intent and not respond to suggestions. Content that is somewhat orthogonal to the user's objective may be most helpful in these cases by appealing to different interests.

Search engines may also benefit by taking into account users revisitation behaviors when crawling the Web. Assuming changes to page content are correlated with people's aggregate revisitation patterns, the revisitation patterns could indicate the optimal rate for re-crawling and re-indexing different Web sites.

4.4.3 Web Site Design

Just as search engines may want to make it easy to find slow and hybrid pages, Web site designers may want to ensure that such pages are easy to find. This could be done by creating appropriate keyword lists for indexing, minimizing page change and movement, and providing navigational shortcuts from common search landing pages.

Modeling users in terms of revisitation behavior might allow a Web site designer to apply these results in the design of their site. By taking into account the four potential revisitation styles, in conjunction with the site design and content, information architects can understand and simulate user behavior with their Web sites. This may make it possible for designers to understand potential interaction patterns even before the Web site is launched, and to evaluate a site's success in meeting the design objectives.

Finally, there are likely opportunities for Web site designers to better support monitoring activities. For example, some Web sites create a list of what's new to aid repeat visitors' discovery of new content. This research shows that different pages, even on the same site, can have different revisitation patterns. As a result, it may be in the site's interest to maintain what's new lists at different granularities depending on the specific page.

4.5 Summary

The study presented in this chapter is an exploration of the diverse ways that people revisit Web pages and the reasons behind their actions. The work, which is based on the largest

study to date of Web visitation logs coupled with user surveys and Web content analysis, has allowed us to develop a unique view of people's revisitation behaviors. By analyzing tens of thousands of Web sites, this study has identified 12 different types of revisitation behavior corresponding to four groups that are orthogonal to previous work. The Fast, Medium, Slow, and Hybrid patterns correspond to different forms of consumption behavior. Understanding which category a page falls into has a number of design implications for client applications, Websites, and search engines. In Section 4.4, for example, a different type of browser history was proposed that would take into account revisitation patterns. Pages that fall into the hub-and-spoke pattern can be quickly removed from the history as the page will likely never be revisted. In contrast, having identified that search as frequently a mechanism of access for pages in the slow category, these pages might be indexed. By building these models of consumption, this study opens up a number of doors for the use of behavior and history.

Chapter 5

Resonance

As described in Chapter 3, one of the ambiguities in Dynamic Web analysis–specifically behavioral analysis–is that it is difficult to tell what drives revisitation behavior. Multiple things on a page may be changing independently (Figure 5.1), and it is difficult to conclude what is driving the revisitation.

As described in the previous chapter, revisitation is very common, but people's reasons for revisiting can be diverse (see also [95]). For example, a person may return to the SIGCHI website, pictured in Figure 5.2, to be reminded of the group's officers or to catch up on the latest news. Furthermore, as described in Chapter 3, while most content on the Web does not change, pages that are revisited change frequently. As revisitation is often motivated by monitoring intent [94], some relationship between change and revisitation is to be expected. However, the subtleties of change and revisitation behavior mean the two do not necessarily have a direct correlation. For example, while new content may be added to a blog hourly, individual posts may move slowly down the page and not vanish until 24 hours later when the daily content is archived. Although the page changes hourly, a user monitoring it may only need to revisit daily to catch up on the content.

Pages are often composed of many sub-pieces, each changing at different rates, illustrated in Figure 5.2. Advertisements on a page may change on every visit, while navigation content may almost never change. A particular user's revisitation rate can sometimes indicate the part of the page that user is interested in. Someone who returns to the SIGCHI page

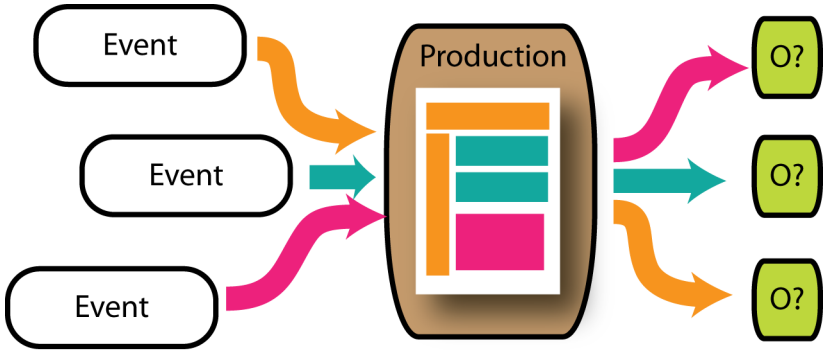


Figure 5.1: Different parts of the page are changing, making it difficult to identify why individuals are visiting the page.

at a frequency similar to news updates may be interested in catching up with the latest news, while someone who returns after a long interval may be interested in revisiting the page's static or navigational content (e.g., "membership"). Aggregate revisitation behavior of many users can indicate the most common purpose of that page.

Web content change can be beneficial to the Web user looking for new information, but can also interfere with the re-finding of previously viewed content [152]. Understanding what a person is interested in when revisiting a page can enable designers to build systems that better satisfy those interests by, for example, highlighting changed content when the change is interesting, actively monitoring changes of particular interest, and providing cached information when changes interfere with re-finding.

This chapter describes how the data about changing content and revisitation behavior, described in Chapters 3 and 4 respectively, relate. The analysis validates a number of hypotheses (e.g., popularity and change are correlated), but also uncovers some surprising results. For example, certain measures of page change (e.g., the amount of change) are not linearly related to measures of revisitation (e.g., inter-arrival times)–a result with consequences to monitoring tools that use these simple measures to watch for page changes to identify events of likely importance. Combining the change curve representation with the revisitation curve helps in identifying the *type* of information targeted by revisitation: static (previously viewed and unchanged), or dynamic (newly available). These results are

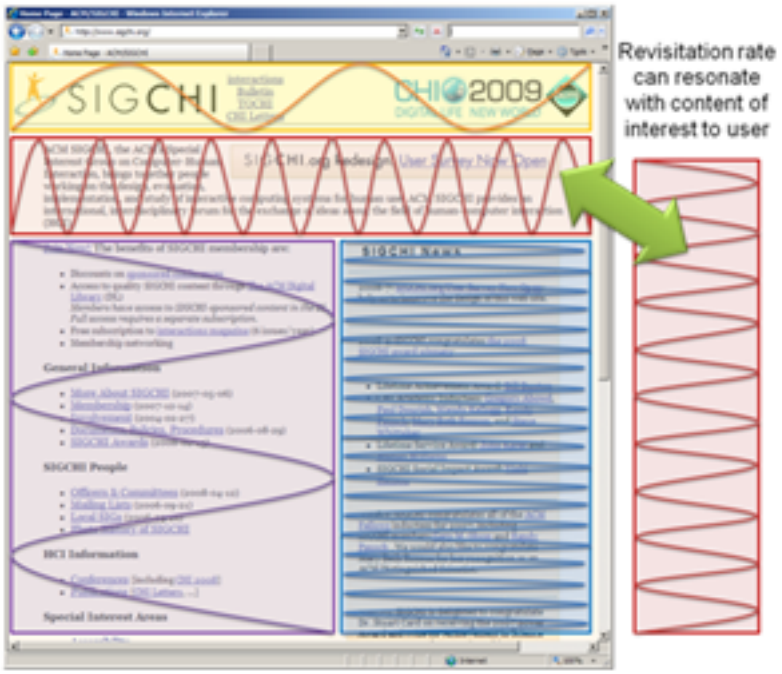


Figure 5.2: Web pages, and their components, change at different rates. People revisit pages for many reasons, and their reasons for revisiting can be affected by content changes. In this paper we show that revisitation rates can often resonate with the rate of change of interesting content.

supported by the the smaller user study described in Section 4.1.1.

Using peak revisitation for a given page in conjunction with fine-grained changes, "interesting" content on pages can be identified. The approach helps in dealing with the inherent ambiguity in temporal data described above. Utilizing an automated algorithm for DOM-level analysis, pages can be broken into components that are more, and less, likely to be of interest. Such analysis has previously been impossible without the use of expensive surveys or other un-scalable research instruments such as eye-trackers.

The ability to identify the target content that is the target of revisitation can greatly enhance Dynamic Web applications including browser, crawler, and search engine design.

Prior work has focused on either the study of change (e.g., [20, 29, 32, 34, 48, 53, 54, 58, 59, 68, 98, 102, 126, 128, 134]) or the study of revisitation (e.g., [16, 51, 80, 94, 95, 127, 150,

144]), but rarely both together. By tracking change and revisitation behavior concurrently, on a very large scale and with very fine granularity, this study offers a novel perspective on content change in revisited pages, revisitation patterns for dynamic and static pages, and the relationship between the two.

The study presented is the result of collaboration with Jaime Teevan and Susan Dumais ([5]).

5.1 Methodology

This study utilizes the original 55k page crawl described in Chapter 3. However, because the revisitation curves determined for these pages may have been different between the initial log sample (August of 2006) and the crawl (May of 2006) a second usage log was obtained (May–June 2007). Using the behavior of 2.3 million English speaking, US-based, non-robot users, 40,817 pages were considered in our final analysis (25% of the 54,788 pages selected were not visited sufficiently during the May/June period and were removed from consideration). Both change curves (see Section 3.2.5) and revisitation curves (see Section 4.2) were calculated for each page.

This study also made use of the survey information collected previously on intent (see Section 4.1.1). For each Web page in the survey, participants were asked whether they remembered visiting and revisiting the page. If they remembered the page, they were asked to indicate their intent when visiting from a list of options (e.g., to check for new information, to purchase, to communicate, etc.). If they recalled visiting the page more than once, they were further asked to describe how often they visited the page, whether they visited it at regular intervals, and how often they expected the page to change. By relating actual page change, people's expectation of change, and their stated intent behind their revisitation, we are able to better explain the behavior we observed.

5.2 Revisitation and Change Trends

The simplest possible explanation of the relationship between revisitation and change is that increased user revisitation behavior correlates positively with increased change. However, such an explanation fails to capture a number of non-obvious connections between behavior and change. For example, pages that change a lot were generally visited more often and were revisited after shorter intervals, and the amount of change from version to version did not correlate to any metrics used in the study.

Table 5.1 summarizes the findings discussed in this section. Both the discussion and the table are broken down by the three measures of revisitation discussed previously (see Section 3.1.1): the number of unique visitors to a page (unique-visitors), the median inter-arrival (i.e., revisit) times for a page (inter-arrival time), and the average and median number of revisits per user for a page (per-user revisits). For each behavior measure, Table 5.1 presents three measures of change (number of changes, time between changes, Dice coefficient for successive changes) and the coordinates of the knot point (time and Dice). The significance of each measure is tested by applying an ANOVA to an ordered binning of the particular metric for overall significance. Bolded results are significant against the previous bin through Tukey's HSD post-hoc.

5.2.1 Unique Visitors and Change

Although static pages can be popular, it is more likely that continued popularity is achieved through some dynamic content and maintenance. As the number of unique visitors increases, the mean number of changes we observed increases, and the time between each successive change decreases–ranging from $\sim$ 138 hours between changes for pages with only 2 unique visitors to $\sim$ 102 hours for those with 36 or more unique visitors.

However, the mean amount of change (as calculated by the Dice coefficient) does not have a similarly distinct trend. The most and least popular pages both have the biggest changes between versions (though the difference between all bins is fairly small, ranging between Dice coefficients of .8 and .83). Thus, while popular sites change more frequently, the same cannot be said about the amount by which they change. This is an indication

Table 5.1: Several measures of change broken down by revisitation bins. The first set of measures represent the mean number of changes for pages in the bin, the mean time between each change, and the mean amount of change. The second set of measures represent the location of the knot point of the change curve (bold items significant).

Revisitation bin		Change Summary			Change Curve Knot	
		Num.	Time	Dice	Time	Dice
Unique visitors	2	**184.93**	**138.23**	**0.8**	146.1	0.76
	3-6	**211.86**	125.78	**0.83**	143.23	0.77
	7-36	**232.44**	**106.86**	0.83	144.51	0.75
	36+	**254.65**	**102.55**	**0.82**	139.25	0.73
Per-user revisits	2	**172.91**	**133.26**	0.82	**157.38**	**0.78**
	3	**200.51**	**119.24**	0.82	154.53	0.77
	4	**234.32**	**109.59**	0.81	**142.98**	**0.74**
	5-6	**269.63**	**94.54**	0.82	**132.13**	**0.71**
	6+	**341.43**	**81.8**	0.81	**116.96**	**0.68**
Inter-arrival time	<1 day	**214.17**	**126.27**	0.82	**145.37**	**0.75**
	1 day+	**245.06**	108.01	0.82	**133.14**	0.76
	1 week+	**289.34**	**91.49**	0.82	133.32	**0.72**
	2 weeks+	**245.66**	88.06	0.82	**141.81**	**0.73**
	4 weeks+	**211.77**	100.44	0.8	**156.69**	**0.74**
Revisit curve	Fast	**182.21**	**150.1**	**0.78**	**147.03**	**0.74**
	Medium	**283.15**	**93.66**	**0.8**	**127.82**	**0.71**
	Slow	**212.66**	**111.58**	**0.81**	**153.82**	**0.75**
	Hybrid	**259.03**	**109.88**	**0.81**	**137.04**	**0.74**

that the amount of change may not be as important as what is changing.

The knot point does not differ significantly with changes to the number of visitors. This may be anticipated as page popularity does not indicate how often revisitation occurs in the specific page only that it occurred more than once. Recall that the knot point is measuring the approximate location when pages have "stabilized"–when every subsequent page is equally (dis)similar to the starting point. If the hypothesis is that individuals will try to synchronize their revisitation behavior to catch content before it "decays" off the page (e.g., at the knot point), this requires requiring the analysis of the number, and interval, of per-user revisitations.

5.2.2 Per-User Revisits and Change

Analysis of the number of times an individual revisits a Web page (what we might call a page's "stickiness") reveals that the pages individuals revisited more times changed more frequently and at shorter intervals. For example, revisited pages visited only twice changed every 133 hours, on average, while those that were revisited 6 or more times changed every 81.8 hours. However, the *amount* of change does not appear to trend in a particular direction when conditioned on per-user revisits, reinforcing the conclusion that the amount of change is not as crucial as the specific information that is changing.

Unlike the popularity category, there is a trending in per-user revisitation when compared to knot location. The more the average user revisits a page, the earlier the knot point and the more different the eventual steady state is to the original page. The implication of this is that users may revisit more often in order to capture content that will vanish from the page. This is consistent with the model that the rate of change before the knot point (the initial, steeper, downward slope) represents the rate at which the dynamic data on the page is "lost."

Further evidence for this is provided by our smaller user study which indicated that people appeared to have a reasonable understanding of Web content change. The knot point for the pages where participants expected meaningful change upon revisiting was sooner than for pages where meaningful change was not expected (60.7 hours vs. 97.0,

$p < 0.05$).

5.2.3 Inter-Arrival Time and Change

Though we might expect similarities between the conclusions drawn for inter-arrival times and the per-user revisits (the two are frequently correlated), this is not always the case. The same number of revisitations can occur very quickly (e.g., 5 revisits in 2 minutes and never again) or very slowly (e.g., 1 revisit per week over 5 weeks). Thus, it is also worth considering the average inter-arrival time.

One might expect that the more rapidly a page changes, the lower the inter-arrival time (the faster the revisits). However, the study revealed that as the inter-arrival time increases, the number of times a page changes increases but *only* for inter-arrival times of less than two weeks. After two weeks this change count goes down again. This is somewhat counter-intuitive as it means that pages with both the high and low inter-arrival times are those with the fewest changes. It is here that we first begin to recognize situations in which revisitation patterns are not necessarily related to frequency of change–an issue we will return when comparing revisitation curves and change curves. The mean time between changes shows a similar bowed pattern, with the longest times for pages that change slowly or rapidly.

This result may also be explained when considering the results described in Chapter 4 in which visitor behavior was described in terms of different types of revisitation (e.g. fast, slow, medium and hybrid). Recalling that the mean inter-arrival time relates to the revisitation peak, a "fast" revisitation curve corresponds to primarily low inter-arrival times. As above, pages in the very fast revisitation category, where people revisit a page a lot during a short period of time but never return after a longer interval, change slowly (see Table 5.1). This would again seem to contradict the hypothesis that revisitation should match change. However, as noted in Chapter 4, 77% of revisitations in the fast category were preceded by a visit to a page from the same domain–indicating a "pogo-stick" browsing behavior (rather than change monitoring). Since users exhibiting this behavior are simply surfing back and forth from the origin page, they are less likely to be interested in monitoring changes on that page in the short term.

Thus, one may refine our hypothesis to exclude those fast revisitations that are more likely the result of page and link structure rather than any kind of monitoring intent. The remaining categories (e.g., medium and slow) do appear consistent with the hypothesis. The number of changes (higher for the faster revisits), the average time between change (lower for faster revisits), the location of the knot point (content vanishes more rapidly for faster revisits) and eventual stable state of the change curve (greater changes for faster revisits) all display significant differences.

With the exception of the very shortest inter-arrival time, the expected relationship given the knot point is observed. The more time it takes for the content to vanish off the page (further knot points) and the less the eventual steady-state (lower Dice), the longer the inter-arrival time is. Again, less revisitation is observed for content that takes more time to change.

5.2.4 Importance of Change

Figure 5.3a shows a different representation of the knot data, tracking inter-arrival times, per-user revisits and knot points simultaneously. The figure shows the number of revisits that occur as a function of the knot point. The data is further stratified into 5 different revisit intervals (i.e., the average revisit in the same day, after 2 days, after 7 days, after 14 days, and after 27 days). As might be expected, the number of revisits with a shorter revisitation interval (e.g., 2 days) is higher than the number for longer revisits (e.g. 27 days). The figure also shows the best fitting linear function (on a log-linear scale) for each of the 5 revisit intervals. In agreement with our prior observations, the linear functions all have negative slopes indicating that when the knot point occurred after a long period (e.g., four weeks), there were fewer revisits than when the knot point occurred early (e.g., one day).

Most interestingly, the linear functions have different slopes. When people revisited a page quickly (e.g., within the same day, top curve), those revisitations are strongly related to how frequently the page changed (people revisited more as the page changed more frequently). On the other hand, when people revisited a page slowly (e.g., after many weeks,

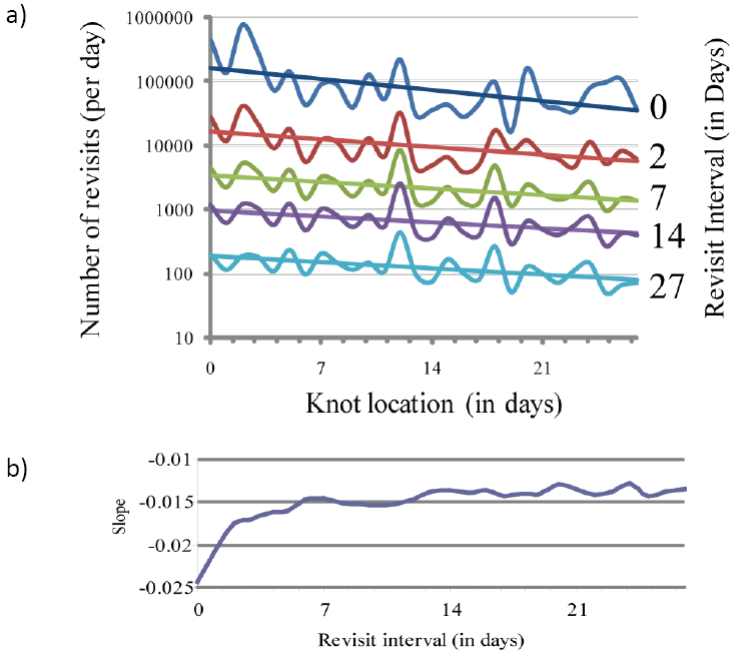


Figure 5.3: A comparison of the average number of revisits as a function of knot location. In the figure, revisits are stratified by different average revisit intervals (e.g., the bottom line is the number of revisits per day as a function of the knot location for pages with an average revisitation rate of 27 days). A linear regression is fit on each individual curve, and in (b) the slope of the different interval regressions is tracked (i.e., the slope of each linear regression in (a) going from 0 to 27).

bottom curves), those revisitations are less related to how frequently the page changed. Or stated another way, the slower the revisits, the less the importance of change. This can be seen further in Figure 5.3b, which plots the slope for the linear function for revisits for the full range of intervals. The flattening of the curves as the revisit interval increases suggests that content change is more closely related to short term revisitation behavior than long term revisitation behavior.

To summarize some of the key findings thus far:

- The more popular the page, the more rapidly it changes (but not vice versa).
- The more times a page is revisited, the more rapidly it changes, and the earlier the

amount of change stabilizes.

- Page revisitation is not directly synchronized to the amount of time between changes or how quickly the change stabilizes. This may be because not all revisitation is motivated by monitoring.

- Quick revisits (e.g., within the same day) are more strongly related to change. Thus, short term revisitation behavior is more closely tied to change, and the relation is non-linear.

5.3 Dynamics and Static Intents

Despite the general tendencies in our data, the precise relation between peaks in revisitation curves and knot points is much more nuanced. If individuals are revisiting with the sole intention of monitoring, we might expect that the bulk of revisits occur before a page's content "turns over". That is, before the page has changed so significantly that data will be lost (i.e., before or around the knot point). However, selecting those URLs with a fixed knot point (e.g., 56-113 hours), the maximum peaks in their revisitation curves are dispersed (see Figure 5.4) for the pages in the medium and slow categories. Note that pages in the fast category which includes, noisy, non-monitoring behaviors (e.g., "pogo-stick") are removed in this example. In this example, 16% of the URLs' revisitation curves peak before the knot, 72% peak after and only 12% peak at the knot point. Thus, while there may be general trending in revisitation–where more changes or earlier knots leads to more revisits–visitors do not appear to be synchronizing to some exact time point. This difference could potentially relate to whether users are interested more in the changing content of the Web page or in the (more) stable content.

Figures 5.5-5.7 shows three different examples relating change to revisits, with revisits peaking before (www.nytimes.com), at the same time (www.woot.com), and after (www.cost-co.com) the knot point. Visitors to the New York Times Web site are typically interested in finding information about current news events and are therefore interested

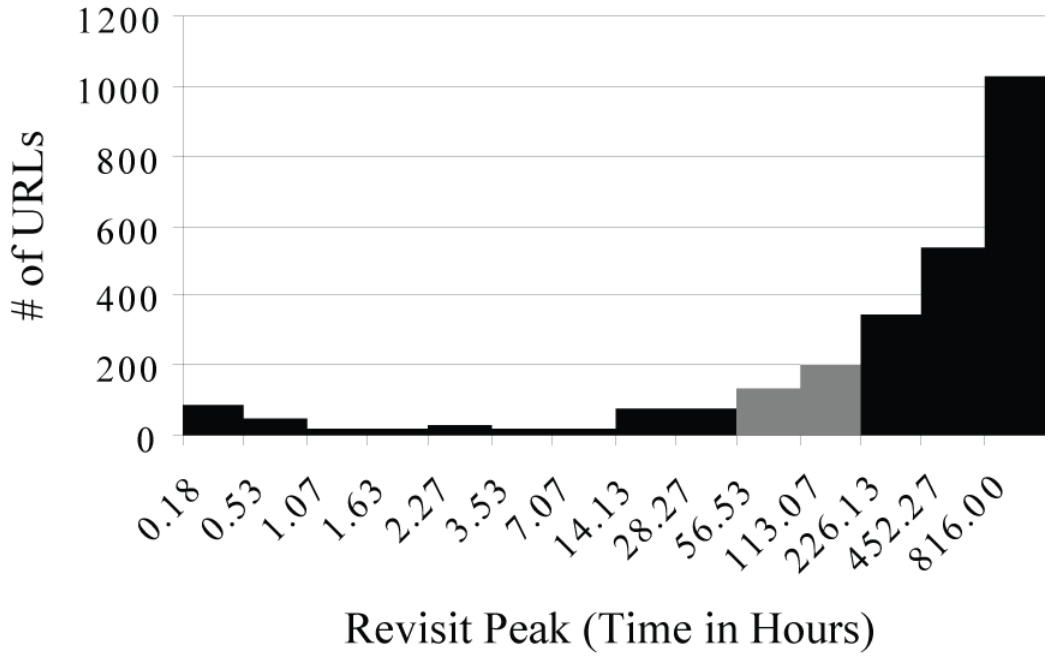


Figure 5.4: Distribution of revisitation peaks (for pages in the medium and slow revisitation categories) for a fixed knot point (56-113 hours, colored in gray, bin labels offset).

in any changing content. Catching those stories before they decay off the page (i.e., before the knot point) leads to higher revisitation in the early periods. In contrast, Woot, a website that offers a new, one-time offer for electronic goods once a day (every 24 hours, corresponding to the knot point), experiences increased revisitation at the same time that the new deal has been posted. The Costco homepage, which provides entry to the mega-warehouse's Internet site has revisitation rates peaking far after the knot point. Although the page presents new deals, it also provides entry to the company's catalog, store information, and other details which are likely not needed on a daily basis. The late peaking of the revisitation curve relative to the early knot point may indicate a user need for accessing the stable, unchanging aspects of the website.

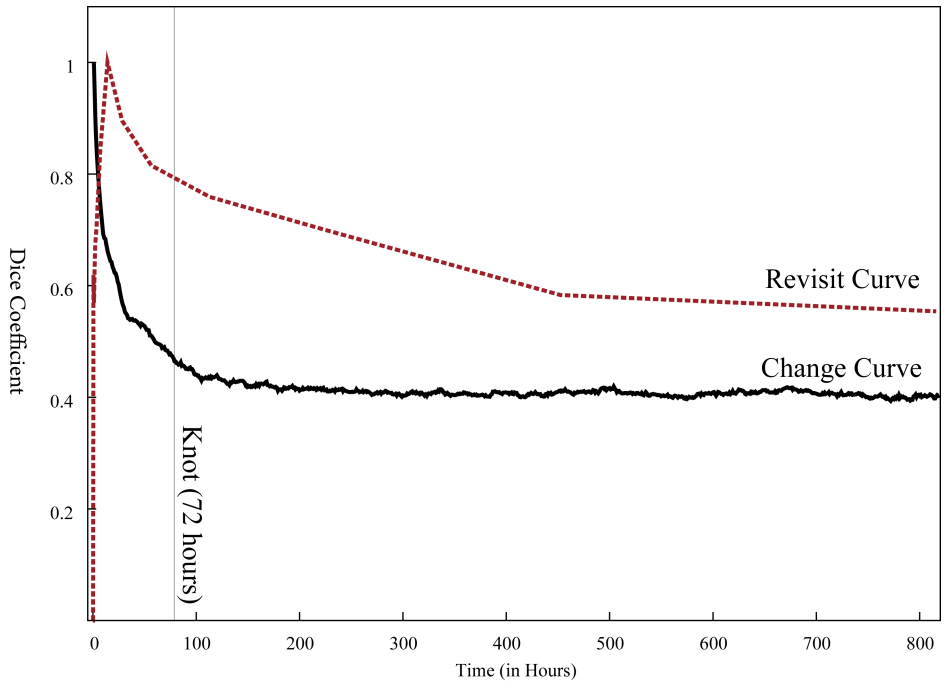


Figure 5.5: Revisitation and (normalized) change curves for the home page of the New York Times (http://www.nytimes.com)

5.3.1 Before, During, and After Knot Points

To more formally verify the hypothesis that depending on the target of revisitation (static or dynamic content) revisitation peaks will come before, at, or after the knot point (i.e., revisitation peak before knot if target is highly dynamic, after knot if target is static) the following experiment was constructed. Subsets of pages were built with each group consisting of pages that have a more clearly defined purpose and relate the change curves to knot points for those pages. Specifically, categories of pages were chosen where there is some expectation about the motivation of revisitation (e.g., news or shopping).

In order to evaluate whether we have significantly more revisits than expected before or after the knot point, a set of bins that are dependent on the position of the knot point was generated. Because the most interesting revisitation peaks are "just before" or "just after" (as in the Woot example), the interval of 2.5% - 5% of the timeline (∼20–40 hours)

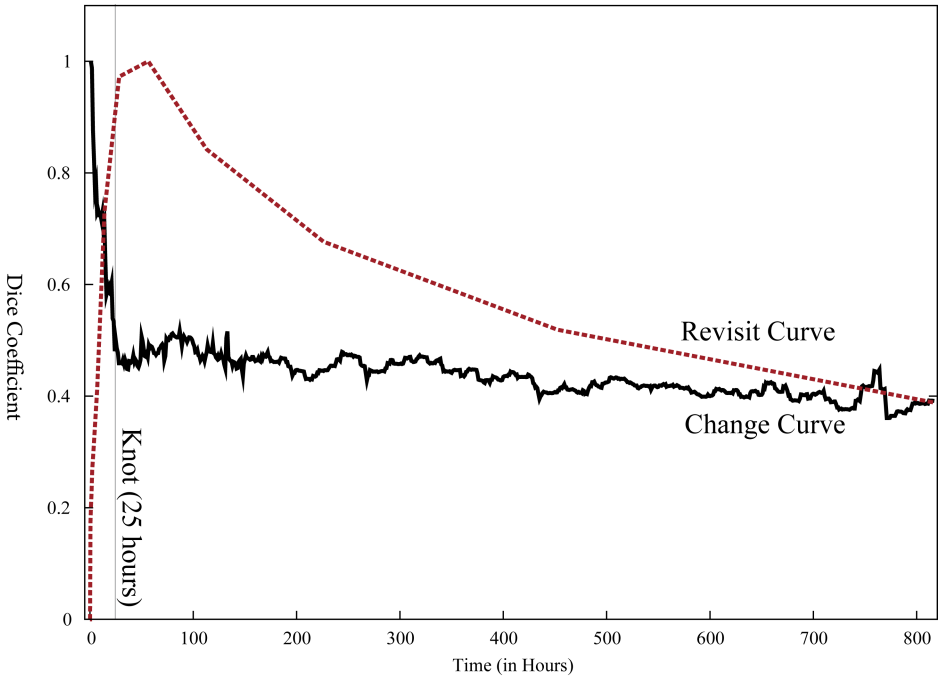


Figure 5.6: Revisitation and (normalized) change curves for the home page for Woot (http://www.woot.com)

was used to define the bins immediately before and after. In total, 4 bins were used for each URL: far before the knot point (more than 20-40 hours before), immediately before, immediately after, and far after (more than 20-40 hours). For each bin the number of revisits was counted, and the counts were normalized by the expected number of revisits for all Web pages. The normalized bins therefore represent the percentage of expected revisits actually observed in each bin (< 1 is less than expected, > 1 is greater). To test the hypothesis, groups of interest were selected (e.g. news pages or forum pages) and a control set of pages not in this group but with a similar distribution of temporal knot locations. These groups were analyzed by comparing the group of interest by the prevalence of revisits in bins before and after the knot point.

Four groups of interest were tested–news pages and forums (which are used to keep up with new information) and pornographic and shopping pages (which often have rapidly changing ads). These four specific types were selected as there is some testable expectation

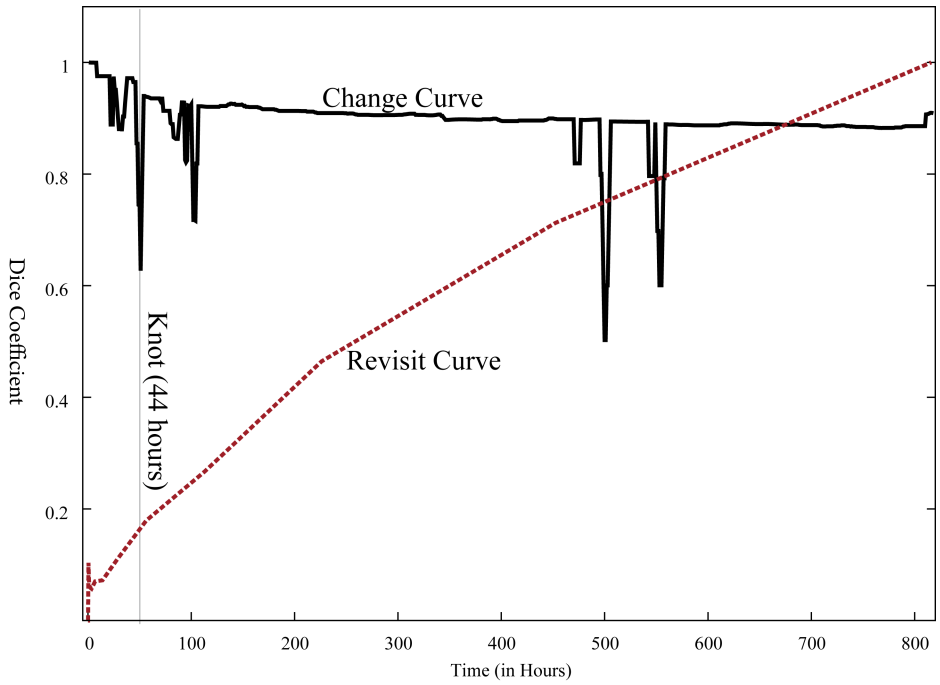


Figure 5.7: Revisitation and (normalized) change curves for the home page of Costco (http://www.costco.com)

about the relationship between change and revisitation. Some key findings include:

- Homepages for news organizations, which are generally used to find new information, tended to have mean revisitation peak *before* the knot point (i.e., before content has vanished, Kruskal-Wallis (KW), $p < 0.00001$).

- Forum pages (those with "forum" in the URL) also display a revisitation tendency to be before the knot (KW, $p < 0.001$).

- One would expect that pages with a large number of rapidly changing ads/spam are less likely to attract revisits that match this frequency of change. This pattern is seen for homepages classified as pornographic, where revisitations are slower than the knot on average (KW, $p < 0.01$).

- Revisits for homepages of retailers ("Shopping" categories), are generally slower than the knot indicating that the rapidly changing information is less critical in driving revisits (KW, $p < 0.01$).

In general, these results confirm a relationship between knot point and "intent" as measured by revisits. However, further evidence can be found by directly asking users in our smaller scale user study.

5.3.2 Revisitation Intent and Change

In the user survey participants were asked about their intent in revisiting Web pages. Intents included finding or monitoring new information, re-finding previously viewed information, form filling, communication, shopping, and homepage (i.e., accessing the browser startup page). Participants were more likely to be interested in finding or monitoring new information in pages that changed rapidly, and more likely to re-find previously viewed information in pages that changed less frequently. For 19 of the URLs, participants explicitly responded they were looking for new information when they visited the page. For 11 of the pages they responded that they were monitoring information, and for nine they were interested in previously viewed information. The mean/median knot point for each was 59.8/46 (new information), 53.3/46 (monitoring), and 88.9/87 hours (previously viewed). This is suggestive evidence that the knot point was sooner (weakly significantly at $p < 0.1$) for pages labeled as "monitoring" and "finding new information" when compared to those labeled as "visiting old information." All four instances where the page was used to communicate with other people (via email or message boards) involved very quick knot points (the longest being 23 hours) and medium revisitation patterns.

Looking more closely at the specific reasons people gave for revisiting certain URLs, the most common reason (given 29 times) was to use a search engine or enter data in a form. Pages marked with this revisitation reason changed less frequently, with a knot point of 85.4 hours instead of 61.9 hours. These are pages where the participants appeared to not be interested in change. As one person stated, "I am pretty sure the page changes regularly, but as I am interested in is the search field, and it doesn't change. I don't notice anything

else."

5.4 Resonance and Structure

As illustrated above, resonance is not necessarily between revisitation and the overall change rate of the page. Some revisitation behavior resonates with the fastest changing content on the page, others with the more stable information. Thus, it should be possible to separate out different portions of the page and identify those most likely to be relevant to the visitor. In the past, identifying such content would require more expensive eye or mouse tracking studies, interviews, or surveys. Instead, this simple technique utilizes revisitation and change information to automatically identify these targets. Though ideally one might combine this algorithm with additional information, this technique represents a novel, extensible mechanism for partitioning pages into more and less important information.

Abstractly, one would like to rank sub-pieces of the page–which are each changing at a different rate–by their similarity to the revisitation rate. To accomplish this, the Structure Analysis Algorithm (see Appendix A) is applied to calculate the change rates of each DOM element.

Figure 5.8a-c shows histograms of the proportion of DOM elements that change at different points in time for three different pages. Note that individual DOM elements change at different times, which is not reflected in the page-level change measures (which change at the fastest rate of any individual element). Above each histogram are the change and revisitation curves for the page as a whole. The grey vertical bars highlight the most prevalent revisitation interval (i.e., the peak of the revisitation curve). Given the change rates associated with each element, we can apply a filter that selects those elements that are changing at approximately the same rate as revisitation. Figure 5.8d-f is an image corresponding to the three Web pages, each illustrating a different revisitation-to-change relationships. The Seattle Post Intelligencer page (d), for example, has very fast revisitation periods. Masking elements with slow rates of change hides navigation elements and slow changing information leaving only "breaking news", current weather, and advertisements). Masking in the Woot page (e) highlights those elements that change approximately every

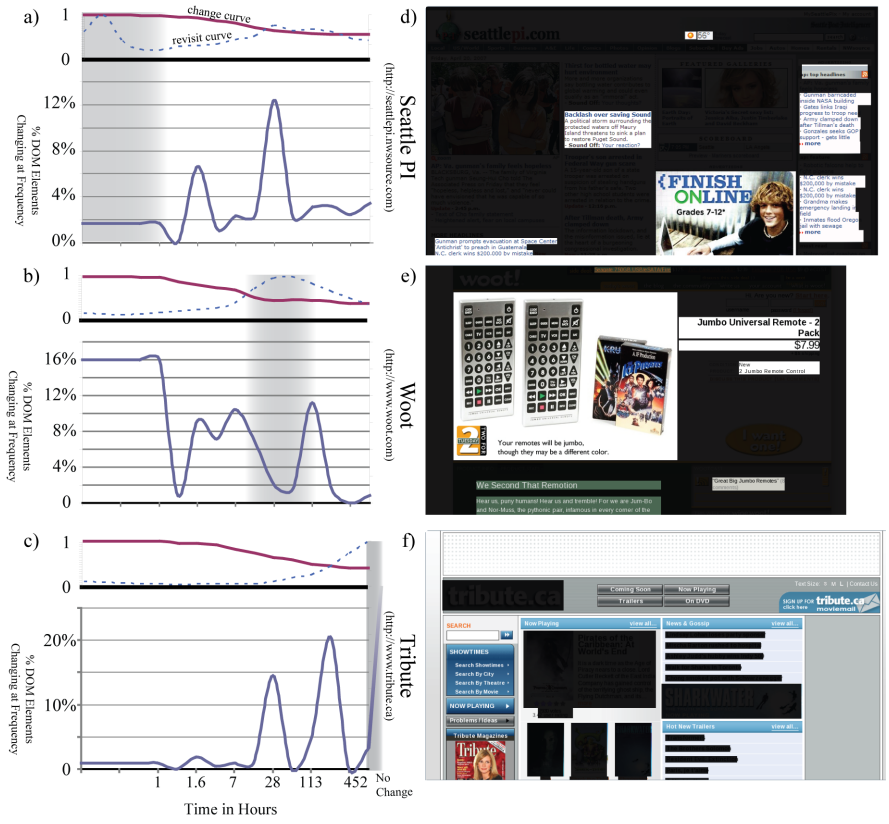


Figure 5.8: Figure 5.8(a-c) A combined chart for each page's revisitation, change (normalized to 1) and DOM curves. The change and revisit curves are normalized to 1. Beneath each change/revisit pair is a plot of the amount of the page (as percent of DOM elements) changing at a given rate. The gray bars represent the approximate peak of revisitation behavior. Figure 5.8(d-e) are the evaluated pages showing DOM elements that are changing at approximately the peak revisitation rate (for the Tribute page this includes unchanging content)

24 hours, which is information about the new product being sold. Finally, the Tribute.ca site (f), the homepage for a movie rating and showtime database for Canada, has very slow revisits. Masking fast changing elements leaves only the navigation and search elements which are the likely targets of revisitations. A demonstration applet of the Structural Analysis Algorithm is available at http://cond.org/resonance.html.

The algorithm works particularly well when there are elements on the page that are clearly differentiated. This may not always be the case as some "undesired" content may change at the same rate as content that is being monitored. For example, the stock market average on the New York Times homepage changes rapidly–likely on every visit of the crawler–as do advertisements. In this situation, both elements are displayed, but it is likely one is of more interest than the other. Additional work on grouping and clustering elements that are near each other visually or have certain shapes consistent with advertising may help. Additionally, use of of click logs and potentially mouse or eye tracking can further improve the results of this algorithm. The benefit of the approach we propose is that it can be automated and scaled to many pages very easily. Historical revisitation information can provide a unique mechanism for inferring which portions of the page are of interest to page visitors. This approach may of course be personalized, with the filter set to different values, as different groups or individuals may have revisitation rates that diverge from the average.

5.5 Applications and Implications

Beyond creating a more refined set of content streams for Dynamic Web analysis, there are many direct ways the results of this analysis can be used to improve the Web experience for website structure, browser, and search engines.

5.5.1 Website Implications

Website designers would like to understand why users are returning to their pages. While some inference can be made from the links that are clicked on, there are many situations when users revisit and do not click on anything. Our research illustrates a mechanism by which a site owner can gain additional insight into the content that is motivating revisitation

behavior.

A more specific application for Website owners is an optimization of the "what's new" pages that visitors utilize to determine new content that is of interest. As argued in Chapter 4, because different pages, even on the same site, are revisited at different rates, "what's new" pages can be designed at different granularities. The work presented here further argues that the resonance between what is changing and the revisitation pattern may point at content that is more of interest. Thus, a website can identify the rate of change of pages or portions of the page, and highlight those that correspond to the peak revisitation rates. This strategy may also be personalized to include an *individual's* last revisit in the decision about which change to highlight.

5.5.2 Web Browser Design Implications

Change monitoring is an area of active interest for browser implementers and researchers alike (e.g., [43, 94, 124, 153]). Just as a Website designer may create optimized "what's new" information for their pages, a client-side implementation may provide additional change analysis features to the user. The ability to expose and interact with meaningful change would be particularly useful within a client-side Web browser context, where a user's history of interaction is known. Pages displayed in the browser could be annotated to provide the user with indications of change (highlighting not just any change, but those changes that are relevant given revisitation resonance). A browser could also act as a personal Web crawler, and pre-fetch pages that are likely to experience meaningful change (as measured by the revisitation/change resonance). This would allow for a faster Web experience and give users the ability to access new content in offline environments. Rather than only storing the most recent version of the page (or all versions), one could develop a caching system that only stores those pages with changed content that is likely to be of interest (or conversely displaying old versions if stability is preferable).

Other applications where resonance may be used include mobile browsers, where content from the original page may be filtered (as illustrated in Figure 5.8) to highlight what is more likely to be of interest to the user. For example, knowing that a news site has fast revisitation

would allow the mobile application to pull out the rapidly changing content and hide or reorder the display to downplay slow-changing or unchanging information. Conversely, a page with slow revisitation patterns might be filtered of fast-changing content to highlight navigation and search structures. Removing information that is less likely of interest might save bandwidth and screen real estate in other applications as well.

5.5.3 Search Engine Implications

The analysis of revisitation and change also has a number of implications for search engine design, in particular for the related issue of re-finding. Prior research has demonstrated re-finding behavior is prevalent [152, 140] in search engine use. Our analysis, which further demonstrates a relationship between the changes to pages and specific kinds of revisitation behavior, suggests several ways search engines can support re-finding in the dynamic environment of the Web.

Just as a browser can provide intelligent re-crawling based on the resonance between revisitation and change, a search engine can achieve the same result on a much larger scale. To our knowledge this is not done currently. Optimized crawling may lead to crawling strategies that understand what information on a page is interesting and should be tracked more or less aggressively for indexing. For example, change in advertising content or change that occurs as a result of touching a page (e.g., changes to visit counters or information about page load time) should not inspire repeat crawls, while change to important content should. Furthermore, a document need not be indexed if it has not changed in a meaningful way, potentially saving server resources.

The types of pages a person is interested in may also suggest how receptive that person is to new information (e.g. suggestions of related content or advertisements). For example, if a query returns results that we suspect are interesting because they contain new content, this could indicate that the user is looking for something new and may be particularly responsive to the suggestion of relevant content. On the other hand, if the results are primarily ones where we suspect the static content is interesting, the user may be more likely to have a specific intent and not respond to suggestions. Content that is somewhat orthogonal to the

user's objective may be most helpful in these cases by appealing to different interests.

A search engine with a rich understanding that the Web is a dynamic information environment could also benefit its end users in more direct, obvious ways by exposing page change in its user interface. If we can make an intelligent guess as to whether searchers who are revisiting previous information sources are interested in re-finding previously viewed content or in viewing newly available content, we can better support both behaviors. For searchers interested in new content (identified either by the query alone or by the user's query history), the search result summaries could highlight the new content

5.6 Summary

Though much research has been generated on both the evolution of the Web and the revisitation behavior of users, little has been done to tie the two together. The research presented here makes a significant step in understanding the association between change and access. The work is unique among studies of Web content change in that the (40k) pages are actually used (by 2.3M users) and unique among studies of revisitation in that we focus on how content change relates to revisitation. The work identified and quantified both the non-linear relationships between behavior and change as well as the importance of changing content in different situations. Because simple assumptions and metrics of change/revisitation interaction limit the ability to understand and leverage this relationship, new metrics and techniques were introduced. While observing that relationships such as increased popularity correlating with rapid changes, the work also identified that some revisits (i.e., quick) are more strongly related to change and that page revisitation is not directly synchronized to the amount of time between changes or when a page "stabilizes." The analysis provides an alternative to other metrics and allows us to infer potential features of interest on the page, be they highly dynamic content, stable search and navigation, or something in between. Specifically, by combining revisitation and change curves it is possible to identify the *type* of information being targetted. Additionally, the approach illustrates how different revisitation patterns resonate with different kinds of changes. Utilizing a novel Structural Analysis Algorithm, this chapter also describes a mechanism for identifying not just the

type of content, but the actual content on the page that might be the target of revisitation.

The implications of the relationships between revisitation behavior and change have applicability to a wide range of services from the individual's browser to the community's search engine. For the individual user interested in monitoring content or re-accessing what was there before, it is valuable to design systems that are cognizant of how information changes. Systems that are aware of potential intent in relation to the changing information should be able to leverage this information in any situation where monitoring, revisiting, and re-finding behaviors exist. This understanding may also enable new applications. For example, by recognizing the revisitation patterns of the user, a mobile browser might filter content to only display stable information or might only render that which is changed.

Chapter 6

Correlated Content Streams on the Web

Though there is a tremendous amount of research on the behavior of users on different web-based systems, there is almost no work on correlating these behaviors. Whether it is by surfing the web, posting on blogs, searching in search engines, or participating in social media systems, users leave traces of their interests all over the web. While interest in some things is constant (e.g., someone, somewhere needs access to their online bank), at other times it is periodic or spontaneously peaks in response to a breaking news event. The effect of these events are ripples of behavioral changes as users rush to search for more information, consume news media, post in blogs, and participate in collaborative systems. However, the size of the response and how quickly users react depends on both the populations and the medium. The reaction can be virtually instantaneous at one extreme (e.g., a search engine query), or require a great deal of time (e.g. the posting of a well-researched news article). One objective of Dynamic Web analysis is to predict and explain behaviors by understanding how, when, and why these variations occur.

The ability to predict and classify behavioral reactions would have wide consequences on everything from the scientific understanding of sociological phenomena to the engineered optimization of search engines. The work described in this study fits into our larger vision of broad, automated prediction by providing the infrastructure and tools necessary to explore

how different Internet systems react in relation to each other. Though not targeted at completely automated predictions, the tools can provide answers to user-directed questions such as: *When do blog posts lead search behavior*? *Do users on different search engines react the same way to different news*? *Can the broadcast of a TV show impact search*?

As described in Chapter 3, one of the main ambiguities in Dynamic Web analysis is precisely when something *has* happened. This chapter focuses on two other problems: when do multiple communities behave similarly in response to some external event? and can one content or behavior stream act as a leading indicator for another? The answer to the first question provides a partial solution to the problem of missing data by identifying situations in which easy to collect data sufficiently models a more difficult to collect stream. By answering the prediction question many services that rely on reacting in advance (e.g., crawl systems, search systems) can do so. Ideally, these techniques could be applied at the granularity of an individual page or user. Unfortunately, there is rarely enough data in these situations to accurately model and predict content and behavior streams. Instead, this chapter focuses on sets of related streams (e.g., content streams from news articles, blogs, TV shows and behavioral streams from searchers). Figure 6.1 represents a potential configuration in which some event elicits reactions in different communities, some potentially "downstream" of others. Note, however, that in this dissertation the focus is on correlating streams rather than trying to infer causality or information flow.

Figure 6.2 reflects the general framework for going from data to predictions. Previous efforts in time series analysis have demonstrated effective ways to apply certain models to data in order to arrive at predictions (e.g., ARMA, ARIMA, Transfer Functions, etc.). However, such tools require that a) data be presented in a compatible format and b) that the research have an appropriate model or reasonable hypothesis. The effort of this work is in solving these first two problems. Specifically, we describe how one can go from structured, unstructured, and semi-structured information to a time series. This work is based on the use of 6 datasets ranging in size from 107M search engine queries, to millions of blog posts, and to 1000's of votes on a specialized website.

To help with model development and data exploration, we develop a system for modeling and measuring responses in different Web systems and a mechanism for comparing those

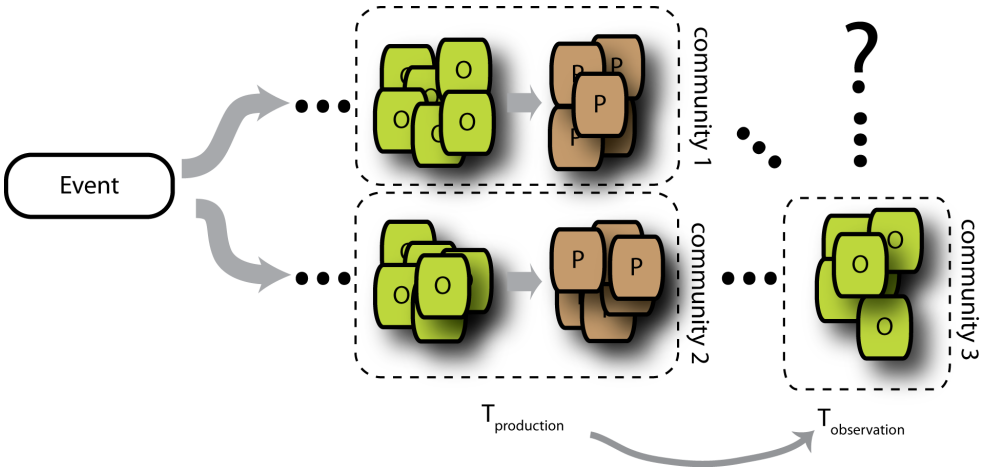


Figure 6.1: Different communities may observe, and react, to the same event at different times.

responses using Dynamic Time Warping (DTW). The system includes a novel visual artifact called a DTWRadar used for summarizing and searching the differences between multiple time-series. The tools and algorithms are motivated by a need to quantify and explore the event space in a human driven fashion as well as automated exploratory analysis.

With these tools a number of models are built to describe how search behavior relates to other behaviors (e.g., blogging or news posting). By understanding specifically how search (or blogging or any other digital behavior) relates to "real world" events, interesting inferences can be drawn. Related efforts in this space have targeted specific prediction and modeling tasks such as flu incidence [69], predicting movie and book sales based on blogging behavior. Predicting incidence of flu in a specific area, for example, might be based on searches for specific medications or blog postings stating "I'm sick" (assuming geo-located blogs). One could also imagine an algorithm for predicting certain kinds of financial instrument behaviors based on postings. Imagine, for example, predicting home buying and selling markets based on queries as people explore mortgage rates and real estate agents.

The work described here focuses on the first steps in the framework described in Figure 6.2. Specifically, contributions over other projects (see Section 8.3) include transfor-

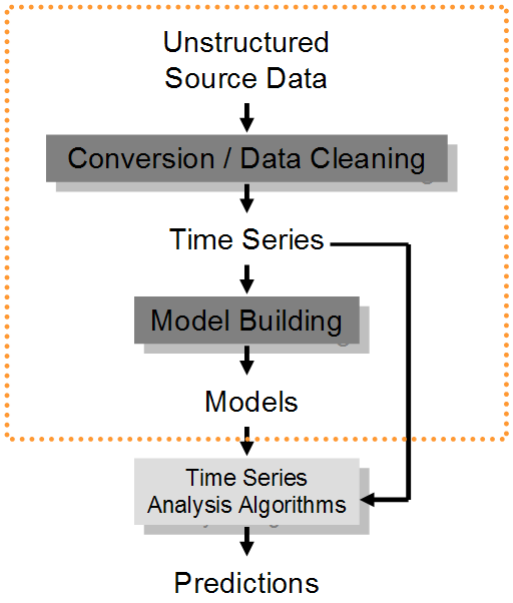


Figure 6.2: High level architecture for trend analysis

mations of different stream data to something that can be quantitatively analyzed, the novel correlation analysis, the visualization infrastructure, and quantitative and qualitative findings.

The study presented is the result of collaboration with Dan Weld, Brian Bershad, and Steve Gribble ([7]).

6.1 Query and Topic Event Streams

Query Event Streams (QES) and *topic event streams* (TES) are specially behavior streams. The former represent all searches (i.e., queries) for a specific phrase. Query event streams may also include all content that *would* match against a given query. Topic event streams aggregate related queries into one stream (e.g., "USPS" and "U.S. Postal Service" and "mailman"). *Query*, here, is used in the usual sense, that of multiple tokens submitted to a search engine (e.g., "american idol"). For simplicity, topic event streams will generally be referred to by the most frequent query for that topic.

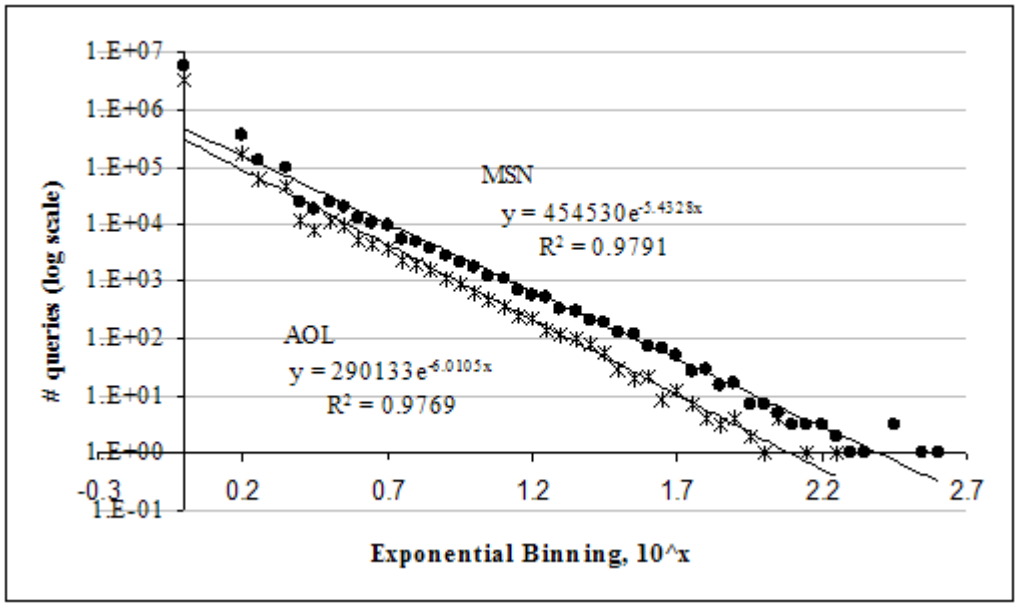


Figure 6.3: Query distribution of the MSN and AOL logs (log-log plot)

Both QES and TES are actually time-series that represent a normalized count of the number of "mentions"–or more precisely, interest–in some specific query or topic in a given period. For example, for search behavior, a QES for "american idol" would count the number of searches for the specific phrase "american idol" in each period (1 hour units at minimum). Thus:

$$QES_{american\_idol} = \{\langle T_0, C_0\rangle, \langle T_1, C_1\rangle, \ldots, \langle T_n, C_n\rangle\}$$

Where $T_0 \ldots T_n$ are the timestamps of the bin (i.e., when binning by hours the bin range, n, for a month would be from 0 to 755, or 0 to 31 when binning by 24 hour periods) and $C_0 \ldots C_n$ are the corresponding weights. In this case, the weights are the number of queries made for "american idol" in that period. In the particular case of query logs, because there is variability in the *total* number of queries or stories published over the course of the day (less queries during the North American nighttime, for example), counts are normalized. Thus, if there were 10 queries for "american idol" in the first hour, and 1432 total queries,

C_0 would be $10/1432 \sim .007$. To maintain the time-series notation, null-tuples (tuples with a weight of 0) are generated for periods in which a given query/event is not witnessed.

Depending on the situation, alternative weighting strategies are used instead of counts. This is frequently done to transform content streams to behavior streams. For example, because readership of particular blogs or news articles is not directly accessible, this might be estimated by the number of incoming links. Thus, the TES or QES that was originally for content can be transformed to something else (i.e., a behavior stream) by changing the weights.

Finally, because this study works with a number of different data sets, specific streams are indicated using notation of the form $QES^{dataset}_{query}$ and $TES^{dataset}_{topic}$. For example, the "american idol" QES from the blog dataset would be $QES^{\text{BLOG}}_{american\_idol}$.

6.2 The Datasets

To support the research goal of understanding how different time-varying behaviors correlate, six different datasets are used. Each dataset represents a historical record of the interest users have in different topics in different systems. the datasets are composed of a log of queries and clickthroughs for MSN and AOL users, blog posts, news posts from CNN and the BBC, and posts about TV shows from TV.com.

6.2.1 MSN Query Logs (MSN)

The primary dataset used in this study is a query log sample from May 1–31, 2006 from the MSN search engine [106]. The log contains 7.47M sessions representing 15 million queries for this period as well as the respective clickthroughs (complete URL). Although the original data does not include complete information about the sampling methodology, the logs represent a random sample of sessions (i.e., a continuous, cookie-tracked browser session). For consistency with the AOL dataset used below, traces are normalized by "breaking" a session when a user has not been active for more than an hour. The result is 7.51M redefined sessions for the period. This is only slightly more than the original count and indicates that most continuous browser sessions last an hour or less. To reduce the memory requirements,

tuples were generated by binning at the hour level with the weight normalized by the total number of queries issued during that period.

6.2.2 AOL Query Logs (AOL)

The second dataset was extracted from the AOL query logs [131]. The trace represents the browsing behavior of 484k users generating 12.2M queries for the period of interest (May 1–31). As above, the AOL dataset was normalized into sessions by breaking long traces into a distinct session with an hour of inactivity (yielding 2.6M sessions for the period).

While breaking sessions in this way was an attempt at normalizing the two query logs, it is notable that the particular sampling methodology creates a different distribution of query popularity. Because users have a known preference towards re-querying information they have previously sought [152], there is a potential bias in the AOL dataset towards fewer unique queries. For example, despite the fact that the AOL session population is a third the size of the MSN population, it nonetheless generates a similar number of queries. Although the distributions (see Figure 6.3) for query recurrence are similar, there are nearly twice the number of unique session-query pairs in the MSN set (6.6M vs 3.7M). While this difference may be caused by some intrinsic feature of the MSN and AOL user populations, it may simply indicate that repeated sampling from the same set of users does not yield the full variety of queries found by random sampling. The effect of this is that the topics found in the AOL set may not represent a sampling of global interest in a topic.

Despite this, both the AOL and MSN datasets share a common rank 1 query ("google") and have a 80% overlap for the first 10 queries. Beyond the 10th query, the overlap drops to 66% and stabilizes for as high as the first 50k ranked queries. This high overlap provides some support for comparing the two datasets, at least for some queries. The consequences of this are discussed below.

One further issue with the AOL dataset was a small period of missing data of approximately 22 hours around May 16th. Clearly, it is unreasonable to simply splice out the data from this period as it would represent a temporal shift that would be unmatched in the other datasets. Other solutions include leaving the data at this range as 0, "patching" the

data by replacing this period with the average values between the two end points, or the linear extrapolation of the two ends. All three options appear to generate roughly the same cross-correlation levels. The only situations for which these techniques would fail to capture a correlation accurately are those when there is a single spike in query behavior that greatly overlaps with the missing time period.

6.2.3 The Blog Dataset (BLOG)

In addition to the two query logs the study also made use of a database of 14 million posts from 3 million blogs from May 1–24, 2006 [36]. Though the posts were annotated with a date/time tag, these tags were not normalized to any time zone. By manually testing the feeds from a number of blogging sites (e.g., Blogspot, LiveJournal, Xanga, etc.) it was possible to generate the correct time zones for a number of the posts. However, because many of the remaining posts came from popular blogs that were not readily validated, blogs were binned by data (24 hours).

6.2.4 The News Datasets (NEWS & BLOG-NEWS)

One hypothesis for this data is that the behavior of bloggers and search engine users was frequently influenced by items in the news. In order to validate this we generated a news dataset by crawling the CNN.com and BBC.co.uk websites in search of documents from the May 1 31 time frame. These two sources were selected as they could be crawled easily and contained normalized (GMT-based) timestamps. The result was that over 12k articles were downloaded from the BBC from the period of interest and roughly 950 articles from CNN (many CNN articles are derived from Reuters and are no longer available online).

Ideally, one would like as many news sources as possible in order to determine the popularity of a certain topic over time. However, because most sources do not provide access to their datasets after 7 days, it was difficult for to find more articles from the period of interest. Since the CNN and BBC sources will likely publish only one story about a given topic, for most topics this does not give us much signal to work with (i.e., each QES contains only one tuple with a weight of 1). One modification to generate more realistic

weights is to find the interest level in specific articles from external sources. For example, one non-binary approximation of popularity is the number of inlinks to a specific article. To find this value, the MSN search engine was used to find all *links* to the articles from external (i.e., non CNN or BBC) sources. Modifying the previous notation slightly, each tuples weight was set to the number of incoming links. We refer to this event stream as NEWS. Because all articles are time-stamped, the news dataset can be grouped at the hour level and normalized by the number of inlinks to all stories in that hour.

While re-weighting as above gives us a certain amount of information about eventual interest (useful in itself) it does not give us a sense of the changing interest level in a topic. By simply adding weight at the time of the article's publication based on interest, we force the mass of weight towards one point. In reality, interest in a topic, and hence an article's popularity, changes over time as links are added. To simulate this we make further use of the BLOG dataset and count the number of daily inlinks to an article in the posts. Think of this as distributing the total number of (blog) inlinks so that the tuple from each day has a weight equal to the number of inlinks from that day. We call this dataset the BLOG-NEWS set. Because of the binning (24 hour binning) and time-span limits (only 24 days of data) of the BLOG dataset, the binning for BLOG-NEWS is at a daily level with data only up until the 24th of May.

6.2.5 The TV Dataset (TV)

A number of the most popular queries in the MSN dataset appear to be targeted at finding information about TV shows, actors, and actresses. It is likely then that the broadcast of a specific TV shows precedes or succeeds a burst of querying activity as users anticipate or react to the show. Similarly, interest in actors and actresses may vary based on the popularity of a show. To test this, all episodes of shows broadcast during May of 2006 were crawled on the TV.com website. The site was chosen as it is a popular site for the discussion of TV shows and contains detailed information about the cast, summaries of the episodes, and user ratings. The crawl produced 2457 different shows for the period with 2 pages per show (a short summary and an extended recap). Because the event "weight" should

represent viewership, this value is estimated by the number of votes an episode received on the website (all sites were given a minimum of 1 vote for normalization purposes). Because a particular TV show may be broadcast at different times on multiple occasions, pages from this dataset were tagged by day of the first broadcast and binned in 24 hour intervals.

The BLOG, NEWS, and TV datasets were indexed in a Lucene (http://lucene.apache.org) index for easy retrieval. Table 6.1 summarizes the datasets.

Table 6.1: A summary of the datasets used.

Dataset Name	**Time range**	**Bins (hours)**	**Source records**	**Weight scheme**
MSN	May 1–31	1	7.51M search sessions	Searches per unit time
AOL	May 1–31	1	2.6M search sessions	Searches per unit time
BLOG	May 1–24	24	14 M posts, 3 M blogs	Posts per unit time
NEWS	May 1–31	1	13K news articles	Total in-links
BLOG-NEWS	May 1–24	24	13K news articles	Blog-source in-links per unit time
TV	May 1–31	24	2547 TV show records	Votes

6.3 From Queries to Topics

One issue ignored thus far is how topics are actually generated. Because the study was primarily focused on studying how query behavior relates to other behaviors, it is important to group sets of queries into related buckets. There is a great deal of literature on topic detection both in data streams and static text collections [10]. While these techniques generate good results, a very simple scheme related to [167] yielded a useful set of queries grouped as "topics." Abstractly, the algorithm works by hierarchically clustering queries based on overlapping clickthrough and search engine result sets.

Our starting dataset includes the 963,320 unique queries that appear two or more times in the MSN and AOL query logs (lowercase normalized). These represent a pool of potential

queries that we can lump together into topics. Each query was submitted to the MSN search engine and up to 50 results were returned. For each of the queries the most popular clickthroughs were noted. In the case of the AOL logs, where only the domain of the clickthrough was logged, the most likely full URL was found by comparing the domain of the click to the returned MSN results. Without much effort, this discovered the full URL for 40% of the redacted clickthroughs. Each query was then mapped to a set of URLs composed of the clickthrough URLs and up to 10 of the top hits from search engine. This set was transformed into a weighted vector using the standard TF-IDF scheme [18] where each of the 6,908,995 URLs (the "terms") was weighted by the number of times they are returned for the queries (the "documents").

A pairwise comparison of the 963,320 queries–using a standard cosine distance metric [18]–resulted in 1,332,470 non-zero edges. An edge represents the similarity of one query to another and is used to cluster our queries for further analysis.

To construct an initial sample of potential topics to study, an initial set of all queries that appear 100 or more times in the MSN logs was collected (these were believed to be slightly more representative of broader search behavior than the AOL logs due to the sampling methodology as explained in Section 4.2). The resulting 5,733 queries are sorted by their frequency. Starting from the most popular query we generate a list of all related queries by traveling the edges described above (1 step). These neighboring queries are "assigned" to the initial query as alternative variants of that query. As we travel down the list of 5,733, those less-frequent queries that have already been assigned are ignored. The end result is a list of 3,771 main queries (i.e., topics) with an average of 16.2 variant queries. Note that the queries we use appear in multiple datasets 96% of the time and do not uniquely identify any user.

In going through the dataset we found that the bulk of variants were misspellings, changes in word ordering, and spacing. Although we have not done so in our experiments, our approach also lends itself to weighting Query Event Streams differently when generating a combined Topic Event Stream for some original query. That is, the weight contributed by a QES of a query that is very similar to some "seed" query can be more than the weight contributed by a less similar query.

Of the generated topics, some appear to be navigational queries for corporate websites (e.g., "amazon," $QES^{\text{MSN|AOL}}_{amazon}$ =  or "bank of america" ) while other are for search engines and information sources (e.g., "white pages"  or "weather" ). However, this set also contains various queries that are connected to external events. These include searches for the ever popular "american idol"  , queries for the holiday "cinco de mayo"  (5th of May), and references to people and events in the news such as "uss oriskany,"  (a US warship sunk in late May of 2006).

The broad distribution of topics was encouraging as it represented both queries from which we would expect to see no interesting correlations, as well as highly correlated events.

6.3.1 Comparing Datasets

There are a number of different parameters to consider in comparing the datasets. These parameters impact the performance of the correlations and require tuning and testing with different variations. There are 756 hourly bins in the period of interest and can conceivably be compared at different levels of granularity. This is supported by the binning parameter, b (recall that a number of the datasets are pre-binned at 24 hours, so for those b must be ≥ 24). As described earlier, binning simply reduces the number of event tuples in each QES by creating a new tuple with combined weights of the tuples in the period. Although binning smooths the data somewhat there is still some noise in variations between bins. In order to diminish this noise in the time series we apply Gaussian smoothing as characterized by the function:

$$G(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-x^2/2\sigma^2}$$

By convolving the Gaussian kernel with each QES or TES we are able to eliminate most noise. Our experiments and tools support arbitrary levels of σ. In the future, it may be worth considering smoothing techniques that are known to more accurately represent the visual characteristics of the original curve [170].

Using the experimental topics defined above, a set of TES's was created for the different topic/dataset pairs. For the log datasets this simply meant finding all queries that matched

one of the queries defined by the topic. To generate the TES's for the remaining datasets we ran each query against the Lucene index. Although initially queries were considered to be Boolean (simulating the most Web search engine behavior), a number of queries generated excessive low-relevance results. Rather than using an arbitrary heuristic threshold for discarding results, Boolean queries were transformed into phrase queries. Thus, the query "britney spears" meant that both words had to appear together. Queries that behaved like stopwords (e.g., "blog" or "www") and matched a large number of documents were discarded (200 or more for the NEWS, BLOG-NEWS and TV datasets, or 25,000 in the case of the BLOG set).

Because not all topics exist in all data sources, of the original 3781 topics we were able to generate 3638 (96% overlap), 3627 (96%), 1975 (52%), 1704 (45%), and 1602 (42%) TES's for the AOL, BLOG, NEWS, BLOG-NEWS, and TV datasets respectively.

To see how many of the TES's displayed random internal behavior, all TES's generated above with a total weight greater than 10 were used (i.e., more than 10 searches for a topic, 10 blog posts, etc.). A test for partial (circular) autocorrelation at .05 significance level finds that 983 ($\sim$ 8 %) of the final 12,324 are considered random ($b = 24$, $\sigma = 0.5$). Though the content of random TES's was not extensively studied, a brief scan of the topics indicates they are composed of company names, websites, and adult queries

6.3.2 Correlating the Datasets

To find the correlation between the two time series (x and y) we use the cross-correlation function:

$$r(d) = \frac{\sum(x(i)-\bar{x})(y(i-d)-\bar{y})}{\sqrt{\sum(x(i)-\bar{x})^2}\sqrt{\sum(y(i-d)-\bar{y})^2}}$$

where the variable d is the delay and is varied from the $length(x)$ to $+length(x)$. This represents the possible shift of the one curve away from the other to each extreme (all the way before, with no overlap, to all the way after). This was done to avoid setting an arbitrary threshold for a match, though with an appropriate model, one could limit d to a smaller range. The maximum value of r is the "best" correlation of the two functions, the value of d at this point is the "best" fitting delay between the two function. We reject

the null of no-correlation by applying a Monte Carlo simulation that reorders one of the time series randomly and repeatedly finds the cross-correlation. By repeatedly generating max cross-correlations values less than the non-randomized time series we can reject the non-correlated hypothesis.

Though this study concentrates on cross-source correlations (i.e., $TES_{topic\_1}^{source\_1}$ versus $TES_{topic\_1}^{source\_2}$) there is no fundamental reason why this analysis could not be applied to different topics in the same source. For example, by shifting the two TES's (e.g., aligning two movies' release dates), one could compare the reaction to two different movies in the same source (e.g. $TES_{x-men}^{\text{BLOG}}$ versus $TES_{superman}^{\text{BLOG}}$).

As a further refinement, only positive correlations are considered. The working hypothesis is that if an event causes an effect in a dataset, all affected topics will see a *positive* change in weight. Although responses may be offset, it is unlikely that a news event will cause decreased search with increased blogging (or vice versa), and so we do not expect that negatively correlated behaviors correspond to valid mappings. In fact, when manually evaluating the results, and taking the positive magnitude of each correlations, the maximum corresponds to a negative correlation the topic appears to not correspond to any real event or trend. Topics of this type appear to be generic searches for company names (e.g., "home depot" or "expedia") or websites (e.g., "nick.com" or "www.myspace.com") and not responses to events. Because of this, the results are restricted to only maximum positive correlations.

Figure 6.4 represents the distribution of all significant correlations found when comparing all TES's in the MSN test dataset against each of its counterparts in the other datasets (bins = 24 hours, $\sigma = 2$). The cutoff point for significance was determined by the formula: $\pm\, z_{1-\alpha/2}/N^{1/2}$, where z is the percent point function of the standard normal distribution, α is the significance level and N is the number of "samples" (in most cases twice the length of the time series). The delay at which those correlations are found is depicted in Figure 6.5. Figure 6.6 is a depiction of the distribution for only those TES's for which the cross-correlation was $\geq .7$ (well exceeding the $p < .05$ significance levels). On average, 38% of delays are centered at 0 days. What is interesting are the remaining, highly-correlated TES's that are shifted from 0. If these shifts are consistent, topics of this class are potentially

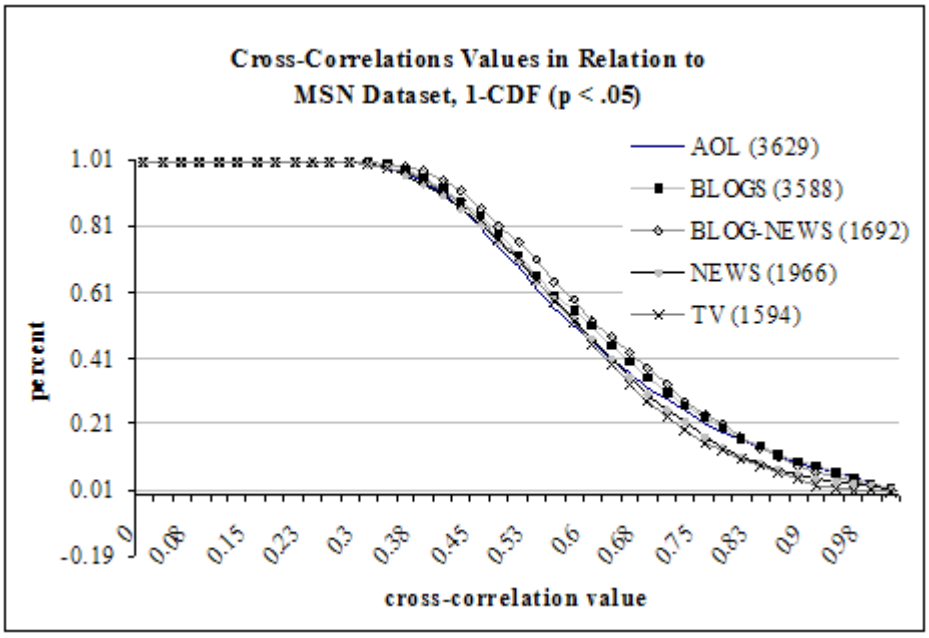


Figure 6.4: The cross-correlation values of all TES's against the equivalent TES on MSN, the y value at each value of x represents the percent of TES's for which the correlation is $\geq x$.

useful for prediction purposes. However, there are many correlated TES's that are not simple spikes but instead repeating patterns, potentially representing a response to multiple events (e.g., a weekly TV show). While there might be an optimal delay as determined by the maximum correlation to align two TES's, each peak in periodic behaviors may not always lead or lag in the same way.

6.4 Dynamic Time Warping (DTW)

One of the limitations of simply shifting two correlated QES's is that it is still difficult to get a sense of how one stream relates to the other in term of magnitude. Imagine, for example, two correlated curves as in Figure 6.7 where one "encapsulates" the other. The difference in magnitude is drawn as a number of vertical lines in the top figure. While this represents a plausible mapping between the two time series, it fails to capture a different

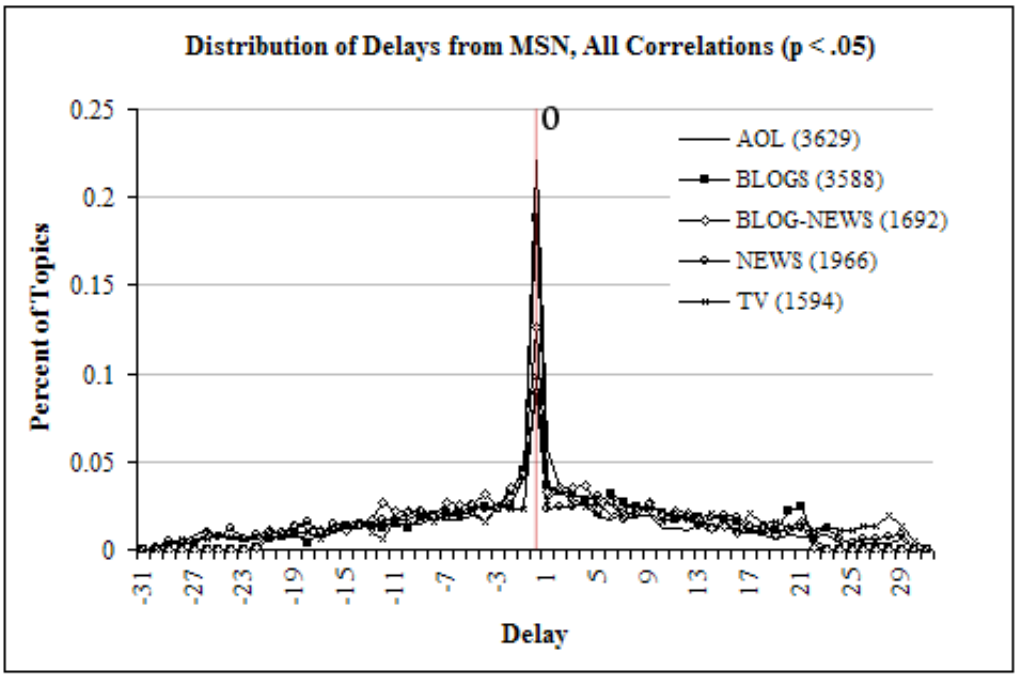


Figure 6.5: Distribution of delays (in days) of all TES's against the equivalent TES on MSN.

interpretation. The bottom series is essentially a smaller version of the top one, condensed both in time and magnitude. Like the top curve, it has a rise, a stable peak, and a fall. Thus a different mapping is one that tries to capture these behavioral properties and map the start of the rise in one curve to the start in the second, and so on, as illustrated in the bottom of Figure 6.7.

A way to achieve this mapping is by making use of Dynamic Time Warping (DTW) [123]. Though there are several versions of this algorithm, a simple scheme using a dynamic programming approach is as follows (assuming x and y are two sequences of equal lengths):

At the end of this run the two dimensional DTW array contains a mapping between the two time series. By starting at the extreme corner of the array and crawling backwards along the minimum gradient the best warp path is determined. In experimenting with the algorithm, it was determined that alternate versions of the cost function that include time as well as magnitude, do not appear to work as consistently due to the sensitivity of the

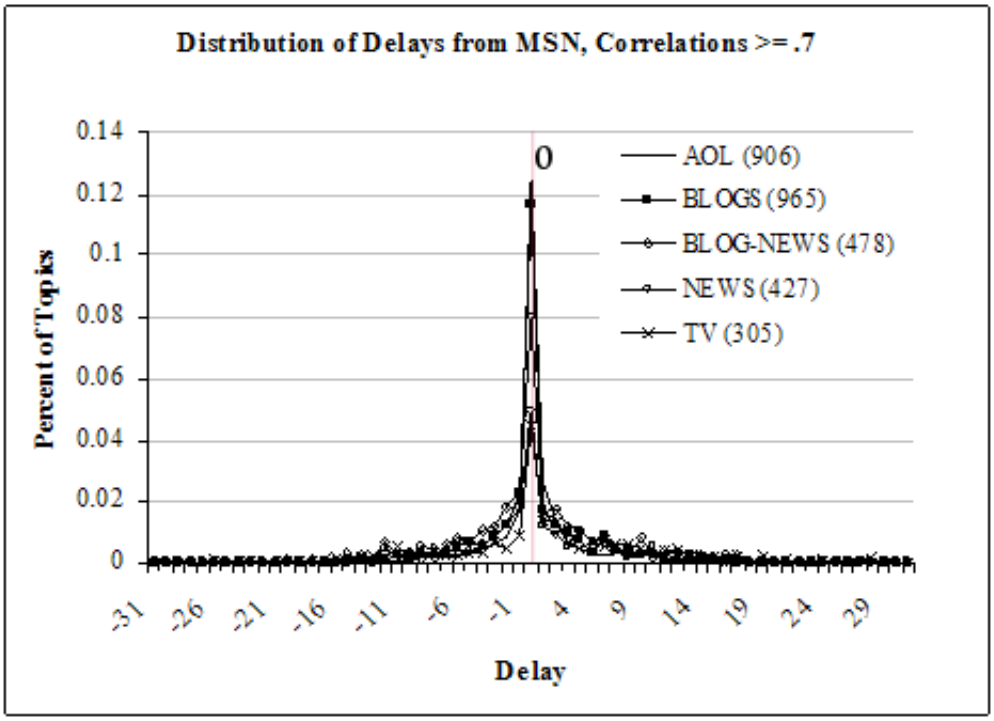


Figure 6.6: Distribution of delays (in days) of TES's with correlation $\geq .7$ against the equivalent TES on MSN.

cost function to the scaling. We also found that in general the results appear more accurate when we used a Sakoe-Chuba band [139], which restricts the warp path to a fixed distance from the diagonal of the DTW array. This technique reduces unrealistic warp paths that map distant events by limiting the max delay.

Another modification that was useful was the use of the delay from the cross-correlation as a shift before calculating the DTW. That is, if the cross-correlation was at a maximum at d_{max}, one curve was shifted by that amount prior to calculating the warp path and subsequently shifted the warp back by that amount.

Finally, we note that the DTW algorithm, as presented, generates an "optimal" mapping from sequence x to sequence y because of the order in which it generates the array. However, this mapping may not be optimal from y to x. To address this issue, the DTW is calculated twice, switching the order of the input, thereby finding those warp mappings that are

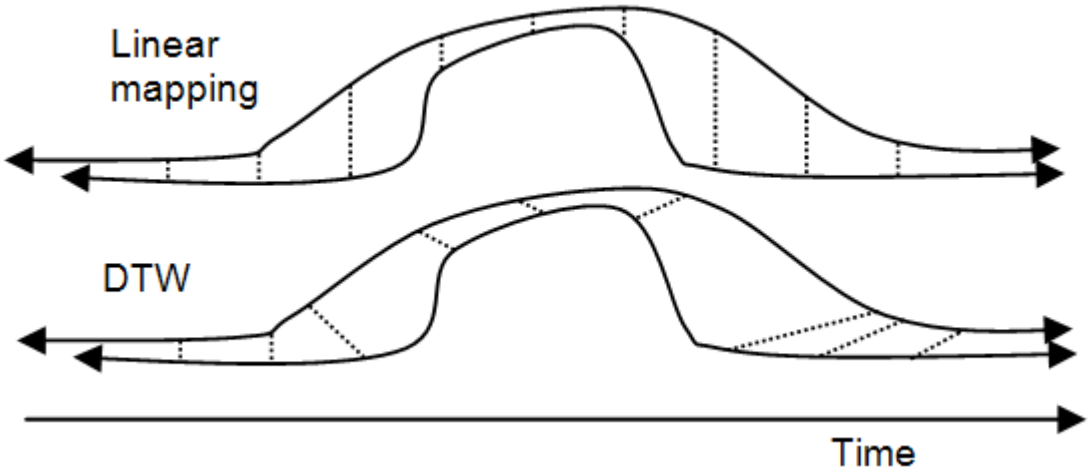


Figure 6.7: Illustration of Dynamic Time Warping (DTW).

Algorithm 1 $warp(x, y)$

Require: x and y, two equal length arrays representing two time series

$\mathrm{DTW}[0,0] \leftarrow 0$
for $i = 1 \ldots length(x)$ **do**
    $\mathrm{DTW}[0,i], \mathrm{DTW}[i,0] \leftarrow \infty$
for $i = 1 \ldots length(x)$ **do**
    for $j = 1 \ldots length(y)$ **do**
        $cost \leftarrow \sqrt{(x[i] - y[j])^2}$
        $\mathrm{DTW}[i,j] \leftarrow min(\mathrm{DTW}[i-1,j] + cost,$
                $\mathrm{DTW}[i,j-1] + cost,$
                $\mathrm{DTW}[i-1,j-1] + cost)$

common to both directions.

To allow users to explore the relationships between QES's and TES's from different sources we constructed an application called DTWExplorer. A sample screenshot is captured in Figures 6.8 and 6.9 for the seed query "lost" (a popular TV show, binned at 24

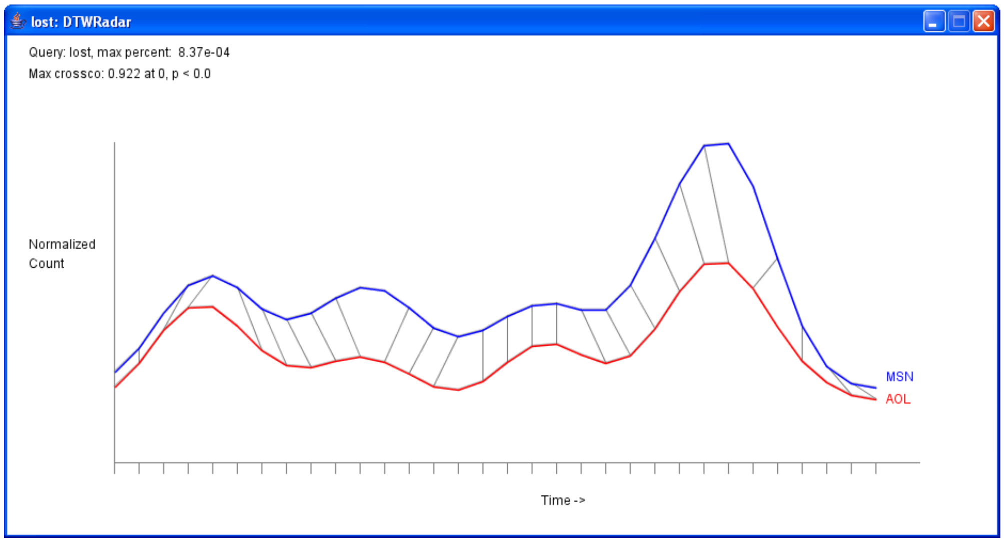


Figure 6.8: The DTWExplorer (time scale in day units). The view is for the initial query "lost," and the two time-series correspond to the MSN (on top in blue) and AOL (below in red). Weekly spikes appear around the time when the TV show Lost was broadcast. The lines connecting the time-series are the warp paths.

hour periods with $\sigma = 2$). Note the weekly oscillations around the time of the broadcast of the show with a significant peak around the season finale.

To use the DTWExplorer, a user submits a seed query to the system which finds a list of similar queries with which to construct a topic. These are listed in the table view similar to the one on the right. The first column indicates whether the events matching the particular query should be included in the analysis. The second column is the query followed by the similarity based on the cosine metric described previously. The fourth column is the number of events of a certain type over the period of interest and the final column contains sparklines [157] that show the cumulative behavior of the QES for that query in both the MSN and AOL logs. Sorting is possible on any column. In the example we have sorted by the time behavior (the "distribution" column) which sorts by finding the cross-correlation of each series to the seed query. When sorted in this way, the top matching queries have a very similar distribution in terms of time. This is inspired by the work of Chien and Immorlica [46] where semantic similarity was determined by time series

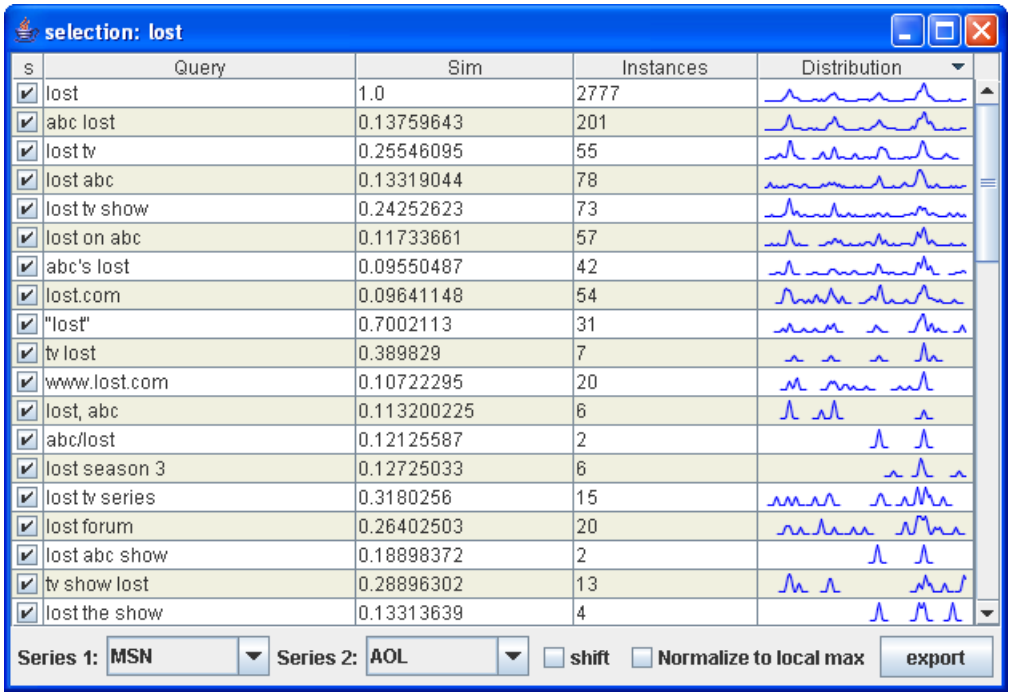


Figure 6.9: The DTWExplorer (associated terms to "lost" corresponding to the visualization in Figure 6.8). When sorting, each line in ordered by the similarity to the original seed query on different dimensions, allowing the user to include or exclude phrases that do not meet similarity criteria (e.g., don't behave similarly over time, don't lead to the same Web pages in search, are uncommon).

correlations. The left hand display shows the output of the DTW on the selected query variants. The thin lines connecting the two curves represent the best DTW mapping. Note that in this implementation the applications supports searches on all indexed queries, not just the smaller high-frequency sample.

The DTWExplorer application also supports the comparison of specific streams to generated time-series corresponding to different periodicities (e.g., once only, weekly, bi-weekly, etc.)

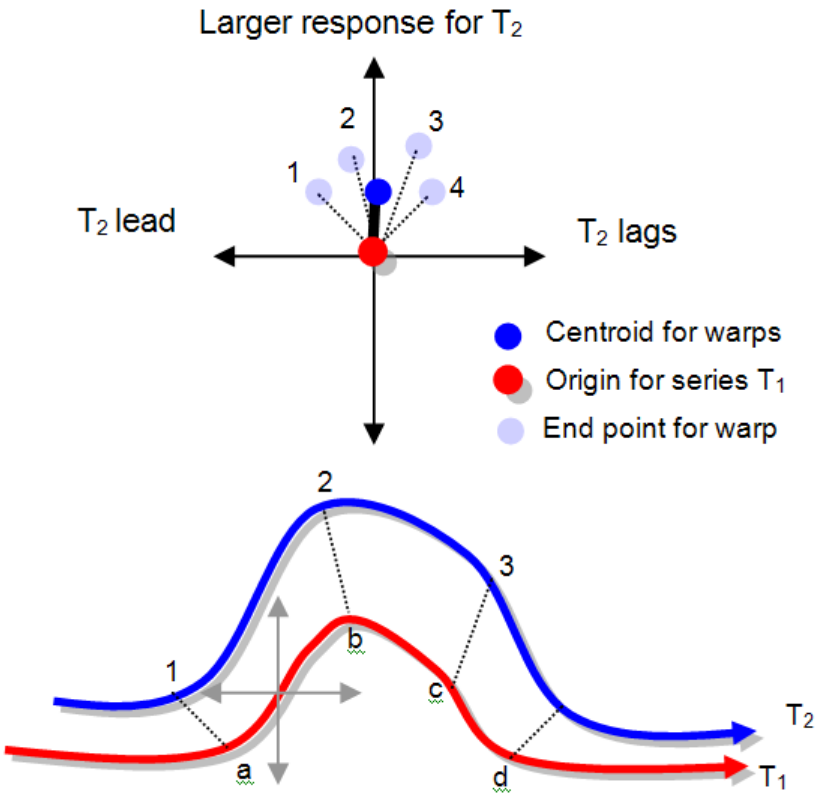


Figure 6.10: A graphical representation of the construction of a DTWRadar. The axis is "moved" over one of the time-series (in this example the bottom, red time-series). As the origin hits each of the warp end points (i.e., points a, b, c, d) the other end point (i.e., points 1, 2, 3, and 4) are added to the plot. Stated another way, each warp path (1–a, 2–b, etc.) is placed on a single two-dimensional axis with points a, b, c, and d placed at the origin.

6.4.1 Warp Summaries: DTWRadar

While the warp paths generated by the DTW algorithm are informative for understanding the changing behavior over time, they do not necessarily give a sense of the overall differences between the two series. Ideally, one would like a summary statistic or visualization that is descriptive of the total behavior of two series in relation to each other.

To achieve this, the DTWRadar was developed. The DTWRadar is a visual construct that takes each warp vector determined by the DTW algorithm and places it on a normalized

axis (see Figure 6.10). Assume two time series T_1 and T_2 and a warp mapping between them (point a to point 1, b to 2, and so on). The radar view is constructed by sliding a two dimensional axis along one of the time series (say T_2) from left to right. As the gray axes hits a mapped point T_1 a new point is drawn on the radar corresponding to the location of the end point of the warp. The result of this motion is illustrated at the top of Figure 6.10. Note the 4, slightly transparent, points on the radar, with a 5th, darker, point showing the centroid of the original 4 points.

In the DTWRadar dimensions the x dimension corresponds to time (specifically the delay between the two time series), and the y dimension corresponding to magnitude (specifically, the difference in magnitudes between the two time series). Understanding this radar simply means looking at the location of the centroid relative to the origin. From this specific example, T_2 has a larger response (high on the y-axis). There is also a very small lag as the centroid is slightly to the right on the x-axis. If the two time series represented the number of queries for some topic X in the two search engines (QES_X^1 vs. QES_X^2) this might indicate that more users query for X on search engine T_2 but with a slight lag. Note that the coordinate system of the DTWRadar maps directly to that of the time series. Thus the view allows the user to know the average lead or lag time by looking at the position of the centroid along the x axis, and the magnitude difference by the position on the y. Keeping the transparent points in the radar allows a user to understand the distribution of the warp (i.e., the magnitude of T_2 is consistently higher than T_1, but varies in terms of lead/lag). Note also that the output of the algorithm is a pair of points, one representing each time series. As shown below, these points are useful for clustering behaviorally-related queries and topics.

Generally, once this representation is explained to a user they are able to easily interpret this view. This one view combines a great deal of data in a compact space and lends itself to the generation of many small multiples and facilitating comparison of multiple relationships simultaneously. The DTWExplorer was extended so that it automatically generates a scaled version of the DTWRadar for each pair of series that are being evaluated. Figure 6.11 illustrates this for the $QES^{\text{BLOG}}_{prison\_break}$ versus $QES^{\text{MSN}}_{prison\_break}$ ($b = 24$ hours, $\sigma = 2$).

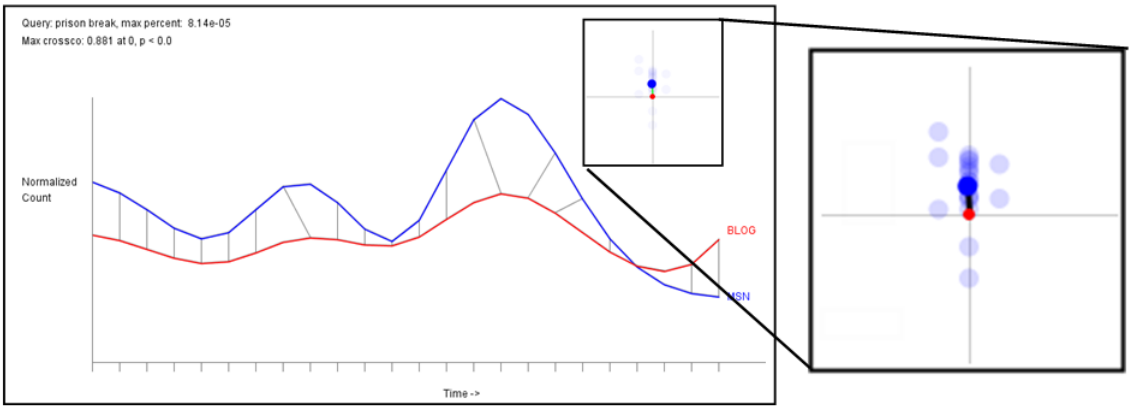


Figure 6.11: DTWExplorer output and overlaid DTWRadar for the query "prison break" (daily scale, radar view magnified on right).

Radar Variants

Clearly, there are a great number of possible variants to the DTWRadar that can enhance understanding of different properties. For example, though time marks have been omitted from the view as they generally present a lot of noise in the scaled versions, these can be added in an expanded representation.

Other variants utilized different centroid/mediod configurations for the "average" point in the radar. From these experiments, it appears that the most useful selection of this point depends on the distribution of the individual warp points. One extreme warp point, correct or otherwise, can greatly influence the location of a centroid. At the moment, the explorer interface supports both forms.

Another possible radar view is one that weights later mappings more than earlier ones. This can be graphically depicted with decreasing transparency on the mapped points and a weighted average to the "centroid."

In general, the DTWRadar appears both flexible enough to support the visualization of different properties while still being easy to quickly interpret. It is likely that even if different alignment techniques [96, 97] are used, this visualization is still informative.

Because we constructed a DTWRadar for each pair of QES's, we have a pair of points

that describes the relationship between the series. One could easily envision support for searching this space of points. In fact, we have created a small application that allows users to choose one source as the origin, and scaling those points representing the relative location of other QES's or TES's. Put another way, all DTWRadars were overlaid with one chosen source as the origin. By clicking and dragging a region of this meta-radar, the system finds all queries or topics that display a certain relationship. All matches are displayed and returned to the user for further analysis.

Clustered results

A natural question is how different the data streams are from each other for all QES's. For example: on average, does a QES from a blog source (i.e., posts mentioning a string) lead the QES for AOL? Figure 6.12 presents a set of overlaid DTWRadar centroids with MSN as the origin. To test this, the set of those QES's determined to have a high correlation was selected from the experimental list created in the correlation analysis. MSN was chosen because we are guaranteed to have data for MSN for any given query, though we can create these figures for any relationship. The centroid-of-centroids is not rendered as it sits fairly close to the origin in all cases. The display illustrates the spread of all TES's relative to the MSN query logs and indicates the distribution of lag times and magnitude differences.

One obvious effect of our chosen weighting scheme for the TV dataset is that all centroids appear in quadrants III and IV (lower magnitude queries). Despite the fact that TV.com is popular, the reality is that limited votes are distributed among many shows (i.e., few votes per show). Additionally, because of the tremendous number of blog posts per day, there is a significant reduction in the magnitude of each individual blog QES. Many blog posts are not in English or do not have anything to do with external events (i.e., the personal diary blog). In the future it is likely worth eliminating blogs that never match against any topic of general interest.

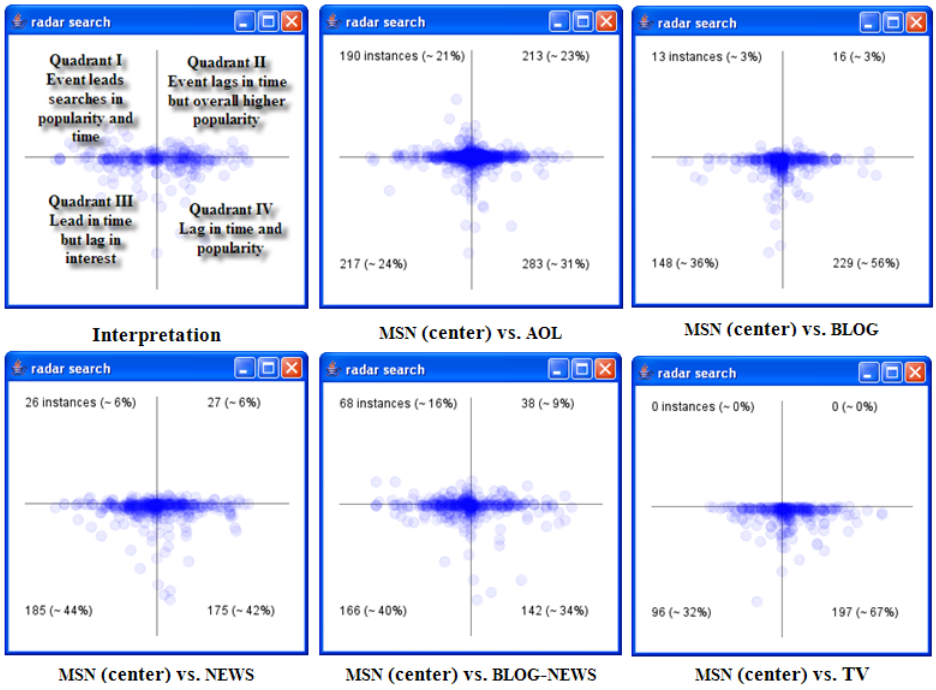


Figure 6.12: The cumulative radar positions for all queries against the MSN dataset. Only queries with cross-correlation $\geq .7$ were considered.

6.5 Behavioral Trends

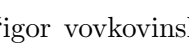
The radar views provide a global sense of the relationship of the responses to an event in one stream versus another. Though, as seen in Figure 6.12, many of the centroids cluster in the middle, those that do not may provide insight into which sources, topics, and behaviors are predictive or predictable. For example, one might expect a ramp-up much earlier for expected events in the query-logs followed by a burst of posting and news activity in the BLOG and NEWS datasets around the time of the actual event. To understand which behavioral patterns are common, it is useful to classify and characterize the relationships between sources and topics.

From the set of highly-correlated queries, 244 were selected at random. Of these, 216 (unambiguous) were manually annotated with various categories (corporate websites, financial, TV shows, in the news, etc.). Those queries for which the objective of the search appeared

ambiguous, and no specific category could be assigned, were discarded. Queries were allowed multiple categories. For example "nicole kidman" was tagged as both a celebrity and for being in-the-news (her engagement to Keith Urban was announced in mid-May). Similarly, "world cup" was tagged as both sports and in in-the-news as the lead-up and preparation for the event generated news stories.

Below, a number of general trends and issues, some hypothetical, are described and motivate them by a few characteristic examples:

News of the weird: Certain events have the property that they are obscure and yet have the potential to become viral and capture the imagination of many individuals. This type of topic was named "news of the weird," (in fairness, not all obscure topics are actually weird, but they may not be front page news). Bloggers tended to be much better at anticipating the eventual popularity of such stories. For example:

- The blogger discussion of "igor vovkovinskiy," ($QES^{\text{BLOG}} =$ ), a 7'8" man who received a custom set of size 26 shoes and was featured in an Associated Press article, preceded queries of his name ($QES^{\text{MSN}} =$ ).

- Similarly, the posting behavior around "burt rutan" ($QES^{\text{BLOG}} =$ ), an aerospace engineer who criticized a NASA decision in early May was discussed in blogs earlier and queried later ($QES^{\text{MSN}} =$ )

- The response to the sinking of the "uss oriskany" ($QES^{\text{BLOG}} =$ ) was nearly identical in shape to MSN, but bloggers appeared to go through the cycle 3 days earlier.

Anticipated events: Blog postings were not nearly as predictive when it came to anticipated events. For example, in the approach to the "preakness" horse race ($QES^{\text{MSN}} =$  vs $QES^{\text{BLOG}} =$), many users queried for information while few bloggers produced posts. This type of reaction is believed to be due to the social nature of blogs, and the reward for finding "new" information, making bloggers unlikely to discuss widely

known events. As the time of the event approaches and new information becomes available, bloggers will begin to react.

Familiarity breeds contempt: A related effect was observed when comparing news to search behavior. Though a topic is still newsworthy, it may be so familiar to the average user that they will not actively seek new information about the topic. One such event is the "enron trial" ($QES^{\text{NEWS}} =$ ). Though the trial was still a topic in the news, searchers ignored it until a verdict was passed leading to a sudden rise in search behavior ($QES^{\text{MSN}} =$ ). Such behaviors form an interesting direction for future research as it may be useful to differentiate "significant" events from "standard" news. With additional news sources the difference is readily identifiable by the number of sources covering the story. This may be detectable with only a few news sources by noticing the time difference between the actual event and the time the story was published, as well as the way in which the source conveys the story (e.g., a "breaking news" headline might be a good signal).

Filtering behaviors: One unanticipated consequence of choosing a highly specialized source such as the TV.com dataset was that it was effective in filtering news about certain classes of topics, in this case TV shows as well as actors and actresses contextualized to the shows they star in. For example, "[M]ischa barton" (an actress on a popular TV show, $QES^{\text{TV}} =$ ) saw a lot of overall activity due to the popularity of the show, with a sudden overwhelming burst of activity late in the month on MSN ($QES^{\text{MSN}} =$ ) anticipating the death of her character in the season finale (the rightmost bump on the TV curve). By creating more specific labels of "source" (for example, by partitioning the blog dataset into communities), it may be possible to filter and classify certain topics by their behavioral correlation to these specific sources.

Elimination of noise: While one may want to break apart a source into highly specific sub-sources, it may also be useful to consider the combination of sources to reduce noise. Because news stories in the BLOG-NEWS dataset are weighted by blogging behavior, and because bloggers frequently reference news stories that they find worth blogging about, the events that are predictive in time and magnitude track the blogging behavior very closely. For example the behavior for "uss oriskany" is nearly identical in both datasets

(cross correlation of .988 at 0 days delay). Because blogs are highly variable in quality and quantity of information, a blogger may simply make a reference to an article without actually discussing the content. Using the content of the pointer, it is possible to identify those bloggers which accurately capture the topic or automatically augment posts to include such information.

Fifty-two of the 216 queries were tagged as in-the-news. Because the 216 were suggested only when there was a high correlation to at least one other data stream, this may provide a mechanism for finding interesting events. That is, if a given topic has correlated appearance in a number of data sources then it has a high likelihood to be a response to a "news" event.

Correlation versus causation: One of the fundamental problems with trying to understand cause and effect is distinguishing between correlation and causation. While two QES's may both react to an event over time, the root cause of their reactions may be different. For example, a number of TV shows featured cast members and spoofs of the movie "poseidon" ($QES^{\text{TV}} =$ ) at the time of the movie's opening. However, search users were frequently querying for the movie beforehand ($QES^{\text{MSN}} =$ ). While there is an obvious correlation in the two streams due to the upcoming release of the movie, search users reacted to the marketing of the release in addition to the actual release. Thus, an "explanation" of the behavior of search users requires taking into account a combination of events (e.g., a marketing/advertising event source and a movie release database) and may require more complex models.

Portal differences: Today's search engine front pages are not simply an empty form box. Search engines are portals with news, entertainment, many other features. The information available on the front page may naturally lead to changes in behavior. While it may be impossible to dismiss these differences as being caused by demographic variations, a few were likely caused by what information a portal chooses to present to its user's. For example, AOL users led in magnitude and time in their response to "mothers day" ($QES^{\text{AOL}} =$ ) and "ashley flores" (a kidnap victim, $QES^{\text{AOL}} =$ ). On the other hand, MSN users led in queries for "flight 93 memorial" ($QES^{\text{MSN}} =$ ), and in general appeared to respond to news a little earlier and with more queries (e.g., "katie couric[s]" move to CBS at the end of May, $QES^{\text{MSN}} =$  , and the death

of the boxer "floyd patterson," $QES^{\mathrm{MSN}} =$ ). In the future, capturing the front pages of different portals may help to more accurately identify these differences.

6.6 Summary

This chapter described a large scale comparison and correlation of multiple Internet behavioral datasets. In addition to describing a number of relationships between different communities and Internet-based media, the work generated a number of techniques and tools that are broadly applicable to the various problems inherent in Dynamic Web analysis:

- It is sometimes necessary to utilize one stream as a proxy for another. This may mean approximating behavior streams from content streams or vice versa. The study illustrates a number of such conversions.

- Because there are datasets that are easy to obtain, and datasets that are more difficult, it is important to recognize situations in which the "cheaper" dataset might correlate significantly with the more "expensive" one–allowing one to replace one with the other.

- Many systems and services rely not on the what has been, or what is, but rather the what will be. Systems, such as the DTWExplorer, can be utilized to identify leading indicators that would allow systems to respond *before* some change occurs in the content or behavior stream.

Chapter 7

Zoetrope

Earlier chapters demonstrated the scale of change in the World-Wide Web. The study of revisitation patterns also showed the *temporal sensitivity* of individuals as they attempt to access and monitor changing content. This sensitivity may be due to trying to capture specific state changes or the complete history of a document. Both situations frequently require users to utilize the revisitation and monitoring behavior described in Chapter 4. However, if a user is unable to anticipate their own data needs or revisit at a sufficiently fast rate, much of the Web will vanish irretrievably. While one might want an automated system to assist in temporal Web extraction, unfortunately interfaces and datasets are temporally *insensitive* and do not support access to ephemeral Web data.

News stories, Wikipedia articles, social network profiles, and corporate pages are continually updated. Automatically generated pages, whether driven by back-end inventory-control systems (e.g., travel and e-commerce sites) or real-time sensor streams (e.g., traffic, weather, and webcams) may change even more quickly. In contrast, most Web usage is restricted to the Web's most recent state. People access the Web through browsers that, with the exception of simple caching, provide access only to the Web's present state. Although search engines provide quick access to a vast corpus, retrieval is again limited to the most recent snapshot recorded by the underlying crawler. The headlines on today's CNN homepage will be gone tomorrow, and yesterday's price for a book on Amazon is likely irretrievable today. Even if historical data still exists, it may be aggregated into averages or split among

many webpages. For a person who does not have access to historical data, or would find it difficult or impossible to recreate, the Web is primarily one-dimensional, containing only one version of any page. The current lack of support for temporal access to the Web makes it difficult or impossible to find, analyze, extract, and compare historical information.

While a service like the Internet Archives (http://www.archive.org) provides a useful function by storing a historical record of pages, it fails to capture these pages at a resolution that is useful for anything but very general analysis or allow for convenient search or browsing. A great amount of temporal information changes many times a minute (e.g., stocks or environmental sensor readings) or is aggregated into longer periods (e.g., final scores of a game, weekly Nielsen reports, or daily Amazon sales rank). As fine grained historical data is unavailable, users are unable to view previous versions of a page at specific times, cannot easily identify changes, and are unable to extract historical data for analysis. If they are lucky, users may find the data they are looking for within some archive or through a "deep-web" search but this may take significant time and effort.

As a solution, this chapter offers *Zoetrope*, a visual query system for extracting information from the historical Web from within the context of the Now Web. Zoetrope is the result of collaboration with Mira Dontcheva, Dan Weld, and James Fogarty ([2]).

7.1 Related Work

In response to the need for dealing with temporal data, a number of solutions have emerged in the research community. Arguably one of the first, the AT&T Difference Engine [57] archived pages and displayed differences between consecutive versions of pages. Because HTML is semi-structured, researchers in the database and data mining communities have expanded ideas on XML differencing to HTML/Web differences (extensively surveyed in [72] and in Section 8.4). Zoetrope is not explicitly designed for comparing two page versions to each other (though this can certainly be implemented). Rather, Zoetrope is targeted at *many* instances of a given page.

Another related area of research includes Web-clipping systems (see the survey in Sections 8.4 and 8.5) that allow people to monitor Web page regions of interest [74, 142, 146].

These systems are typically focused on tracking future changes rather than working with previous versions of Web pages. These efforts have also inspired commercial efforts, such as Internet Explorer 8's *WebSlices*. Multi-page clipping and mashup/extraction applications (e.g., [55, 56, 78, 82]) are valuable tools for extracting and combining data from the present Web, but are not designed for historical versions. Of notable exception are programming by demonstration (PBD) systems such as Koala [111], which must take into account the changing Web to preserve the validity of macros and scripts going forward. Zoetrope implements Web-clipping features in a number of visualizations (time lines, clusters, etc). This architecture is well-suited to such applications and we hope to investigate them further in the future.

A second major area of research has been the visual display of temporal differences in information. Certain solutions are targeted at news while others act more generally to represent differences between versions of Web pages. These include techniques for converting textual information into time series [62], spatio-temporal visualizations [138], and other difference visualizations (e.g., [112, 114]).

Architecturally, a number of database visualization systems are related to the stream and operator design in Zoetrope. Particularly, DEVise [115] was originally constructed to operate on streams of data that required visualization. Similarly, Polaris [145] operates to visualize relational data. In both, the operators provided by the system are primarily targeted at the rendering step (deciding how graphs, charts, and other visualizations should be constructed). Zoetrope instead focuses on unstructured temporal Web content from which data can be extracted for these types of visualizations. Other systems-focused research that could support future additions to Zoetrope include work on automation and scalability issues of large-scale Web archiving (e.g., [32]) and full-text search (e.g., [24]). Such solutions are described in Section 8.7.

Finally, although these approaches have not been targeted at temporal Web data, we draw inspiration from the Video Cube [99] and Magic Lenses work [26]. The former allows video stream frames to be layered and "sliced" to find an abstraction of the video. In the latter, a magic lens is a widget that can be placed directly on a document to illuminate the underlying representation while maintaining the visual context of the document.

To illustrate the difficulty of answering certain questions given current tools, consider the following simple example.

A Simple Example

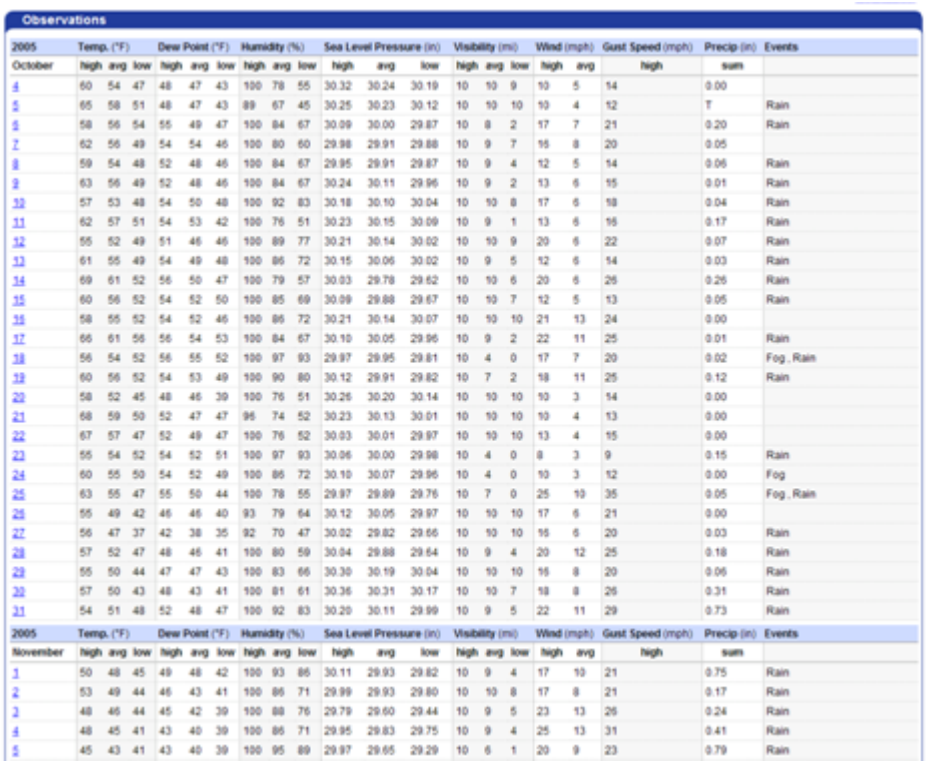
To demonstrate the problem, consider the following simple example: a student wanting to find the impact of Seattle temperature on bridge traffic. To start, the student attempts to find Seattle weather conditions for the past few years. The data is not private and one might think it would be readily accessible. A query for "Seattle historical weather" on Google produces various hits, the first of which is the page for the local news station (Figure 7.1). The page itself contains the current weather and links to the Weather Underground but does not have historical data itself. The Weather Underground page again has various current weather statistics and projections, but requires going to another page for actual historical data. The historical data on this page defaults to a 10 minute view of the current day. If the user happens to notice the tabs at the top of the page she might discover various aggregations of the data, but at best we can only get a year's worth of statistics at once (Figure 7.2). Each extra year requires an additional click and manual extraction of the data. Now let us consider an alternative solution, a system that allows the user to select any piece of information on the page and to pull out that information for historical instances of the page. If the student had such a tool, she would have been able to stop at the first page to find the information they needed. Selecting the current weather on any of the pages she visited and clicking "extract" would have generated a time series of Seattle's historical weather.

Expanding the example further, the student would like to find the travel times from Seattle to Redmond over the 520 bridge on days when the temperature fell below freezing. Again, such data should be public but a query for "Seattle to Redmond travel times historical" shows no useful results (the top hit is a descriptive page of travel time by hour and the remaining nine are news articles or unrelated links). She has slightly better luck by asking for "Seattle to Redmond travel times," finding a page that has the current travel times (Figure 7.3). Unfortunately, the only historical data on this site are maps that again

Figure 7.1: The top search result for "Seattle historical weather" contains the current weather information and a link to the Weather Underground.

must be selected one at a time. For all intents and purposes the data the student wants is gone from the Web. The student might instead prefer a system that would allow her to select the number on the travel time page for Seattle to Redmond and extract the data. The student could combine the two datasets (weather and traffic) into a spreadsheet and work from there. Additionally, this system could provide the ability to "connect" these two extractions, conditioned on the temperature falling below 32 degrees (the student may, of course, also select weather from non-freezing days at random to perform a statistical comparison). The results would be instantly filtered and work with the spreadsheet could be reduced.

Finally the student would like to identify any public notices in local Seattle newspapers regarding bridge closures from around these "freezing" dates. Searching through Google News archives would still require a great deal of manual extraction. However, in the idealized



Observations

2005	Temp. (°F)			Dew Point (°F)			Humidity (%)			Sea Level Pressure (in)			Visibility (mi)			Wind (mph)		Gust Speed (mph)	Precip (in)	Events
October	high	avg	low	high	avg	low	high	avg	low	high	avg	low	high	avg	low	high	avg	high	sum	
4	60	54	47	48	47	43	100	78	55	30.32	30.24	30.19	10	10	9	10	5	14	0.00	
5	65	58	51	48	47	43	89	67	45	30.25	30.23	30.12	10	10	10	10	4	12	T	Rain
6	58	56	54	55	49	47	100	84	67	30.09	30.00	29.87	10	8	2	17	7	21	0.20	Rain
7	62	56	49	54	54	46	100	80	60	29.98	29.91	29.88	10	9	7	16	8	20	0.05	
8	59	54	48	52	48	46	100	84	67	29.95	29.91	29.87	10	9	4	12	5	14	0.06	Rain
9	63	56	49	52	48	46	100	84	67	30.24	30.11	29.96	10	9	2	13	6	15	0.01	Rain
10	57	53	48	54	50	48	100	92	83	30.18	30.10	30.04	10	10	8	17	6	18	0.04	Rain
11	62	57	51	54	53	42	100	76	51	30.23	30.15	30.09	10	9	1	13	6	16	0.17	Rain
12	55	52	49	51	46	46	100	89	77	30.21	30.14	30.02	10	10	9	20	6	22	0.07	Rain
13	61	55	49	54	49	48	100	86	72	30.15	30.06	30.02	10	9	5	12	6	14	0.03	Rain
14	69	61	52	56	50	47	100	79	57	30.03	29.78	29.62	10	10	6	20	6	26	0.26	Rain
15	60	56	52	54	52	50	100	85	69	30.09	29.88	29.67	10	10	7	12	5	13	0.05	Rain
16	58	55	52	54	52	46	100	86	72	30.21	30.14	30.07	10	10	10	21	13	24	0.00	
17	66	61	56	56	54	53	100	84	67	30.10	30.05	29.96	10	9	2	22	11	25	0.01	Rain
18	56	54	52	56	55	52	100	97	93	29.97	29.95	29.81	10	4	0	17	7	20	0.02	Fog , Rain
19	60	56	52	54	53	49	100	90	80	30.12	29.91	29.82	10	7	2	18	11	25	0.12	Rain
20	58	52	45	48	46	39	100	76	51	30.26	30.20	30.14	10	10	10	10	3	14	0.00	
21	68	59	50	52	47	47	96	74	52	30.23	30.13	30.01	10	10	10	10	4	13	0.00	
22	67	57	47	52	49	47	100	76	52	30.03	30.01	29.97	10	10	10	13	4	15	0.00	
23	55	54	52	54	52	51	100	97	93	30.06	30.00	29.98	10	4	0	8	3	9	0.15	Rain
24	60	55	50	54	52	49	100	86	72	30.10	30.07	29.96	10	4	0	10	3	12	0.00	Fog
25	63	55	47	55	50	44	100	78	55	29.97	29.89	29.76	10	7	0	25	10	35	0.05	Fog , Rain
26	55	49	42	46	46	40	93	79	64	30.12	30.05	29.97	10	10	10	17	6	21	0.00	
27	56	47	37	42	38	35	92	70	47	30.02	29.82	29.66	10	10	10	16	6	20	0.03	Rain
28	57	52	47	48	46	41	100	80	59	30.04	29.88	29.64	10	9	4	20	12	25	0.18	Rain
29	55	50	44	47	47	43	100	83	66	30.30	30.19	30.04	10	10	10	16	8	20	0.06	Rain
30	57	50	43	48	43	41	100	81	61	30.36	30.31	30.17	10	10	7	18	8	26	0.31	Rain
31	54	51	48	52	48	47	100	92	83	30.20	30.11	29.99	10	9	5	22	11	29	0.73	Rain
2005	Temp. (°F)			Dew Point (°F)			Humidity (%)			Sea Level Pressure (in)			Visibility (mi)			Wind (mph)		Gust Speed (mph)	Precip (in)	Events
November	high	avg	low	high	avg	low	high	avg	low	high	avg	low	high	avg	low	high	avg	high	sum	
1	50	48	45	49	48	42	100	93	86	30.11	29.93	29.82	10	9	4	17	10	21	0.75	Rain
2	53	49	44	46	43	41	100	86	71	29.99	29.93	29.80	10	10	8	17	8	21	0.17	Rain
3	48	46	44	45	42	39	100	88	76	29.79	29.60	29.44	10	9	5	23	13	26	0.24	Rain
4	48	45	41	43	40	39	100	86	71	29.95	29.83	29.75	10	9	4	25	13	31	0.41	Rain
5	45	43	41	43	40	39	100	95	89	29.97	29.65	29.29	10	6	1	20	9	23	0.79	Rain

Figure 7.2: Part of a page containing historical weather information for Seattle on the Weather Underground page.

system the student simply lists a few local news pages of interest. The system automatically identifies "blocks" of changing content, extracts those blocks that mention the 520 bridge, and outputs unique stories from each day. Combining the results from weather, traffic, and news, the system generates a timeline visualization, allowing the student to quickly get a sense of travel times in these situations.

An Alternative

Zoetrope is a system designed to deal with cases such as the one described above, allowing users to interact with the extended history of the World Wide Web. With Zoetrope, one can explore past versions of entire pages or place *lenses* (Figure 7.4) on any part of a Web page to scan backwards through time (e.g., to see all previous top headlines on CNN's

State Route/ Interstate	Route Description	Distance (miles)	Average Travel Time (minutes)	Current Travel Time (minutes)	Via HOV (min.)
167	Auburn to Renton	9.8	12	11	10
405	Bellevue to Bothell	9.7	25	21	11
405 5	Bellevue to Everett	22.6	52	37	25
405 5	Bellevue to Federal Way	24.7	60	46	29
405 90	Bellevue to	9.5	16	16	11

Figure 7.3: Current travel times from Seattle to Redmond

page). Lenses can be *filtered* with keyword and other queries or *bound* together to explore correlated historical data. To support analysis of this data, Zoetrope provides a number of *visualizations* for the extracted content. Because temporal Web data is not currently available at fine granularity, Zoetrope also includes a set of crawling and indexing technologies for obtaining high-resolution temporal data. We use an experimental dataset of 250 Web pages from a range of domains and websites (news, sports, weather, entertainment, etc.) crawled at one-hour intervals for more than a month.

By providing a variety of lenses, filters, and visualizations that can be combined and composed in powerful ways, Zoetrope presents a new way to think about historical Web data. As a tool, Zoetrope allows anyone to explore the Web over time. Furthermore, Zoetrope provides developers a mechanism for easily exploring new applications and interactions. The contributions of this work include:

- a novel visual programming toolkit and a set of interactions for rapidly building and testing temporal Web queries,

- a semantics for temporal data streams,

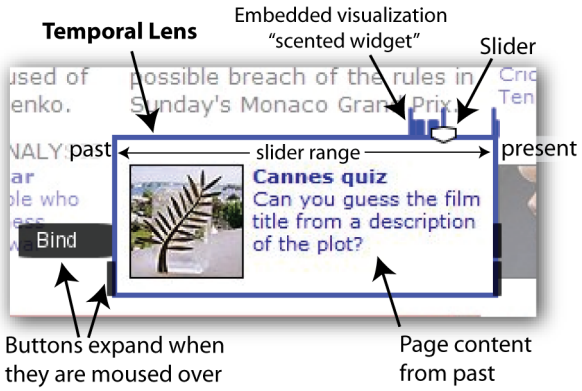


Figure 7.4: Temporal lenses include a slider for accessing previous versions, a scented widget that shows the location of available content, and buttons for creating visualizations and binding lenses together.

- a set of recomposable operators to manipulate temporal data streams,
- indexing structures for fast processing and interaction with Web content over time,
- and a unique dataset collected by a new crawler design.

7.2 The Zoetrope Interface

Zoetrope presents Web pages in a zoomable canvas, within which a person can simultaneously explore past versions of any number of Web pages. To browse previous versions, one can move the slider at the bottom of the display or alternatively place one or more lenses on different parts of a Web page. For example, consider a related (but much simpler) case to the traffic/weather example described above: Ed, a potential Zoetrope user, who at lunchtime becomes anxious about his evening commute. Although he knows the traffic page, the page only displays the current traffic conditions. Without Zoetrope, Ed might be able to find older traffic flow maps, but this search would likely require considerable effort. Instead, Ed can adjust the Zoetrope slider and instantly view previous versions of any page. To focus on a specific part of a Web page, Ed selects a *visual* lens from the lens menu and

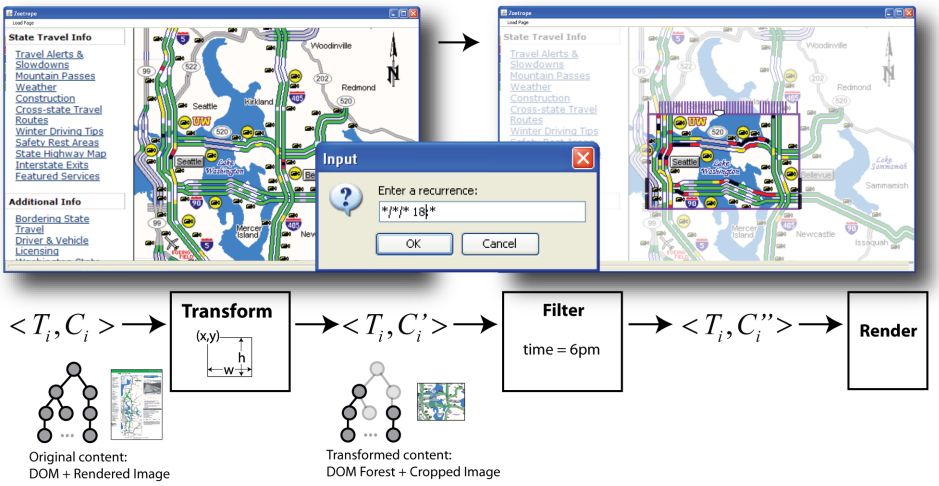


Figure 7.5: The left browser window shows a Web page for Seattle traffic conditions. A person draws a lens on the region of interest (the bridges crossing Lake Washington) and specifies a temporal filter, selecting only content fetched between 6-7pm (i.e., 18-19 in 24 hour format). The browser window on the right shows the resulting lens, in purple, with tick marks on top indicating each instance of 6pm (the page is faded in the figure to emphasize the lens). Below the browser windows we show the Zoetrope transform and filter pipeline that enable this interaction.

draws directly on the Web page to specify which part of the traffic map is of interest (see Figure 7.5). He also adds a *temporal filter* to the lens, because he is only interested in traffic information at 6pm. Zoetrope creates a visual lens for Ed's selection that includes a slider, a scented widget [169], and several buttons for further refinement. By moving the slider, Ed is able to shift backwards and forwards day-by-day to review past congestion. Buttons to the left and right of the lens allow Ed to synchronize the entire Web page to a selected time, to close the lens, to bind multiple lenses together, or to generate a visualization based on the selected content. Throughout this task, Zoetrope allows Ed to use a Web page he is familiar with, in its present version, to contextualize his query into the past. Although Zoetrope displays the *rendered* Web page as an image, the system maintains the interactivity of the *live* Web page. Clicking on a hyperlink opens a browser for the present version of the hyperlink target (or the historical version corresponding to the slider selection, if it exists within Zoetrope).

7.3 System Architecture

The Zoetrope architecture, summarized in Figure 7.6, includes a *Web crawler* that collects data, a set of *databases* to store collected content, and an *interface* that provides access to the stored data. The crawler collects data at regular intervals and associates each crawled page with a time. Each page is stored in two databases: (1) as XML providing Zoetrope access to the structure and text for indexing and extraction, and (2) as an image providing Zoetrope nearly instant access to the page's visual appearance. When a person views a page in Zoetrope, each instance of the crawled page and associated time are loaded as a *content stream*. Recall, a content stream is a sequence of tuples, $\langle T_i, C_i \rangle$ (where C_i is a *content item* and T_i is the time when that content was sampled). When a person creates a lens or some other visualization, Zoetrope generates a sequence of *operators*, which process

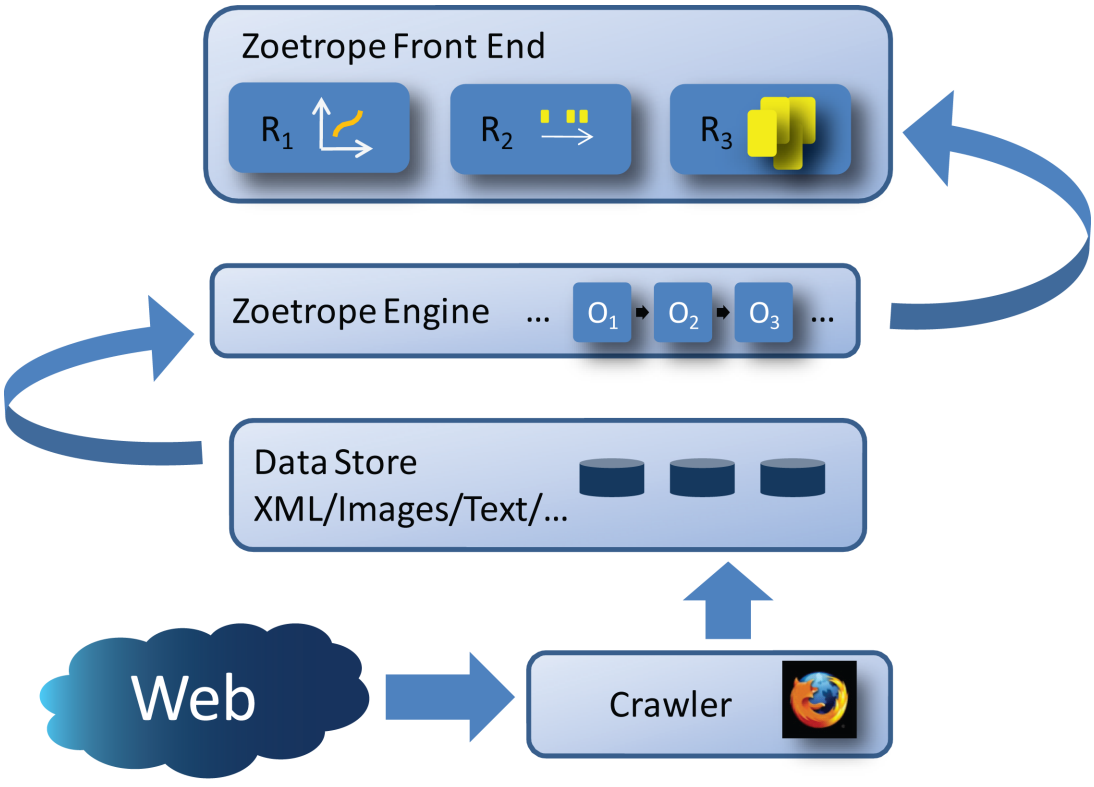


Figure 7.6: An overview of Zoetrope's architecture.

data with transforms and filters, and routes the content stream through them. Finally the stream is displayed using *renderers*. Currently, renderers draw into the *lenses* themselves or into separate *visualizations*. Additionally, Zoetrope can export content streams to external systems, such as Google Spreadsheets (*docs.google.com*).

Figure 7.5, for example, shows a graphical representation of the operators corresponding to Ed's visual lens: a transform consumes the content stream, it extracts a small cropped image and a small part of the document text, a filter decides which instances occur between 6 and 7pm, and the renderer uses the slider location to display a specific instance from this stream.

7.4 Temporal Lenses

A lens allows a person to select some content and track it over time. Although there are various flavors of lenses, their creation and use is nearly identical. A person creates a lens simply by drawing a rectangular area on the Web page surface. In the underlying semantics of the Zoetrope system, the creation of a lens produces a parametrized transform operator that acts on the original page content stream, an optional filter (or set of filters) that processes the transformed stream, and a renderer that displays the historical data in the context of the original page (see Figure 7.5). The specific selections of transforms and filters depends on the lens type. Zoetrope currently provides three types of lenses for tracking different types of information: visual, structural, and textual.

7.4.1 Visual Lenses

The simplest Zoetrope lens is the visual lens. To create this type of lens, a person specifies a region on the original page (e.g., a portion of the traffic flow map in Figure 7.5). The specification produces a lens with a slider. The slider is parametrized on the width of the lens and the range of data being displayed. As the slider moves, the lens renders the corresponding data (i.e., the content that existed at that coordinate on the Web page at the time the slider is set to).

More formally, the lens specifies a rectangle R (coordinate of origin, width, and height),

which is used to parametrize a cropping transform. The cropping transform consumes a stream of Document Object Model (DOM) trees and rendered page images and outputs a stream of DOM forests (the subset of the original DOM that is rendered within the rectangle) and cropped images. Recall that a content stream is represented as a set of tuples, $\langle T_i, C_i \rangle$, where C_i is the content and T_i the time. In this case, C_i contains the document (as a DOM tree), the CSS templates, images, and flash files (call these $C_i.\text{DOM}$, $C_i.\text{CSS}$, etc.) As an optimization, the tuple content also contains the rendered representation of the page, $C_i.img$. Thus, a visual lens converts each tuple of the form $\langle T_i, C_i.\text{DOM}, C_i.img \rangle \rightarrow \langle T_i, \text{SELECT}(C_i.\text{DOM}, R), \text{CROP}(C_i.img, R) \rangle$. The generated stream is passed to the lens renderer, which displays the cropped image corresponding to the selected time.

7.4.2 Structural Lenses

Not all Web pages possess sufficient stability for a visual lens. Slight shifts in rendering or more significant movement of elements can cause distracting jumps when a person moves the lens slider. To counter this effect and to allow for more precise selections, Zoetrope provides structural lenses. Structural lenses are created in the same way as visual lenses, by drawing a rectangle around an area of interest, but they track selected HTML content independent of visual position.

Formally, R denotes the rectangle that encloses the DOM elements that the user wants tracked at lens creation time, s. The lens creation process finds the XPath expression, P, that describes the upper-leftmost element rendered within R. When the path P is applied to the DOM at time s, $\text{XPATH}(P, C_s.\text{DOM})$, the result is some DOM element, E_s. In this syntax, let $E_s.x$ and $E_s.y$ be the origin coordinate for the crop *in all other versions* and are able to track selections more accurately. A precise specification of R is not necessary. Zoetrope finds all DOM elements contained in the selection and resizes the rectangle (now R'), visually snapping to the selected elements.

The transform works on tuples in the following way:

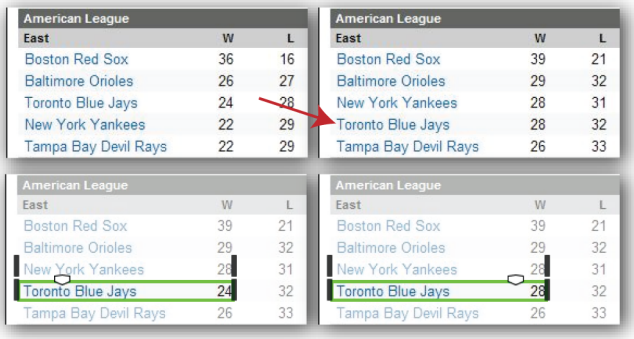


Figure 7.7: A textual lens can track content regardless of where it appears on the page, such as the Toronto Blue Jays, which over time shift in the ordered list above.

$$\langle T_i, C_i.\text{DOM}, C_i.img\rangle \rightarrow \langle T_i, \text{SELECT}(C_i.\text{DOM}, R'),$$
$$\text{CROP}(C_i.img, R'' = (E_s.x, E_s.y, R'.width, R'.height))\rangle$$

As before, this output stream of tuples is sent to a lens renderer which displays the image for the selected time.

7.4.3 Textual Lenses

Visual and structural lenses are dependent on certain types of Web page stability. A visual lens relies on stability of the rendering, whereas a structural lens takes advantage of structural stability. Both are reasonable in many scenarios, but it is also worth considering selections based on unstable or semi-stable content. For example, consider tracking a specific team in a list of sports teams that is ordered by some changing value, such as rank (see Figure 7.7). As teams win and lose, the team of interest will move up and down the list. Specifying a rectangle at $(100, 400)$, or the fourth row in the list, will not work when the team moves from this position. To address this type of selection, we introduce the notion of a textual lens, which tracks a textual selection regardless of where the text is located on the page. A textual lens can track exactly the same string (e.g., a blog story) or approximately the same string (e.g., a sports team name with a score, where the score changes from time

to time).

In its most general form, a textual lens tracks arbitrary text regardless of where it appears on the page, which DOM elements contain the text, and the size of those elements. This generalization is unfortunately too computationally intensive, even in scenarios where the text is unchanging, and the problem becomes intractable for an interactive system. To make textual lenses interactive, the search space is restricted by making use of the document structure. Textual lenses often track items that appear in tables, lists, or structurally similar sub-trees (e.g., posts in a blog all have similar forms). Zoetrope takes advantage of this structural similarity by generalizing the paths of selected DOM elements. For example, a path of the form `/html[1]/ body[1]/table[4]/tr[5]/td[2]` can be generalized to `/html/body/table/tr/td`. Applying this more general XPath to the DOM returns all elements that match this pattern, instead of just the specific item that was selected. In practice, this transformation overgeneralizes the selection and CSS class attributes are useful for constraining the patterns to eliminate unlikely DOM elements. For example, the headline and body of an article may both match the pattern above (they are both in a table), but the headline will have "headline" as its class attribute while the article will have "article" as its class attribute. Zoetrope, therefore, adds class attributes to the generalized path (e.g., `/html/body/table/tr/td [@class='headline']`) to constrain the resulting set of similar items. Given this generalized path, P', the transform produced by the lens identifies all matches at a particular time by applying the path to the DOM (i.e., $\textsc{xpath}(P', C_i.\textsc{dom}) = \{D_{i,1}, \ldots, D_{i,n}\}$).

Next, Zoetrope finds the closest matching element for each time using a similarity metric, SIM, which calculates the textual similarity of the originally selected content, D_s, to each of the extracted elements. For each time, i, Zoetrope finds the $D_{i,best}$ that is most similar to D_s. Zoetrope currently uses the Dice coefficient to compute similarity, which calculates the overlap in text tokens between two pieces of text (i.e., $\textsc{sim}(A, B) = 2 * |A \cap B|/(|A| + |B|)$, where A and B are sets of words). When the initial selection is a forest rather than a tree of DOM elements, Zoetrope generates the cross product of each generalized path. In other words, given two generalized paths, P^1 and P^2, Zoetrope calculates

$$\begin{aligned}&\textsc{xpath}(P^1, C_i.\textsc{dom}) \times \textsc{xpath}(P^2, C_i.\textsc{dom}) = \\ &\{D^1_{i,1}, \ldots, D^1_{i,n}\} \times \{D^2_{i,1}, \ldots, D^2_{i,m}\} = \\ &\{\{D^1_{i,1}, D^2_{i,1}\}, \{D^1_{i,1}, D^2_{i,2}\}, \ldots, \{D^1_{i,n}, D^2_{i,m}\}\}\end{aligned}$$

These groups are filtered to find those that satisfy the initial hierarchical relationship of the selection. For example, if two sub-trees in the original selection have a hierarchical distance of 5, the pairs tested in the generalized path should have the same distance. An alternative to hierarchical distance is visual distance (i.e., both elements should fit inside a rectangle of the size originally specified by the lens). Similarity in such cases could be determined as a combination of similarity scores. The initial implementation simply merges all the text from a group into one text block before calculating the similarity. A threshold can be set as input to the transform, restricting returned elements to those sufficiently similar to the input. This threshold can be varied depending on how much the content is likely to change from the original selected version.

To summarize, a textual lens is created by the selection of a region, the automatic conversion of the selection to a generalized path, and the use of this path as a parameter to the transform. The transform works on the DOM and images for each time, generating an image cropped to the upper-left coordinate of the best matching element set ($D_{i,best}$) and sized to the width and height of the original selection, R.

$$\begin{aligned}\langle T_i, C_i.\textsc{dom}, C_i.img\rangle \rightarrow \langle T_i, D_{i,best}, \textsc{crop}(C_i.img, R'' = \\ (D_{i,best}.x, D_{i,best}.y, R'.width, R'.height))\rangle\end{aligned}$$

As before, the output is passed to a lens renderer that displays the appropriate cropped image according to the time selected with the slider.

Taken together, the visual, structural, and textual lenses form the basis of Zoetrope's interactions. They are sufficient for tracking and extracting a wide range of data types within a page, thus enabling Zoetrope's filters and visualizations.

7.4.4 Applying Filters to Lenses

Given the large volume of data encoded in a content stream, it is natural to want to focus on specific information of interest. Zoetrope uses *filters* to provide this capability. If we consider the evolving state of content as a database relation, then a filter is simply a select operation in relational algebra (e.g., *select* tuples *where* some condition). There are a number of useful conditions for selection, including:

- **Filtering on Time:** One may wish to see the state of one or more streams at a specific time or frequency (e.g., 6pm each day).

- **Filtering on a Keyword:** The selection condition may also refer to C_i, the content half of the tuple $\langle T_i, C_i \rangle$. If C_i contains text, then keyword queries may apply. For example, one might only be interested in headlines that contain the word "Ukraine" (Figure 7.8).

- **Filtering on Amounts:** One may also select content using an inequality and threshold (e.g., $> k$). If the content is numeric and the inequality is satisfied, then the tuple is kept; otherwise it is filtered. Similarly, one can select the maximum or minimum tuple in a numeric stream.

- **Duplicate Elimination:** It may also be useful to select only those tuples whose content is distinct from content seen earlier in the stream.

- **Compound Filters:** Logical operations (conjunction, disjunction, negation, etc.) may be used to compose more complex selection criteria.

- **Trigger Filters:** An especially powerful filter results when one stream is filtered according to the results of another stream's filter. For example, Ed can filter the traffic page using a conjunction of the 6pm time constraint and a trigger on the ESPN page for the keyword "home game." This is further considered when discussing lens binding.

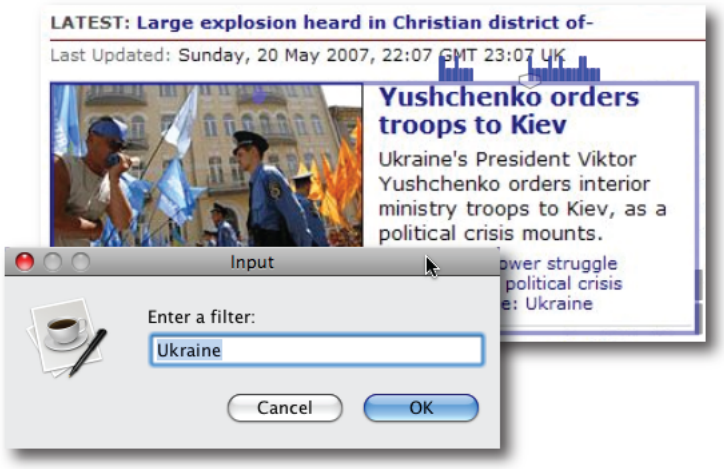


Figure 7.8: This lens only displays articles that include the word "Ukraine." The scented widget above the slider indicates the location of such articles.

Because filtering is useful for many tasks, it is provided as an option whenever a visual, structural, or textual lens is applied. When selecting a region with filtering enabled, a lens is created based on the underlying selection and a popup window asks for a constraint to use in the filter, such as a word or phrase. Other appropriate constraints include maximum, minimum, and comparison operators.

Time-based filters can include a time range (e.g., the weather between June and March) or a sampling rate (e.g., the ending stock price every day). To use a time range, a person must specify a start time and end time. Sampling rates are specified with recurrence patterns of the form: `Month/Day/Year Hour:Minute`. For example, a lens with time filter `*/1/* 18:*` displays content from the first day of every month between 18:00 and 19:00 (i.e., during 6pm), as shown in Figure 7.5.

Filtering is visually depicted with a scented widget [169], which is displayed as a small embedded bar graph (Figure 7.4 and 7.8). The bar graph is displayed above the slider, indicating the location in time of the matching tuples. As a person moves the slider, the slider snaps to the bars, which act like slider ticks. Note that the bars need not be all of the same height and may reflect different information. A tall bar can indicate the appearance

of new content that matches a filter, and a short bar can indicate content that appears previously but still matches the filter. Consider Figure 7.8 in which a person has specified stories that mention "Ukraine." The tall bars indicate that a new story has emerged about the Ukraine at that time (e.g., "Yushchenko orders troops to Kiev"), whereas the short bar means that the "Ukraine" story still exists on the page but is not new.

7.4.5 Binding Lenses

People are often interested in multiple parts of a page or parts of multiple pages, as they may be comparing and contrasting different information. For example, in addition to traffic maps from 6pm each day, Ed might also want to look at traffic conditions on days when there are baseball games in the nearby stadium. Zoetrope flexibly allows for the simultaneous use of multiple lenses. Lenses can act independently or be bound together interactively into a synchronized *bind group*. Sliders within a group are linked together, causing them all to move and simultaneously update their corresponding lens.

People may bind lenses for different reasons. For example, to check traffic at 6pm on home game days, Ed can bind a lens for traffic maps at 6pm with a lens for home games from his favorite baseball site. Each lens in a bind group constrains its matching tuples to only include versions allowed by all other lenses in the group. Recall that this is achieved through a trigger filter. Each lens can add a new trigger filter parametrized to the time intervals that are valid according to other members of the bind group. Only tuples that satisfy all trigger filters are allowed. Thus, the resulting stream shows traffic data at 6pm only on days for which there are home baseball games.

Lenses can also be bound disjunctively. For example, one may want to find when book A's price is less than $25 or when book B's price is less than $30 (i.e., one of the two books has dropped in price). Zoetrope supports this type of bind, which is currently obtained by holding the shift key while performing the bind operation. However, this operation creates an interesting twist as it causes data to be *un*-filtered. When binding two lenses in this way, filter operators can be thought of as operating in parallel rather than serially. A tuple passes if it matches *any* filter.

7.4.6 Stacking Lenses

In addition to binding, Zoetrope also supports the stacking of lenses. For example, consider a person who creates one lens on a weather page, filtering for "clear" weather, and would like to further apply a filter that restricts the selection to between 6 and 7pm daily. Explicitly drawing one lens over the other and then binding them is visually unappealing and does not take advantage of the underlying semantics of the language. Instead, Zoetrope uses the notion of lens *stacking*. The toolbar in the Zoetrope window, which allows people to select the type of the lens, can also be used in a specialized binding operation which is called stacking. By dragging a lens selection from this toolbar to the bind button of the lens, a person indicates that they would like to further filter the existing lens. The original lens is replaced, and a new combined lens is generated, which takes the transform and filter from the original selection and augments it with additional transforms and filters. This new lens satisfies both the selection and constraints of the original lens as well as the new one. Furthermore, because some filters and transforms are commutative, stacking provides the opportunity to reorder the internal operations to optimize the processing of tuples.

Finally, consider the partial stacking of lenses where a person wants to make a sub-selection from an existing lens. For example, a person may apply a textual lens that tracks a specific team in the ranking. The textual lens will track the team no matter where they are in the ranking, but the person would further like to pull out the wins for that team at various time points. Thus, they may create a second structural lens that consumes the selection of the textual lens and selects the wins. While most lenses can be easily stacked without modification, lenses that stack on top of textual lenses require a slight modification to utilize relative information (paths or locations). This subtle modification is necessary because the textual lens selects information that is not in a fixed location in either the x, y space or the DOM tree. Because the textual selection is variable, the structural lens must utilize a path relative to the selection rather than an absolute path.

7.4.7 Design Considerations

The data Zoetrope manipulates irregularly changes, shifts, and vanishes, and thus the design had to address and accommodate this unpredictable behavior.

DOM Changes and Selection Failures

Since Zoetrope lenses track information over time, Zoetrope must be able to robustly extract the desired content from different versions of a Web page. However, it is not just the information within pages that changes; occasionally the structure of a page also changes. Indeed, we believe that any page displaying interesting, dynamic content will eventually undergo a template change which is sufficiently large that most extraction schemes will fail [54]. Others have responded to this challenge by constructing increasingly sophisticated extraction and tracking systems (e.g. [31, 111, 133]). These methods might be readily integrated into Zoetrope, however, by design Zoetrope can partially sidestep this problem. Zoetrope's real-time feedback allows a person to *immediately* see whether a desired selection effect was obtained. This is only possible because Zoetrope works over *historical* data and need not be robust in the face of unknown future changes.

A unique feature of Zoetrope is that targeted data that is changing quickly and thus requires crawling at a fairly rapid rate relative to other systems (currently once per hour). It is therefore worth briefly examining the amount of structural change that occurs in this type of data. Using the test collection, the stability of DOM paths was tracked. The first crawl of each page was examined and all of the leaf DOM nodes were collected. Additionally, the study included a re-crawl from an hour later to remove formatting nodes (e.g., "bold" or "italic") that quickly vanish and are not part of the page's template. The remaining DOM nodes were tested against versions of the Web pages a full day, a full week, five weeks, and a nearly a year later. The results show that the median survival rate of DOM elements within a page is 98% after one day (87% mean), 95% after one week (83%), 63% after five weeks (61%), and 11% after one year (23%). The differences between the median and mean survival rates indicate outliers in both directions (pages where nearly all elements disappear within a day and pages where nearly all elements remain after a year). Between

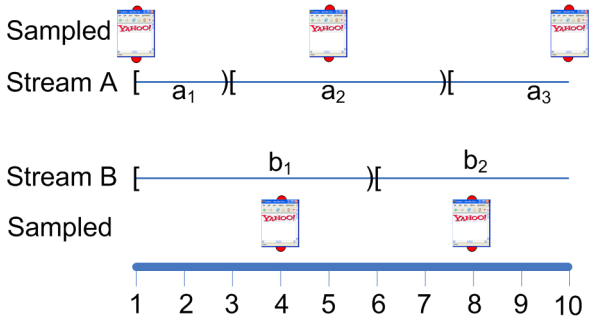


Figure 7.9: This illustration displays Zoetrope's interpretation of sampled content streams.

the initial crawl and the crawl at the end of five weeks, we see that vanished DOM elements, on average, contained 53 characters of text (38% of vanished characters were within links, each link averaging 17 characters). DOM elements that survived, on the other hand, had an average of 131 characters, suggesting greater stability in larger DOM elements. The tremendous amount of path death after a year is unsurprising. It confirms that websites undergo significant template changes that cause extractor failures. Anecdotally, however, we find, as did [54], that although some pages (e.g., *www.bankrate.com* or *money.cnn.com*) undergo major HTML structural changes, the *visual placement* of key features (e.g., a box of mortgage interest rates) is constant even after a year. While Zoetrope's interaction style allows people to quickly validate their extractions, one might wish to improve the extractors by augmenting DOM paths with visual and textual features.

Extraction failures may occur even when a page is sampled frequently over a short interval. The best response depends on whether such a failure is believed to be permanent or intermittent. Intermittent failures may be caused by randomization of page data (e.g., in a temporary promotion, Amazon interjects a counter of Harry Potter book sales on the front page). The simplest solution for addressing such transient effects is to introduce a dummy (blank) image when an element does not exist. In these experiments, however, it was discovered that introducing blank images for failures results in a disconcerting visual oscillation—the motion appears much like a low frame rate animation. Zoetrope instead displays a valid extraction from an immediately previous or future version. More formally, suppose that a lens slider is at time j and the nearest sample does not exist. Let i and k

denote the times of the closest valid samples such that $i < j < k$. In this case, the cropped image from the closer of i or k were displayed. If time i does not exist (i.e., there are no past instances where the element exists) or k does not exist (no future instances), the first or last valid instance is used respectively. Although artificial, this technique creates the illusion that the selection is continually present over all times. To avoid "deception," a cue (such as a red frame on the lens) can be displayed.

Windows in Time

It is worth briefly considering the design decision to display past content in the context of a Web page. In the design, the historical information overwrites the part of the page where the lens is located. An alternative would be to respond to slider motion by loading the entire page for a specific time and offsetting that page so that the selected area is underneath the lens (similar to automatically panning the lens "camera" at each step). Although this mode can be enabled, there are two main reasons why this type of movement was unappealing.

First, the motion and replacement of the entire page (rather than a small area) was highly distracting. Rather than visually tracking changes in a small area of interest, a person must track changes across the entire screen. Second, it is not clear what should happen when a person creates multiple lenses on the same page. As described above, multiple lenses represent different selections within a page. These selections may not always have the same visual relationship (for example, being 10 pixels apart in one time and 500 pixels apart in another) and may be set to different times. Given these constraints, it may not be possible, without distortion, to offset the underlying page to a position that works for all time periods or combinations of lenses.

Notions of Time in Zoetrope

When binding lenses on different pages, it becomes important to consider the sampling rate of each Web page, as dynamic pages that change rapidly may be sampled more frequently than those that are more static. To present a continuous and easily manipulated abstraction that hides the details of sampling, several assumptions are made: (1) that Zoetrope's sample

Figure 7.10: This timeline visualization shows the duration and frequency of news articles on the *cbc.ca* website.

rate is sufficiently frequent to capture every relevant state of the page, (2) and that, when content changes, it does so at the time point which is midway between the times of the two differing samples.

Figure 7.9 summarizes this interpretation of content streams. Suppose page A is sampled at times 1, 5, and 10; B is sampled twice, at 4 and 8. This is interpreted as if stream A has content a_1 during the half-open interval $[1, 3)$, has content a_2 during $[3, 7\frac{1}{2})$, and content a_3 during $[7\frac{1}{2}, now)$. By making these assumptions, Zoetrope supports the illusion of a fully-defined state over time as a person manipulates a lens over a temporal range.

The details of this interpretation become especially important when a person binds two lenses, making them show the same time (as in Ed's investigation of the effect of baseball games on traffic). Specifically, the interpretation defines a global state for the partially-sampled system, which allows Zoetrope to display corresponding states in multiple lenses simultaneously. In the example of Figure 7.9, there are four global states, defined by the cross product of the two streams:

$$[1,3) = a_1 b_1 \quad [3,6) = a_2 b_1 \quad [6,7\tfrac{1}{2}) = a_2 b_2 \quad [7\tfrac{1}{2}, now) = a_3 b_2$$

Formally, the evolving global state of the system may be viewed as a database relation where time is a foreign key.

7.5 Aggregate Visualizations

Lenses enable viewing of Web content from different moments in time, but this exploration may be just the first part of satisfying an information need. For example, a book's cost at

specific points in time is interesting, but a person may also want to graph the price over time, calculate averages, or test variability. To facilitate this type of analysis, a number of renderers that visualize or otherwise represent selected data were created. Visualizations, like lenses, create a sequence of transforms, filters, and renderers to display results. Although visualizations can exist independently of a lens, a lens typically defines the data displayed in the visualization. Lenses can thus also be used as a prototyping tool for testing selections and aggregations.

The transforms, filters, and processed streams generated by the lens can be directed to visualization rendering components that implement the visualization itself. For example, in a typical workflow one might place a (potentially filtered) lens on a book price, move the slider to test the selection, and click on the visualization button to graph the price over time. Internally, the visualization step reuses the transform module of the lens and connects it to a time series renderer (see Figure 7.12). As described below, other renderers provide additional visualization alternatives.

7.5.1 Timelines and Movies

The simplest Zoetrope visualization type is the *timeline* (see Figure 7.10), which displays extracted images and data linearly on a temporal axis. This visualization allows, for example, viewing weather patterns over the course of a year, headline images in stories that mention Iraq, or unique articles about a favorite sports team (all ordered by time). As before, the rendered images visualized in the timeline are live and a person can click on any of the links. Double-clicking on any image in the visualization synchronizes the page (or pages) to the same time, allowing a person to see other information that appeared on the page at a particular time. This visualization can also eliminate duplicates and display a line next to each image depicting its duration. This type of display shows when new content appears and how long it stays on a page (e.g., a story in the news, a price at a store) To prevent the timeline from running indefinitely to the right, the visualization can fold the line into a grid with each row denoting activity over a day (or other interval).

The timeline visualization gives an instant sense of everything that has happened over

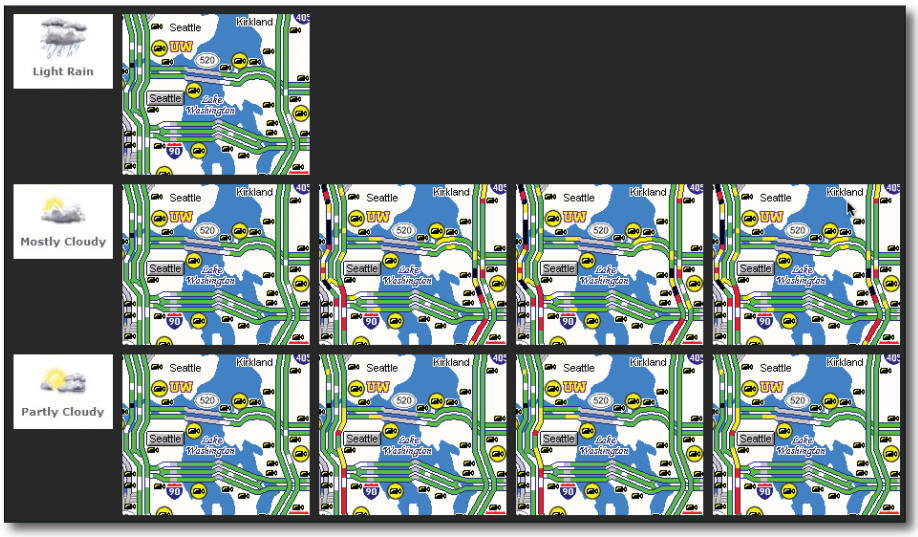


Figure 7.11: This cluster visualization shows traffic data according to weather conditions.

some period. However, other examples are best served by cycling through the cropped images to produce an animated movie. Although this is equivalent to simply pulling the slider through time, a *movie visualization* automates and regulates transitions and looping while the visualization cycles through the images. For example, a static USGS earthquake map can be transformed into an animation of earthquakes over time, helping to pinpoint significant events.

7.5.2 Clustering

The timeline visualization is a simple example of a grouping visualization, where extractions are grouped by some variable (time in this case). However, other more complex groupings are also possible. *Clustering visualizations* group the extracted clips using an external variable derived from another stream. For example, a clustering visualization can merge data from two different lenses, using one lens to specify the grouping criteria while the other lens provides the data. Figure 7.11, for example, shows a cluster visualization of traffic and weather data. The visualization creates a row for every weather condition (e.g., sunny, clear, rain), and every instance from the traffic selection is placed in the appropriate row

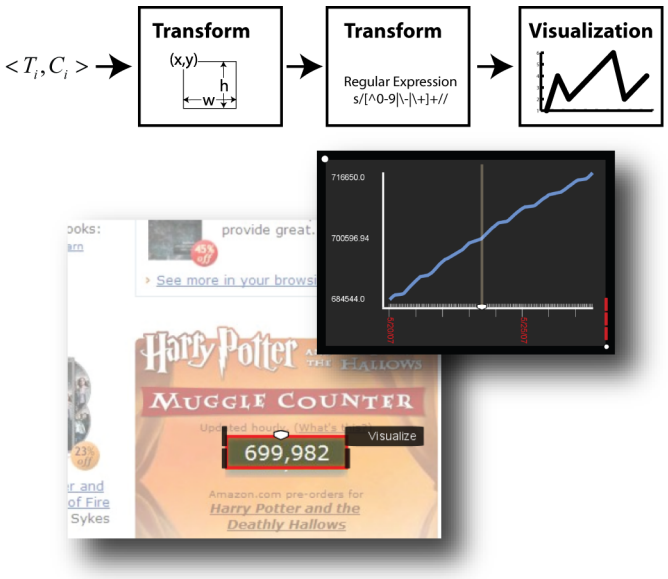


Figure 7.12: A time series visualization displays the Harry Potter book pre-sales, or Muggle Counter, over time.

depending on the weather condition at the time of the clipping. If it was rainy at 8:15pm, for example, the traffic map from 8:15pm is assigned to the rainy group.

7.5.3 Time Series

A variety of interesting temporal data is numerical in nature, such as prices, temperatures, sports statistics, and polling numbers. Much of this data is tracked over time; however, in many situations it is difficult or impossible to find one table or chart that includes all values of interest. Zoetrope automatically extracts numerical values from selections of numerical data and visualizes them as a time series. Figure 7.12 shows a visualization of the number of Harry Potter book sales over time. The slider on the x-axis of the time series visualization is synchronized with the slider in the original lens selection. When the person moves the lens slider, a line moves on the time series visualization (and vice versa).

Time-series visualizations take the output of a lens and transform it to extract numerical values (i.e., $\langle T_i, C_i.Text \rangle \rightarrow \langle T_i, Number_i = \textsc{findnumber}(C_i.Text) \rangle$). At present, this is a simple regular expression, but a more complex transformation can be included to handle

selections with multiple numbers or mathematical transforms.

7.5.4 Exporting Temporal Data

Zoetrope offers many useful visualizations, but was also designed for extensibility. Data extracted using Zoetrope lenses is therefore also usable outside of Zoetrope. Systems such as Swivel (*www.swivel.com*) or Many Eyes [161] excel in analysis and social interactions around data, but are not focused on helping people find the data in the first place. Zoetrope is able to generate a temporal, data-centric view of different websites to meet this need. To export data outside of Zoetrope, a Google Spreadsheet "visualization" is created that sends the lens-selected values to the external Google Spreadsheet system. When a person selects this option, appropriate date and time columns are generated along with the content present at that time interval (either strings or numerical data). This scheme greatly expands the capabilities of Zoetrope by allowing people to leverage external visualizations, integrate with Web mashups, or perform more complex analyses.

7.6 Implementation

Zoetrope is a Java-based program implemented using the Piccolo library [23]. Zoetrope primarily relies upon Piccolo for zooming features, but generates special image functions to manipulate, slice, and display Web pages. Zoetrope's implementation is designed to balance the need for interactivity with scale. Pre-rendering and caching certain data structures and images in memory allows Zoetrope to respond in near-real time. Using low-resolution images, the system provides immediate interactivity during slider motion. The system also takes advantage of idle time to background load high-resolution cropped images. Visualizations based upon images also use low-resolution substitutes that are gradually replaced as higher-quality versions become available.

7.6.1 Crawling

The underlying Zoetrope data is generated by crawling pages at semi-regular intervals. Although at present re-crawls are done hourly, this interval could be strategically determined

by observing the amount of content change. Initially, the crawler was constructed using a modified *wget* process (the unix command) that retrieved both the page of interest as well as all content attached to a page (JavaScript, CSS, images, flash files, etc.). This, in itself, is generally sufficient to create an approximation of the page as it appeared at a given time. However, dynamically re-rendering pages creates unattractive delays that are inconsistent with Zoetrope's need to support highly-interactive rapid shifts among different times. This led to the previously mentioned optimization of storing pre-rendered images of pages. Image generation was initially achieved with a program containing an integrated Internet Explorer browser. The program, run separately from the crawler, would read the crawled data and produce a rendered image and a modified HTML file that included the rendered location of each element in the DOM. This transformation is consistent with the notion of temporal tuple processing, as the system iterates over page instances and produces a second stream of images. The difficulty with this approach was that content was frequently missed in the crawl (what *wget* captures is different from what the browser expects), and dynamic content produced a mismatch between what was retrieved and what was rendered (e.g., a JavaScript program that inserts the current time into the page, at rendering time, regardless of when the page was crawled).

We solved this problem with a new system that combines crawling and rendering into a synchronized process. Two Firefox plugins were extensively modified and integrated. The first, Screengrab! (*www.screengrab.org*), produces a capture of whatever is being displayed in Firefox. The second, WebPageDump [135], waits until the page DOM stabilizes in memory and subsequently outputs a single JavaScript-free HTML page, images, and a single CSS document describing all necessary layout constraints. As before, the HTML page is modified to contain the coordinates of all rendered DOM elements. Based upon both of these plugins, the current Firefox-based crawler captures both an accurate representation of the page as it looked at the time of retrieval and sufficient content so that the page can be correctly re-rendered if necessary. Using a Firefox plugin has the further advantage that people can add to their own database as they browse the Web.

The experimental dataset includes 250 pages and over 1000 samples per page. Each page requires, on average, storing 92Kb (68Kb median) per crawl beyond the original page,

which requires an average of 320Kb. This includes all content necessary to render any crawled version of the page (e.g., HTML, all images, Flash files). The first page crawl is much larger, because in subsequent crawls linked files, such as CSS and images files, are checked for changes and only downloaded when they have changed. Currently, an MD5 checksum is used to check for changes. Subsequent crawls are thus only about 28% of the size of the original crawl. Most of the new data, 92%, is due to HTML changes rather than changes in linked files. We believe that the dataset size is artificially high for a number of reasons: (1) the pages in this dataset were selected because they are known to change frequently, (2) currently content that is duplicated between pages from the same website is not thrown out, and (3) no compression is used on any files. Eliminating these inefficiencies will likely lead to significantly improved crawls and reduced storage space requirements. For example, compressing the pages using *rzip*, which takes advantage of repetitions across files, the average HTML page crawled 5 weeks after the original visit requires only an additional 2Kb of storage (272 bytes median) over the compressed version of the original page (15% of the size of the original compressed page).

7.6.2 Backend

The DOM for each version of a page are loaded into an in-memory XML database. Saxon (*www.saxonica.com*), provides XPath query functionality and allows us to rapidly find matching elements or entire versions of pages. An in-memory index also tracks DOM elements by their rendered x and y coordinates. This structure allows us to quickly find which elements are clicked or selected.

7.7 Summary

This chapter presents Zoetrope, a system for interacting with the ephemeral Web. Zoetrope allows people to interactively access and manipulate a historical record of the evolving Web. Familiar pages, which have previously existed only in the present, now have the added dimension of time. To help manage and utilize this multi-version Web, Zoetrope implements a number of interactions based in lenses and visualizations that allow people to

access and analyze historical information through contextualized selections and extractions. Furthermore, developers can use Zoetrope to rapidly prototype and test temporal queries through Zoetrope's visual query interface. With the Zoetrope system, we introduce a novel semantics for dealing with temporal Web data and show a small number of operators that can be utilized for many different tasks. The present implementation of Zoetrope has only scratched the surface of potential lenses and visualizations. By creating an underlying language for selecting and processing temporal page data, we hope to inspire and enable others to explore new interactions and applications in this domain.

In addition to continuing to build new visualizations and extractors, we are extending the semantics of the underlying language to support more sophisticated queries and binding operations for specifying temporal logic operations and functions (e.g., the gas price for two days *after* oil prices are $60 per barrel). Also, the highly regular sampling of the dataset allowed for a number of assumptions in the semantics, and in the future further exploration of it would be desirable to include support for irregular crawl rates and the interpolation of data as necessary.

Chapter 8

Related Work

Each previous chapter contains a short description of related work and the specific contribution of the novel studies and system described relative to that work. Below is an expanded survey of the current state of the art in Dynamic Web research. The breadth of the contributions in the previous chapters is also reflected in the variety of related work described here.

8.1 Web Evolution

The amount of change on the Web, both in content and page structure is of critical importance to our idealized system. Characterizing the amount of change on the Web has been of considerable interest to researchers and corporate interests alike ([20, 29, 32, 34, 48, 53, 54, 58, 59, 68, 98, 102, 126, 128, 134], see Table 8.1). These studies have variously focused on growth, decay, and change primarily from the perspective of systems design (e.g. how often are crawls necessary? how fresh is an index? how well do caching schemes work?). In part due to their broader focus, Web evolution research has concentrated on random cross sections of the web in their study of change. Additionally, because of the complexity of the calculations, most have avoided metrics of actual content change from version to version, though there are some notable exceptions.

Perhaps the largest scale study of Web page change was conducted by Fetterly et al. [59]

which tracked 150 million pages once a week for 11 weeks, and compared the change across pages. Changes were measured through the overlap of "shingles" [35]. They found a relatively small amount of change, with 65% of all page pairs remaining exactly the same. The study additionally found that past change was a good predictor of future change, that page length was correlated with change, and that the top-level domain of a page was correlated with change (e.g., edu pages changed more slowly than com pages). Ntoulas, Cho, and Olston [126] similarly identified minor changes according to bag-of-word similarity on weekly snapshots of 150 websites collected over a year. Even for pages that did change, the changes were minor. The longest longitudinal study is the 360 page sample studied between 1996 and 2001 by Koehler [102], finding that page change plateaus as a page ages. Specifically, Koehler observed that the collection of pages defined in 1996 stabilized as the collection aged.

While general studies of content change in web documents is rare, studies of structural change are rarer still. One early exception was the work by Gibson et al. [68], which identified an overall increase in the complexity and number of structural templates in web pages. More recent work by Dontcheva et al. [55] tracked a small collection of pages and determined the tree edit distance between subsequent versions. The work identified both minor and major structural changes (changes close to the root of the DOM tree), with 74% of tracked pages displaying at least on major structural change. Their work also identified a likely relationship between the number of changes and the popularity of the site.

For the purpose of understanding how users interact directly with pages–rather than through a search engine or cache–there are a number of missing elements that need to be resolved. First, evolution studies are primarily directed at understanding change in a random cross section of the web. As we are interested in pages that are actively monitored or accessed repeatedly for information, a random subset is not appropriate. Second, with the exception of instrumented site studies (e.g., [134]), the fastest sampling rate of any previous evolution work has been at the rate of a day. While there are many interesting changes that can be captured at those resolutions, a great deal of change is unnoticed. Much human driven data (e.g., stock prices, bidding information, political news events, etc.) and other tracked phenomena (e.g., weather, earthquakes, celestial events) is published to the

web significantly faster than once a day. Thus, it is necessary to collect this data at a faster rate, and develop new models to describe finer grained evolution. Finally, little attention has been paid to the structural changes to web pages. In part this is due to the complexity of comparing structural data. In order to understand and track such changes it is important to create more scalable algorithms to analyze many samples from the same page (of the type described in Appendix A).

8.2 Revisitation

Early research aimed at understanding general Web navigation and surfing behavior first reported a large amount of re-access of information (e.g., [38]). To better understand and quantify this phenomenon, subsequent studies were designed to specifically address revisitation behavior. These studies come primarily in two main flavors (though some, like our own, mix elements):

- Log studies, where browsing patterns are monitored either through proxies or instrumented browsers ([51, 80, 95, 127, 150]).

- Survey/interview studies, in which a questionnaire or interview is constructed to understand specific behaviors ([16, 94, 144]).

Studies of revisitation have demonstrated that 50% [80, 150] to 80% [51] of all Web surfing behavior involves pages that users have previously visited. While many revisitations occur shortly after a page's first visit (e.g., during the same session using the back button), a significant number occur after a considerable amount of time has elapsed [127].

Two significant outcomes of earlier efforts have been the creation of taxonomies of browsing behavior in relationship to revisitation [95] and suggestions for improving the design of Web browsers [16, 127], search engines [16, 152, 168], and personal information management systems [91]. Taxonomies in this space have sought to define revisitation in terms of user intent, goals, or strategies [122]. New browser designs have concentrated on better back buttons [75, 120], history or bookmarks [17, 92, 147], and monitoring and notification

features [94]. Furthermore, several products and browser plug-ins have emerged to assist users in revisitation and monitoring activities (e.g., [153] and more in Section 8.4 below).

As previously alluded to, past work on revisitation was limited due to the small user populations. The likelihood of two or more users revisiting all but the most popular sites made page level analysis all but impossible.

Douglis et al. [58] studied how the number of visitors to a page relates to change, finding that frequency of change was higher with increasing access. Pitkow and Pirolli [134] looked at a different facet of Web evolution, in particular they found that desirable pages were less likely to disappear (i.e., were more likely to survive). The notion of monitoring pages for changes in content is discussed in some revisitation literature ([94, 127]). Obendorf et al. [127] recorded a hash of each page downloaded in their study of revisitation, and observed that revisited content frequently changed. They additionally found changes often interfered with long-term re-finding. Similarly, Teevan et al. [152] found changes in re-finding behavior as a result of changes to a search engine's result pages.

8.3 Temporal Prediction

A critical aspect of analyzing time-series data is the ability to detect trends. We benefit here by the long-running interest in the data-mining community on trend detection in web and textual data (e.g. [67, 100, 163] and extensively reviewed in [101]). Such systems can extract a number of interesting events that can then be analyzed and compared in multiple datasets using our system.

Another requirement of our system is the ability to generate topics from query logs and text collections. As we are primarily motivated by query behavior in this work we have made use of algorithms similar to [167]. However, there is a large literature on mining topics from document collections (e.g., [10, 108]). These may be useful to our work in the future as we begin to utilize textual data source to automatically generate topics of interest and compare these using our techniques.

In predicting the effects of one data source on another there are a number of examples targeted at specific application areas including: the prediction of purchasing behavior [76],

the effect of published text on stock prices [108], as well as interest levels in web pages based on the (dis)appearance of links [8]. These systems are most closely aligned with our goals, as they frequently utilize automatic alignment of documents to events.

Because we wish to support targeted data analysis (rather than simply automated mining) we are interested in providing users with visual representations of the data that can be quickly and easily interpreted at a glance. Though there are a number of systems for the visualization of time-series data (e.g., [79, 110, 160, 166]), there are far fewer that support visual summaries ([110, 160, 166]), and none that we are aware of that provide a visual summary of the differences between two datasets. This visualization (the DTWRadar) may be of independent interest to other domains where time-series are compared.

8.4 Change Monitoring and Passive Clipping

In response to the need for monitoring temporal data, a number of solutions have emerged in the research community (see Table 8.2). Arguably one of the first, the AT&T Difference Engine [57] archived pages and displayed differences between consecutive versions of pages. Others have also described solutions to the monitoring problem, although the focus between them differs. Some researchers have described large scale, server-based monitoring solutions for groups [32, 57, 114, 113, 130] and some concentrate on the UI aspects of conveying and displaying change [1, 45, 73, 88, 142, 146]. This is in line with research in the database and data mining communities and adapts XML differencing to HTML/Web differences (extensively surveyed in [71]). Although these techniques are useful for the comparing two versions, Zoetrope is designed to deal simultaneously with many instances of a given page.

In general we will refer to monitoring applications that provide clipping services as passive clipping. The user selects portions of the page (i.e., clips) and creates a new synthetic page (i.e., "personalized") containing all the clips. The system either automatically or manually updates the clips to match the current data on the web. In such applications, clips are treated independently with the only relationship being that in the spatial placement of the clips.

A second major area of research has been the visual display of temporal variation in

information. Certain solutions are targeted at news while others act more generally to represent differences between versions of webpages. These include techniques for converting textual information into time series (e.g., [62]), and other difference visualizations for entire websites (e.g., [112, 114]). The majority of these systems are designed to address the visualization needs of the user after the data has been extracted and generally do not concern themselves with the difficulty of the extraction itself.

8.5 Active Clipping and Mashup Applications

Perhaps the research area closest to Zoetrope comes from the end-user programming community. Topics in this literature include active clipping and mashup applications ([14, 21, 22, 27, 40, 55, 56, 66, 78, 82, 83, 84, 85, 111, 116, 121, 146, 156, 171], see Table 8.3). We differentiate between active clipping applications and the passive clipping previously described. Though both types allow users to select regions of interest, active clipping builds more complex applications and allowing the user to define some relationship between clips. In some situations this may mean that clips are templates and extraction rules to be used to represent additional data. For example, Dontcheva et al. ([56, 55]) allow users to select restaurant a restaurant's address from one page, reviews from another, and bus schedules to the area from a third. Clips are connected (e.g., restaurant name in the address clip to the name in the review page, and the address in to the bus schedule). Subsequent queries for restaurants retrieves data from all three sites, linking them together as in the first example. Other efforts in the area have allowed for "generalizations" of queries [27], potentially to the deep web, or extractions [40, 85, 155, 156], to both similar "objects" (e.g., products or restaurants) on the same page or other pages on the site. These systems are targeted at providing users with a different view of the data present on pages either through faceted searching/browsing or visualization.

A different type of active clipping allows users to extract the computational components of different pages (e.g.. calculate the exchange rate from dollars to yens or look up the stock price for some company) into one application (e.g., enter the stock name and automatically convert the price to yen) [22, 66, 86, 148, 149].

Similarly to active clipping, mashups allow a user to define a relationship between web components. However, unlike clipping applications which scrape the content from the page, mashups utilize programmatic APIs. The recent trend in the Web (i.e., "Web 2.0") of producing programmatic APIs has opened access to many widgets, datasets, and systems. Some are visual functions (e.g., a Google maps widget or Exhibit timeline [83] but many are simply functions or datasets over which a program can operate. Mashup construction has seen interest in both academic [78, 171] and commercial interests (e.g., Intel's Mashup Maker, Yahoo's Pipes, Microsoft's Popfly, and Metafy's Anhracite).

Mashups and active clipping frequently rely on ideas from the Programming by Demonstration (PBD) literature to let users more rapidly express a workflow or set of actions to invoke on web pages [111, 171]. Regardless of the mechanism, previous applications have focused fairly exclusively on the "present web," potentially on many pages at the same time. Multi-page extraction schemes attempt to extract information from many templatized pages. This is essentially the transpose of the problem we wish to address. For our solution we would like to extract information from many templatized versions of the same page. This subtle difference is in fact crucial for two reasons. First, the data on the page primarily differs over time rather than in some other dimensions–we extract the weather over time for the same city, the stock price for the same stock, and so on rather than the stock price for many stocks, the weather for many cities. Second, by concentrating on one page, the extraction can be visualized in the context of the page, something which may not make sense for many different pages.

8.6 Extraction, Web Scraping and Wrappers

In addition to end-user programming, a significant amount of early web research was focused on automating screen/web scraping (e.g., wrapper generation). Most solutions concentrate on the creation of extraction languages and automated algorithms to learn extraction patterns through examples (surveyed extensively in [42, 107]). Information extraction systems range from completely manual with hand coded rules (e.g., [44]), supervised learning where pages are labeled (e.g., [104, 105]), semi-supervised learning where rough templates are pro-

vided (e.g., [12, 15]) and un-supervised learning (e.g., [165]). These systems can be further divided into the target of the extraction (single record or multiple records from a page, and single and multiple pages). Extraction techniques may include language, structural and even visual (e.g., [41]) features.

Active and passive clipping applications may rely on underlying extraction systems (though notably, many make use of simple heuristics rather than full blown learning algorithms). Similarly, an idealized version of Zoetrope may make use of extraction techniques, and in particular those supervised techniques that work with examples. By generalizing the extractions created on one page, a robust extraction system may be able to "save" an extraction and use it on a different page (i.e., apply the temporal extraction pattern for the historical stock price of GOOG to the historical stock price of MSFT). Furthermore, extractions systems that are able to determine instances of failed extraction may be used to help the user identify historical periods where their extractions have failed and where they should create a new extraction.

Extraction schemes that identify and index tables on the Web (e.g., [37]) are also interesting for our application for two reasons: a) historical temporal information is frequently held in tables, and b) users may select information that is held in tables for tracking over time. The ability to automatically assign column headings, identify rows, and so on is therefore of great interest. Because table extraction and indexing can work independently of any manual extraction, we may also be able to use this data to complete gaps (we discuss this possibility below).

The present proposal is not aimed at developing new forms of Web information extraction. Our idealized system may simply make use of existing solutions. That said, the nature of the temporal extraction task does have some interesting features that may be taken advantage of regardless of the extraction system that is used. For example, the particular interface design of Zoetrope, in which users can select and test their extraction, may be conducive to better extractions. In particular, because a user defines a range of documents over which their extraction has succeeded, many versions of the page can be used as training examples for the extraction system. Additionally, because our system works on the same page over time, a period in which the template of the page changes may not be as disastrous.

Assume, for example that a page has template A and data D at some time t. At time $t+1$, the template is now A', causing the extraction system to fail. However, because the data has not changed in that instant (e.g., the book price or the restaurant review) have not changed (still D), it may be possible to map the structure of the two templates by using the fixed data. The extraction may then continue, unhindered by the change in template.

8.7 System Solutions, Databases and Information Retrieval

The notion of versioning semi-structured documents (principally XML) is of great interest [47, 119]. As with any versioning system, research has concentrated on efficient deltas, snapshots management, and compression but targeted at a specific data domain. In addition to XML structures, the hypertext community has also recognized the need to archive information in larger information spaces [162]. The results of both chains of research, one at the document level, and one at the larger collection levels are both interesting to us as they may be utilized to optimize the underlying data store for Zoetrope. As we later show, because we store many versions of the same page, the compressibility of this information is fairly high. Thus, we are more interested in efficient queries (e.g., [47, 125]) on these collections as many of our algorithms require multiple lookups in the database.

In the past few years interest has developed in the indexing of temporally annotated full text documents [24, 81, 89, 141, 159, 172] (though some non-web literature exists prior to this [13]). This research concentrates on two aspects of the problem: a) (space) efficient indexing, and b) search. In a way, the former is a problem that is easier to evaluate than the latter and has received much more attention (e.g., minimizing the amount of IDF data that needs to be stored, number of times PageRank needs to be calculated, etc.). These systems satisfy one particular information need for our idealized system as we would like to be able to locate documents in the past containing certain terms and keywords. However, because we are interested in the presence of absence of text in a sub-part of a document, further work is necessary to adapt temporal IR solutions to our domain. Below we discuss mechanisms by which blocks of related text may be identified. These blocks may represent document subunits that can be indexed independently.

8.8 Natural Language Processing (NLP) and Topic Detection and Tracking (TDT)

Finally, we briefly consider efforts in the NLP/TDT domains in relation to temporal issues. The NLP literature, in particular, has focused fairly extensively on attempts to label discourse with temporal annotations. The extraction of time stamps (e.g., [60]) and more broadly the induction/inference of temporal order (e.g., [33, 77, 118, 117]) have generated a number of interesting results. Notably, in Zoetrope we annotate data with the time of its observation on a web page. The data may in reality be in the past relative to this time or in the future. However, as we discuss below, we label that data as valid for the interval that is half way to the previous observation and half way to the subsequent. Although, this can be refined slightly using learning rates of change and other features, without semantic understanding of the document we are left with an ordering of data based on the time at which that data was observed on the page. This is sufficient in situations when the data is understood to represent the "current" state (e.g., the current stock price, the current weather, etc.) or has a constant offset (e.g., 10 day forecast). NLP techniques may be attractive additions to Zoetrope in situations where temporal relations and unclear or the time label of the document is different than the "real" time associated with the data. However, this is a somewhat orthogonal problem to the one we are pursuing with Zoetrope.

A more interesting area that has appeal to Zoetrope is in Topic Detection and Tracking (TDT). When dealing with numerical temporal data, the notion of an "event" can be defined in terms of some property of the time series (e.g., thresholding, anomalous values, etc.) . With text, this is not quite as easy. Though there are mechanisms for transforming text into numerical data for certain kinds of event and trend detection [100], summarizing and tracking textual information is a somewhat different problem. Fortunately, the TDT literature (covered extensively in [9]) offers a number of potential solutions for both identifying events, identifying useful additional information (i.e., an additional piece of information about an event that has sufficient value over what we already saw before) [62, 67], and summarizing or aggregating events [11, 87].

Table 8.1: Summary of Web evolution studies (continued on next page)

Study	Size/Methodology	Key Findings
Bar-Yossef et al. [19]	1000 randomly chosen pages from a 2B crawl 2004, limited selection of WWW conference pages, Yahoo nodes, etc. since 1998 from archives	Calculation of page decay indicated a high rate of page "death" (47.5%) with 71% pointing at dead links or soft-404 pages (pages replaced with missing page info but not explicitly a 404 message). Use of PageRank-like metric to label pages as decayed utilizing the number of their decayed neighbors.
Boese and Howe [29]	6 corpora from 1996-2004, ranging in size from 70 to 8282 with 1-100+ identified genre labels	High similarity (by cosine metric) for surviving pages (survivability was low between 1999 and '05 at 42%). Only 2-3% change in accuracy when training on 1999 data to classify 2005 data.
Bogen et al. [30]	62 popular blogs, daily crawl from 9/2006 to 12/2006	Witnessed stabilization in the absolute change over time similar to [6] but with less granularity
Brewington and Cybenko [34]	100k pages per day (sampled from clipping service), 7 month sampling in 1999	Determined a crawling model for capturing data on the changing pages, estimated 104Mbits/second link to capture on 800M pages. 56% of pages did not change, 4% changed every single time.
Cho and Garcia-Molina [48]	720k pages from 400 sites, 4 months, 3000 pages per-site per-day	20% change on every visit (40% in com domain), 70% of pages lasted for one month, 50 days for 50% of the web to change
Dasgupta et al. [53]	42 web sites sampled over 23 weeks from 11/2004 to 6/2005 (used [126]). 640k starting pages, 223k new pages emerging	All new content can be discovered with 9% overhead (number of fetches to discover one new page) given perfect knowledge. However, without this knowledge, only 80% can be discovered with 160% overhead.
Dontcheva et al. [55]	100 pages from 20 sites over 5 months, crawled daily	Measured the amount of structural change on the page. Identified a relationship between page popularity and number of changes. Change measured by tree-diff algorithm. Structural changes were minor for most pages, 74% of sites exhibited a major change (change close to the root).
Douglis et al. [58]	Changes measured to pages re-accessed through web proxy, 105k resources (pages, images, etc.) retrieved 2 or more times, 10 months	16.5% of resources changed every visit, for HTML 50% changed. Probability distributions calculated for various domains/resources. Concentration on the use of web caches and identification of replicated content.

Fetterly et al. [59]	150M, once a week for 11 weeks in 2002	85% or more successfully downloaded over 11 week period. 65.2% of pages do not change at all, 9.2% only change in markup
Gibson et al. [68]	72K pages from 1380 snapshots over 8 years (Internet Archives).	40-50% of page content is template material, template content has grown 6-8% per year.
Kim and Lee [98]	1.8 3M in two day intervals for 100 days in 2003, mixture of "famous" and random sites (34K total)	Found 73% of pages were never modified, less than 5% changed on each crawl. For famous pages, 57% had no modification cycles (42% of random).
Koehler [102]	361 random pages and 344 sites for 1996-2001	Half-life of pages and sites is approximately 2 years, domain variability. Find a possible stabilization in survival (i.e., documents that persist long enough are likely to continue to persist)
Ntoulas et al. [126]	Weekly for 150 sites over one year (10/2002 to 10/2003), used DMOZ to select sample	Estimate 8% growth of the web per week, 20% of pages will be accessible after a year. Estimate of 50% new content every year. 80% of the link structure changes over a year. 70% of changed pages show ¡= 5% difference from initial version after a week.
Olston and Pandey [128]	10K random pages, crawled once every 2 days for 50 snapshots, 10k DMOZ pages every 2 days for 30 snapshots	Find a weak correlation (.67) between change frequency and longevity of content. 90% of content fragments are static, 3% are extremely dynamic. Can refresh pages at a much lower rate (every 10 days instead of 5) and still preserve same "staleness" level (calculated as Jaccard over shingles)
Pitkow and Pirolli [134]	5.6k pages at GVU From 1/1996 for 215 days, access logs for 226 days	Comparison of change identified increased likelihood of death for pages accessed internally whereas external pages were more likely to survive. Items that receive more editing attention survived longer.

Table 8.2: Summary of change monitoring and passive clipping projects (continued on next page)

System	Description
Ackerman et al., The Do-I-Care Agent [1]	The Do-I-Care agent is able to monitor pages at user specified rates. The change is test for "interestingness" given a user's profile. Changes are ranked and then the user is notified of the change.
Boyapati et al., ChangeDetector [32]	ChangeDetector is primarily focused at monitoring business pages. The system automatically generates a likely site map for monitoring and utilizes entity extraction on pages at periodic intervals.
Chakravarthy et al., WebVigiL [39]	Monitoring is defined through an Event-Condition-Action (ECA) model. Pages are monitored at a specific rates (or "best effort"). Changes satisfying a condition activate an action (e.g., email user)
Chen et al., WebCiao [45, 57]	WebCiao is able to track changes on entire websites. The user specifies the website of interest, the website are converted into an entity-relationship database and two versions are compared with the differences output in a "pretty printed" format through HTMLDiff. Later emerged as the AT&T Difference Engine.
Fishkin et al., WebTracker [61]	Simple system using the Unix diff command to actually check for differences.
Flesca et al., CDWeb [63]	Attempts to monitor changes of specific portions of the document, recognizing that not just any change is interesting. Uses WebTrigger, a language to specify events and actions on those events.
Greenberg et al., WebTracker [73]	WebTracker allows a user to select clips from a page, a rate for how often to test for changes and a threshold for the amount of change. A VCR like interface allows the user to go back and forth on captured clips.
Jatowt et al., WebSCAN, CWB, and change visualization [90, 124, 137]	WebSCAN considers both changes to individual pages as well as sites. "Change worth" is calculated through a combination of a number of factors. A freshness score is calculated which takes into account the number of pages with similar content, the amount of change (in vector space), the density of pages, and the amount of time between changes. Additionally popularity, browsing frequency, and update frequency are also considered. Not completely implemented. The Comparative Web Browser (CWB) allows for two pages to be tracked simultaneously. Scrolling in one moves the second window to display the most similar content. A later system describes various annotations to web pages including a clock indicating when the DOM element changed and a personalized measure based on revisitation rate, last change, first visit, etc. (i.e., WebSCAN features)

Liu et al., WebCQ [114]	A large scale monitoring infrastructure. Different sentinels are generated to track different types of changes (links, regular expressions, etc.)
schraefel et al., Hunter Gatherer [142]	A simple clipping application to collect portions and components from web pages into a collection. Collections/clips reflect live content.
Sugiura et al., Internet Scrapbook [146]	An early clipping application supporting the extraction of web clips into one application. Clipped content can be "updated" to contain the most current information found on the web.
Wang, WebDNA [164]	A complete system for monitoring content. The system utilizes a tree differencing mechanism to determine changes to pages. Additionally, changes are ranked based on page popularity, a decaying function for time, and a "quality" measure (length of new information)

Table 8.3: Summary of active clipping and mashup applications (continued on next page)

System	Description
Anupam et al., WebVCR [14] and Freire et al., WebViews [64]	A macro recording system for the web. A number of heuristics are used to deal with broken elements (e.g., "Smart Bookmarks"). WebViews extends the WebVCR by allowing extractions from the page (a simple language to find an pull data from the page). A GUI is used to identify clips with more information producing a more robust clip.
Bauer et al., InfoBeans [21] and InfoBoxes [22]	InfoBeans are web clips defined by HyQL, "a SQL-like WWW query language" by tracking user actions. In an extension, information agents are designed to extract and connect InfoBeans so that data pulled as "output" from one bean can be propagated as input to another.
Bigham et al., Transcendence [27]	Transcendence provides access to the deep web by utilizing query expansion through KnowItAll extractions. This data is used to search the site for more matching documents/items which are then collected and visualized.
Chan et al., Vispedia [40]	The system provides a mechanism for visualizing RDF data held within Wikipedia. Users select tables, which formulates a query that is generalized to explore the RDF paths. Matching data is presented for visualization depending on the data type.
Dontcheva et al. [55, 56]	A collection of systems that are used to clip portions of a pages. Clips can be laid out in appealing ways as "cards" which can then be linked together such that the relationship between fields in one card are explicitly made to fields in another. A search interface allows a user to initiate a query which will compile the results using the original card template.
Fujima et al., Clip, Connect Clone (C3W) [66], IntelligentPad [86, 116, 149], RecipeSheet	C3W is a clipping application focused on utilizing the computational logic provided by multiple websites by integrating form and result elements into a composite web application. The composite application responds to user input and propagates calculations or queries and extracting results. The system utilizes a spreadsheet metaphor behind the scenes. The work culminates with RecipeSheet, a workflow style interface to design "information processors" using pieces of web pages (including executed content) as well as a number of visualizations
Hartmann et al., d.mix [78]	d.mix is a proxy based system that provides annotations on pages that indicate how information can be extracted from the page. The API is custom programmed for each site. Resulting programs can be shared through a wiki-like system.
Miller et al., Turquoise [121]	Turquoise is a combined browser/editor utilizing PBD to create "dynamic pages." Page elements and form interactions are collected through demonstration and generalized into a program.

Tuchinda et al., Karma [155, 156]	Spreadsheet/table style system. Users can drag one example into a table (e.g., a restaurant). The system will fill in other rows with more data. The system can go to individual details page for each row and add additional data.
Wong et al., Marmite [171]	A Unix pipes-like metaphor for creating mashups. Various operators can be linked together visually to manipulate web data and display that data in various modes (tables, maps, etc.).
Huynh et al., PiggyBank [82], Exhibit [83], Sifter [84], Potluck [85]	These four projects represent different ways of interacting with the semantic web (or producing semantic information when it is not there). PiggyBank, for example, will identify semantic content (or utilize screenscrapers) when available on a page. This data can then be used to enhance access to the content (e.g., faceted search). Similarly, Potluck highlights fields (e.g., "name" or "address") on pages, allowing users to link equivalent fields. Utilizing a few examples, the page can be labeled and data extracted for faceted browsing type applications. Exhibit, consists of a number of AJAX-driven widgets that are able to represent semantic web data in different ways (timelines, maps, etc.). Like PiggyBank, Sifter collects object data (e.g., books sold on Amazon), potentially from multiple pages. On identifying blocks, and pulling fielded data, Sifter allows users to resort or perform faceted browsing on the data within the context of the browser.

Chapter 9

Contributions and Future Work

This dissertation advances the understanding of the Dynamic Web as well as describing a set of novels tools for manipulating temporal data. Specifically, this work aims to answer the following questions:

1. **Resonance** What is the relationship between changing Web content and the ways this data is consumed?

2. **Correlation** How can temporal content produced by different communities, and in different mediums, be connected?

3. **Aggregation** What systems can manage Dynamic Web content and provide new interaction techniques to consume this information?

In answering each, many novel datasets and models were created that expand our understanding of the Dynamic Web. The scale of data also led to the development of many new algorithms, techniques, and interfaces. Below, I expand on this, highlighting the various contributions made in the previous chapters and discussing some future work.

9.1 Resonance

The process of identifying the relationship between content and use led to a number of studies that uniquely contribute to the understanding of the dynamics of the WWW. In

particular, by making use of the MSN click logs (described in Chapter 3), two unique datasets were built: (1) A fine-granularity (once an hour crawl augmented by sub-hourly crawls) collection of Web pages that were utilized during the study period (rather than the large portions of the "unvisited" Web), and (2) a collection of revisitation patterns on *specific pages* at sufficient scales to make quantitative analysis of revisitation possible at a *page* level.

9.1.1 Content Change

One of the most significant gaps in Dynamic Web understanding has been the lack of research on the evolution of those pages that are *continually used*. Any system that utilizes models of page change (e.g., a crawler, a search engine, etc.) potentially relies on numbers and statistics that do not apply to the pages that are likely to be used continuously. Because researchers have not had access to large traces of user behavior, the traditional approach to selection of pages for evolution studies is simply a random selection. However, as most pages are never visited once they have been created, there is very little sense of how this critical subset (the "visited Web") differs from the Web at large. The disadvantage of completely random crawls is that they may not capture the true rates of change.

Furthermore, the studies on Web evolution, thus far, have focused on re-crawling for changes at fairly slow rates (the fastest are at most once a day). Because there is a great deal of data that changes faster than once a day (sometimes significantly faster), studying the Dynamic Web in the context of use requires crawls at faster rates. Derived models based on these slow random crawls may–and as demonstrated here, *do*–misestimate the dynamics of the portions of the Web that are actually used.

The selection of data that is used in the study reported in Chapter 3 is unique in that it a) represents a diverse sample of the visited Web and b) is crawled at a faster rate than any other previous study (with both hourly and sub-hourly crawls).

The dataset enabled the calculation of change rates for different types of pages and sites as well as the Web in general. For example, whereas earlier studies indicated that 65% of the Web is unchanged after 11 weeks [59], the work presented here concluded that most

pages (66% of the visited Web) change within 5 weeks. More surprisingly, over 40% of pages change every hour.

Although such numbers may change in the future, the study demonstrates a number of techniques and models that are likely to remain useful. For example, the study introduces the notion of a change curve–a model of the evolution of a page over time that describes the "decay" of content on the page and subsequent equal (dis)similarity to some original starting point. This two-segment, linear model appears to be common across many of the pages observed. However, many pages would not appear to display this behavior without the fast crawl rates (i.e., many knot points appear earlier than a day). The two-segment model is useful for identifying the ratio of static to dynamic content on the page and how long it takes for the dynamic content to be replaced.

Additionally, this work introduced techniques for discriminating between various types of terms. By quantifying the survivability (staying power) of terms coupled with divergence, it is possible to identify terms that are unique to the page that persist for long periods (i.e., terms from the page's core language model) or vanish quickly. Similarly, terms that are not unique for the page and survive for long periods were identified primarily as the navigation information. Using these measures, a search engine might identify "good" or "bad" terms for including (or rejecting) in the index.

Finally, Chapter 3 described the change rates of the structure of the document. To make this large scale analysis possible, new algorithms were developed to rapidly compare many similar versions of the same page to detect when a DOM element is removed. Whereas previous algorithms traditionally compare two versions of the DOM tree, The algorithm (fully described in Appendix A) works by serializing and processing many copies in seconds.

9.1.2 Revisitation

Another area of active research on the Dynamic Web has been the study of revisitation–or the revisitation of pages over time. Here also research has been limited by the lack of large scale data sources. The traditional approach to revisitation work has been through either surveys or small scale log studies extracted from participants willing to instrument their

browsers. While this type of study design is able to ascertain revisitation properties for *users* (e.g., the percentage of a user's clicks that are revisits), not much can be said on a *page* or *site* level. This is due to the fact that the chance of two or more users, in a small study population, revisiting the same page is very small (for all but the most popular pages).

The study described in Chapter 4 has uniquely been able to describe page-level revisitation characteristics. This work has defined the notion of a revisitation curve (a normalized histogram of revisitation intervals) and applied this to the categorization and analysis of different types of pages. The study specifically identified 4 main revisitation curves (fast, medium, slow, and hybrid) corresponding to the peak(s) in the histogram (indicating the time when more than expected number of users revisit a page).

By additionally categorizing pages through additional properties (e.g., depth within sites, domain type, page type, keywords within the page), the study demonstrated particular characteristics of different pages (and types of pages) in terms of consumption patterns.

The implications of this categorization are that there is a fairly small and consistent set of patterns that are observed across the Web. Using these, it maybe possible to react different to these patterns. For example, when information is quick and short (the fast pattern) individuals may be more open to new content as they are not participating in monitoring activities which require the information they have seen before. Additional applications include the use of different bookmarking and history technologies depending on the type of revisitation: where some (fast) revisitations are decayed quickly from the history, pages with medium revisit patterns may be pre-fetched, and additional search features may be added to support re-access to the slow revisitation pattern pages.

9.1.3 Revisitation + Change = Resonance

One difficulty with Web content generally, and perhaps even more so with Dynamic Web content is that it is frequently unclear what motivates individuals to revisit a page. There are many possible items on the page–ads, comments, news stories, automated system info, etc.–each changing at a different rate. Understanding the relationship between these variously changing page pieces with the way they are consumed over time. Chapter 5 addresses this

hole in our understanding by studying both change and revisitation simultaneously.

While reconfirming that there *is* a relationship between change and revisitation–specifically that increased change correlates with increased revisitation (e.g., [127])–the study also identified some less obvious relationships. First, as might be evident from the discussion of motivations above, not every change should illicit a revisit. The study validated this by confirming that users *do not* synchronize their revisitation behavior to the overall page change rate. Furthermore, the study revealed that some revisits, particularly fast revisits, are more strongly related to change. On the other hand slow revisits appear to indicate a targeting of navigation and other static data.

While determining that there existed *some* relationship between revisitation and change is interesting, the combination of change and revisitation was also found to provide a powerful insight into the motivations for change. In particular, the relationship of the peak revisitation relative to the knot point was a powerful indicator about the *type* of information that is likely the target of revisitation–whether it is the static navigation information, fast changing headlines, or once daily product deals. The study, however, also went one step further.

Beyond identifying the type of information, it is desirable to identify the *specific* target of revisitation. Determining this presently requires expensive equipment (e.g., eye-tracking) or incomplete data (e.g., within-page cursor movements and mouse clicks). Chapter 5 describes a mechanism for automatically determining the likely target of revisitation by making use of change and revisitation data. By utilizing the novel serialization algorithm described in Appendix A, the frequency of change for each DOM element on the page was calculated. DOM elements were then either kept or removed depending on the similarity of their change rate to the peak revisitation rates. This notion of resonance, that people should synchronize their behavior to the rate of change of the content they find interesting, led to the creation of this novel algorithm for automatically identifying potential targets of interest.

9.1.4 Future Work

One of the issues with utilizing revisitation data is that this data still needs to come from somewhere. Although it may, in some situations, be more readily available than other behavioral signals, revisitation information is traditionally confidential information held by large search engine companies or private information held locally on a user's computer. That said, it may be possible to infer the general revisitation characteristics of a page given more easily collectible information. Features for doing this might include the classification (i.e., type) of the page, the depth within the site, and the link structure around that page. Using these, the general revisitation pattern for a page might be learned.

Additionally, there is an opportunity for future work in implementing and testing many of the systems described in this dissertation. For example, the use of resonance to enhance search snippets (as described in Chapter 4.4.

Finally, it is worth considering again the reality that models of the Dynamic Web might change as new technologies are introduced. Thus, it is important to continue to re-evaluate models and statistics of the Dynamic Web going into the future.

9.2 Correlation

An interesting side effect of collecting Dynamic Web data is that it is possible to identify reactions of individuals and communities not just to specific changes on the page but more broadly to topics. For example, it is possible to gauge public reaction to specific events (e.g., an election result) as well to longer running "topics" (e.g., some TV show) by observing the consumption and production of information over time. There has been recognition of the importance of studying population reactions as leading signals for events and trends but very little has been done on large scale datasets from many *different* types of Internet media.

Chapter 6 describes a study that uniquely utilized large temporally-annotated datasets from many different communities and forms of Web use. These datasets ranged in size from 27 million queries to 14 million blog posts to 20,000 news articles. The study specifically described methods of converting different types of data into normalized time series which

could then be grouped and compared. For example, one might want to know how and when the general searcher population react to a news story about some topic (in terms of increased queries). This reaction may then be compared to the blogger population, for example, for comparison and correlation analysis.

9.2.1 DTWExplorer and DTWRadar

The techniques described in Chapter 6 also included methods of identifying related terms and phrases to a given topic. This form of query expansion allowed the DTWExplorer system (also described in the study) to collect related time series for analysis. The DTWExplorer system offered analysts a mechanism for pairwise visualization of different time series (i.e., the behavioral response in two sources conditioned on some topic) with novel correlation analysis. The correlation analysis not only found the delay or shift of one time series' response relative to the other, but also utilized a novel Dynamic Time Warping (DTW) technique to align the different parts of the time series (e.g., the increase, plateau, and decrease in behavioral response).

In addition to the DTWExplorer tool, a novel visualization primitive, the DTWRadar, is introduced. This artifact provides a visual summary of the warp paths (or vector) necessary to transform one time series into the other. Each vector is then normalized and averaged to indicate to the user of the visualization which reaction precedes the other and by how much.

Using the dataset, DTW-based alignment and the DTWExplorer tool, a number of observations were made on the dataset highlighting situations (and possible causes) of one group's increased behavior relative to another. For example, differences in reaction of two different search engine populations (MSN and AOL) were described, as were possible motivational reasons for bloggers to seek out "bottom of the fold" articles.

9.2.2 Future Work

The DTWExplorer system offers a mechanism for analysts and expert users to evaluate lead/lag behavior of different time-varying streams of data. This is an important first step

in utilizing such signals for (automatic) predictive functionality. Future extensions to the DTWRadar might include additional predictive algorithms (i.e., time-series analysis algorithms) that could be supplied with additional predictive variables gleaned from exploratory data analysis and testing within the DTWExplorer tool.

Future modifications to the DTWExplorer system may also make it possible to integrate the system with Zoetrope. Doing so would allow users to utilize the Explorer tool as well as the DTWRadar visualizations to correlate information extracted from Web pages as well as combine those extractions with external content streams.

9.3 Aggregation, Manipulation, and Visualization of Dynamic Web Data

A difficulty in preserving Dynamic Web data is what to do with the once ephemeral information. Whereas before only one copy of the Web was preserved for easy access (the most recent), if many copies are kept, the question remains: how to support user interaction with this historical information.

Chapter 7 offers one possible solution in the form of Zoetrope. Zoetrope is a visual query system for Dynamic Web data that allows for the definition and execution of queries from within the context of the Now Web. The Zoetrope system allows users to select information on a page that they would like to track "backwards" through time (whether that information is constrained by the location on the page, a portion of the DOM tree, or contains some unique text).

Users manipulate the historical data by dropping "lenses" on the page. Lenses are a novel construct for exploring historical information by means of a slider that appears above the lens. As the slider is moved, the data within the lens changes to past states. Zoetrope lenses are a unique solution to accessing this type of historical data, but their power is further enhanced by a set of visualizations that can be generated based on the data within the lens. The textual, numerical, or image data with a lens can be transformed in a number of ways: static images (e.g., earthquake maps) can be transformed into movies and static price information (e.g., the used book price for a book on Amazon) can be rendered as a

time series.

Internally, Zoetrope offers a number of novel features including a custom crawler that stores not only the page content but also the rendered version, as it would have appeared to a user accessing the page at a particular time. This data is stored in a number of databases that are acted upon by a small set of operators (transform, filter, and render operators). The internal language can be used to form most of the Zoetrope operations and represent. In total, Zoetrope represents a complete system from crawling, to databases, to operators, to a front end that provides a novel interface to Dynamic Web data.

9.3.1 Future Work

The present instantiation of Zoetrope can be considered a first step in the creation of a more general tool for Dynamic Web data. Such an idealized solution might include the following features:

1. Fast, searchable access to historical versions of any *page with as* fine *a granularity as possible*

Realistically, this level of breadth and granularity may not be feasible. However, it may be possible to get closer to this reality with additional improvement to the Zoetrope crawling infrastructure. For example, because the crawler is built into Firefox, it may be possible to distribute crawling work to a community of individuals willing to run the required plug-ins. As individuals browse the Web, the pages they access will be recorded to disk. A peer-to-peer (P2P) network could connect this community with a Distributed Hash Table (DHT) providing access to page copies. This design is particularly attractive because, as discussed in Chapter 5, users revisit pages at rates that try to capture "interesting" content. However, such a P2P system may still require the occasional centralized snapshotting, so a hybrid P2P/Server architecture may be desirable. The centralized server may be useful in situations where not enough data is available in the peer to peer network thereby guaranteeing that at least some versions are available. This type of distribution would also require some mechanisms to ensure that the privacy of each participant is maintained (i.e., private bank pages should not be recorded and shared).

Having this data it is critical to:

2. Provide a rich mechanism for defining and analyzing different types of extractions.

Although Zoetrope provides numerous extraction mechanisms, there are many others that could be added. For example, at present Zoetrope does not support an query operations on images. Such extractions might be useful in identifying when *graphical* content changes (e.g., a figure enters a Webcam shot). Additionally, it might interesting to track content as it moves *off* one page and onto others. Features with increasing complexity, such as the ones described above, may introduce imprecision and require a user to iterate on their task. Such iterations, whenever possible, iterations should be minimized so:

3. Extractions should be as accurate as possible and easily testable from within the context of the page.

This requirement motivates the rapid testing feature available in Zoetrope (i.e., the ability to move the slider and instantly see previous versions). That said, there are extractions within Zoetrope that are time consuming. For example, the textual lens requires a number of comparisons and could be sped up greatly to ensure that Zoetrope continues to provide an interactive experience. Providing interactivity necessitates optimizations including custom indices, compression, and the ability to utilize any hints provided by the evolutionary models of the page or historical user behavior.

Part of the accuracy condition is minimizing gaps in the data. If we had been able to completely satisfy condition 1, gaps in page samples, and therefore gaps in data would be non-existent. Realistically, this may not be feasible as the speed of crawls is constrained by various resources. However, data need not be obtained solely from a single page. An extraction may provide hints to the system about the user's data requirements and additional content can be extracted from elsewhere on the Web.

Accuracy may also be enhanced by using Zoetrope as a debugging and learning environment for future extractions. Extractions that are marked incorrect and (possibly) fixed by the user can be remembered and used to learn better extractors in the future (potentially using techniques such as those described in [104]). One could also imagine that successful extractions can be automatically used in *new* pages. For example, a new lens dropped on a new page from a site can be automatically tested and adjusted based on other lenses created

within the site.

9.4 The Promise of the Dynamic Web

The Now Web is our present in more ways than one. It both represents the most recent version of the Web that most individuals and companies interact with, and also the paradigm under which we currently labor as researchers and system designers. The latter feature, unfortunately, impacts the former, leading to limitations to what users and systems can do.

The lack of research and development is itself limited by a history of insufficient resources (i.e., memory, disk, CPU) that make maintaining and manipulating many historical copies of the Web and obtaining large behavioral traces extremely difficult. Though many engineering problems remain, the constraints imposed by resources are slowly easing up. That said, there is still a fundamental lack of understanding in what the Dynamic Web actually looks like and how it can be used. This dissertation has addressed a number of these issues but many remain, providing a rich area for continued research and development.

Despite the early stages of temporal-informatics research, the results of this work and recent related efforts point towards the power of shifting from the paradigm of the Now Web to the Dynamic. Specifically, the Dynamic Web promises that *historical* data can provide *insight into the present* and *predictive power into the future*. Clearly, any system that attempts to model users in systems in the Now Web would benefit from a notion of historical trajectory. A user attempting to refind or revisit a Web page would prefer a system that respects past behavior in displaying results over a system that constant re-arranges results. Similarly, a user monitoring for change would likely prefer a system that notifies her of interesting changes (based on what she saw last time) and the ability to work with changing data over a system that simply displays the "Now." An individual running an advertising campaign would find benefit in a system that predicts ad demand and response over one that does not. Simply put: the more historical information, the more power is made available to end-users, designers, engineers and researchers. The caveat to this, of course, is that this data should be *useful* and *usable*. An uncontained flood of information will create more problems than it solves.

By understanding the flood of historical information and providing mechanisms to manipulate, manage, and interpret this data the work presented here attempts to provide building blocks for Dynamic Web applications. Using these blocks provides a basis for future work on the temporal aspects of the Web that will come closer and closer to taking advantage of the promise of the Dynamic Web.

Bibliography

[1] M. S. Ackerman, B. Starr, and M. Pazzani. The Do-I-Care Agent: Effective Social Discovery and Filtering on the Web. *Proceedings of RIAO*, 97:17–31, 1997.

[2] E. Adar, M. Dontcheva, J. Fogarty, and D. S. Weld. Zoetrope: interacting with the ephemeral web. In *UIST '08: Proceedings of the 21st annual ACM symposium on User interface software and technology*, pages 239–248, New York, NY, USA, 2008. ACM.

[3] E. Adar, M. Skinner, and D. S. Weld. Information arbitrage across multi-lingual Wikipedia. In *WSDM '09: Proceedings of the Second ACM International Conference on Web Search and Data Mining*, pages 94–103, New York, NY, USA, 2009. ACM.

[4] E. Adar, J. Teevan, and S. T. Dumais. Large scale analysis of web revisitation patterns. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 1197–1206, New York, NY, USA, 2008. ACM.

[5] E. Adar, J. Teevan, and S. T. Dumais. Resonance on the web: web dynamics and revisitation patterns. In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*, pages 1381–1390, New York, NY, USA, 2009. ACM.

[6] E. Adar, J. Teevan, S. T. Dumais, and J. L. Elsas. The Web changes everything: understanding the dynamics of Web content. In *WSDM '09: Proceedings of the Second ACM International Conference on Web Search and Data Mining*, pages 282–291, New York, NY, USA, 2009. ACM.

[7] E. Adar, D. S. Weld, B. N. Bershad, and S. S. Gribble. Why we search: visualizing and predicting user behavior. In *Proceedings of the 16th international conference on World Wide Web*, pages 161–170. ACM Press New York, NY, USA, 2007.

[8] J. Aizen, D. Huttenlocher, J. Kleinberg, and A. Novak. Traffic-based feedback on the web, 2004.

[9] J. Allan. *Introduction to topic detection and tracking.* Kluwer Academic Publishers, Norwell, MA, USA, 2002.

[10] J. Allan, J. Carbonell, G. Doddington, J. Yamron, and Y. Yang. Topic detection and tracking pilot study: Final report. In *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, volume 1998. Morgan Kaufmann Publishers, Inc, 1998.

[11] J. Allan, R. Gupta, and V. Khandelwal. Temporal summaries of new topics. *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 10–18, 2001.

[12] H. Andrew and K. David. Thresher: automating the unwrapping of semantic content from the World Wide Web. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 86–95, New York, NY, USA, 2005. ACM.

[13] P. G. Anick and R. A. Flynn. Versioning a full-text information retrieval system. *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 98–111, 1992.

[14] V. Anupam, J. Freire, B. Kumar, and D. Lieuwen. Automating Web navigation with the WebVCR. *Computer Networks*, 33(1-6):503–517, 2000.

[15] N. Ashish and C. A. Knoblock. Semi-Automatic Wrapper Generation for Internet Information Sources. In *CoopIS*, pages 160–169, 1997.

[16] A. Aula, N. Jhaveri, and M. Kki. Information search and re-access strategies of experienced web users. *Proceedings of the 14th international conference on World Wide Web*, pages 583–592, 2005.

[17] E. Z. Ayers and J. T. Stasko. Using Graphic History in Browsing the World Wide Web. In *WWW'95*, pages 11–14, 1995.

[18] R Baeza-Yates and B Ribeiro-Neto. *Modern information retrieval.* Addision Wesley, 1999.

[19] J. Bar-Ilan and B. C. Peritz. Evolution, continuity, and disappearance of documents on a specific topic on the web: A longitudinal study of "informetrics". *Journal of the American Society for Information Science and Technology*, 55(11):980–990, 2004.

[20] Z. Bar-Yossef, A. Z. Broder, R. Kumar, and A. Tomkins. Sic transit gloria telae: towards an understanding of the web's decay. *Proceedings of the 13th international conference on World Wide Web*, pages 328–337, 2004.

[21] M. Bauer. Infobeans–configuration of personalized information assistants. *Proceedings of the 4th international conference on Intelligent user interfaces*, pages 153–156, 1998.

[22] M. Bauer and G. Paul. Instructible information agents for web mining. *Proceedings of the 5th international conference on Intelligent user interfaces*, pages 21–28, 2000.

[23] Benjamin B. Bederson, Jesse Grosjean, and Jon Meyer. Toolkit design for interactive structured graphics. *IEEE Trans. Softw. Eng.*, 30(8):535–546, 2004.

[24] K. Berberich, S. Bedathur, T. Neumann, and G. Weikum. A time machine for text search. *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 519–526, 2007.

[25] M. Bhide, P. Deolasee, A. Katkar, A. Panchbudhe, K. Ramamritham, and P. Shenoy. Adaptive Push-Pull: Disseminating Dynamic Web Data. *IEEE Transactions on Computers*, 51(6):652–668, 2002.

[26] E. A. Bier, M. C. Stone, K. Pier, W. Buxton, and T. D. DeRose. Toolglass and magic lenses: the see-through interface. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 73–80, New York, NY, USA, 1993. ACM Press.

[27] J. P. Bigham, A. C. Cavender, R. S. Kaminsky, C. M. Prince, and T. S. Robison. Transcendence: enabling a personal view of the deep web. *Proceedings of the 13th international conference on Intelligent user interfaces*, pages 169–178, 2008.

[28] Nielsen BlogPulse. http://www.blogpulse.com, 2009.

[29] E. S. Boese and A. E. Howe. Effects of web document evolution on genre classification. *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 632–639, 2005.

[30] P. L. II Bogen, L. Francisco-Revilla, R. Furuta, T. Hubbard, U. P. Karadkar, and F. Shipman. Longitudinal study of changes in blogs. *Proceedings of the 2007 conference on Digital libraries*, pages 135–136, 2007.

[31] M. Bolin, P. Rha, and R. C. Miller. Automation and customization of rendered web pages. *Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 163–172, 2005.

[32] V. Boyapati, K. Chevrier, A. Finkel, N. Glance, T. Pierce, R. Stockton, and C. Whitmer. ChangeDetector ^{TM}: a site-level monitoring tool for the www. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 570–579, New York, NY, USA, 2002. ACM Press.

[33] P. Bramsen, P. Deshpande, Y. K. Lee, and R. Barzilay. Inducing temporal graphs. *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 189–198, 2006.

[34] B. E. Brewington and G. Cybenko. How dynamic is the web? *Computer Networks*, 33(1-6):257–276, 2000.

[35] A. Broder. On the resemblance and containment of documents. In *Proceedings of the Compression and Complexity of Sequences*, page 21, 1997.

[36] Nielsen BuzzMetrics. ICWSM Conference dataset, 2007.

[37] M. Cafarella, A. Halevy, D. Wang, E. Wu, and Y. Zhang. WebTables: exploring the power of tables on the web. *Proc. VLDB Endow.*, 1(1):538–549, 2008.

[38] L. D. Catledge and J. E. Pitkow. Characterizing browsing strategies in the world-wide web. *Computer Networks and ISDN Systems*, 27(6):1065–1073, 1995.

[39] S. Chakravarthy, A. Sanka, J. Jacob, and N. Pandrangi. A learning-based approach for fetching pages in WebVigiL. *Proceedings of the 2004 ACM symposium on Applied computing*, pages 1725–1731, 2004.

[40] B. Chan, L. Wu, J. Talbot, M. Cammarano, and P. Hanrahan. Vispedia: Interactive Visual Exploration of Wikipedia Data via Search-Based Integration. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1213–1220, 2008.

[41] C. Chang and S. Kuo. OLERA: Semisupervised Web-Data Extraction with Visual Support. *IEEE Intelligent Systems*, 19(6):56–64, 2004.

[42] C. H. Chang, M. Kayed, M. R. Girgis, and K. F. Shaalan. A survey of web information extraction systems. *IEEE Transactions on Knowledge and Data Engineering*, pages 1411–1428, 2006.

[43] ChangeDetect. ChangeDetect Web Page Monitoring, 2007.

[44] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. The TSIMMIS project: Integration of heterogeneous information sources. In *Proceedings of IPSJ Conference*, page 718, 1994.

[45] Y. F. Chen, E. Koutsofios, M. Hill, and N. Jersey. WebCiao: A Website Visualization and Tracking System. *Proceedings of WebNet97, Toronto, Canada, October*, 1997.

[46] S. Chien and N. Immorlica. Semantic similarity between search engine queries using temporal correlation. In *Proceedings of the 14th international conference on World Wide Web*, pages 2–11. ACM New York, NY, USA, 2005.

[47] S. Y. Chien, V. J. Tsotras, C. Zaniolo, and D. Zhang. Efficient Complex Query Support for Multiversion XML Documents. *Lecture Notes in Computer Science*, pages 161–178, 2002.

[48] J. Cho and H. Garcia-Molina. The evolution of the web and implications for an incremental crawler. *Very Large Data Bases: VLDB'00, Proceedings of the 26th International Conference on Very Large Data Bases, Cairo, Egypt*, 2000.

[49] H. Choi and H. Varian. Predicting the present with google trends. Technical report, Technical Report, Google Inc, 2009.

[50] A. Cockburn and S. Greenberg. Issues of page representation and organisation in web browsers revisitation tools. *AJIS*, 7(2), 2000.

[51] A. Cockburn and B. McKenzie. What do web users do? an empirical analysis of web use. *International Journal of Human-Computer Studies*, 54(6):903–922, 2001.

[52] S. Cronen-Townsend, Y. Zhou, and W. B. Croft. Predicting query performance. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 299–306, New York, NY, USA, 2002. ACM.

[53] A. Dasgupta, A. Ghosh, R. Kumar, C. Olston, S. Pandey, and A. Tomkins. The discoverability of the web. *Proceedings of the 16th international conference on World Wide Web*, pages 421–430, 2007.

[54] M. Dontcheva, S. M. Drucker, D. Salesin, and M. F. Cohen. Changes in Webpage Structure over Time. Technical report, University of Washington, CSE, 2007.

[55] M. Dontcheva, S. M. Drucker, D. Salesin, and M. F. Cohen. Relations, cards, and search templates: user-guided web data integration and layout. *Proceedings of the 20th annual ACM symposium on User interface software and technology*, pages 61–70, 2007.

[56] M. Dontcheva, S. M. Drucker, G. Wade, D. Salesin, and M. F. Cohen. Summarizing personal web browsing sessions. *Proceedings of the 19th annual ACM symposium on User interface software and technology*, pages 115–124, 2006.

[57] F. Douglis, T. Ball, Y. F. Chen, and E. Koutsofios. The AT&T Internet Difference Engine: Tracking and viewing changes on the web. *World Wide Web*, 1(1):27–44, 1998.

[58] F. Douglis, A. Feldmann, B. Krishnamurthy, and J. Mogul. Rate of change and other metrics: a live study of the world wide web. *Proceedings of the USENIX Symposium on Internet Technologies and Systems on USENIX Symposium on Internet Technologies and Systems table of contents*, pages 14–14, 1997.

[59] D. Fetterly, M. Manasse, M. Najork, and J. L. Wiener. A large-scale study of the evolution of web pages. *Software Practice and Experience*, 34(2):213–237, 2004.

[60] E. Filatova and E. Hovy. Assigning time-stamps to event-clauses. *Proceedings of the workshop on Temporal and spatial information processing-Volume 13*, pages 1–8, 2001.

[61] K. Fishkin and E. Bier. Webtracker-a web service for tracking documents. *Proceedings of 6th International World Wide Web Conference (WWW 1997)*, 1997.

[62] J. A. Fitzpatrick, Reffell J., and M. Aydelott. Breakingstory: visualizing change in online news. In *CHI '03: CHI '03 extended abstracts on Human factors in computing systems*, pages 900–901, New York, NY, USA, 2003. ACM Press.

[63] S. Flesca, F. Furfaro, and E. Masciari. Monitoring web information changes. *International Conference on Information Technology: Coding and Computing (ITCC01)*, 2001.

[64] J. Freire, B. Kumar, and D. Lieuwen. Webviews: accessing personalized web content and services. *Proceedings of the 10th international conference on World Wide Web*, pages 576–586, 2001.

[65] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting. *The Annals of Statistics*, 28(2):337–374, 2000.

[66] J. Fujima, A. Lunzer, K. Hornbk, and Y. Tanaka. Clip, connect, clone: combining application elements to build custom interfaces for information access. *Proceedings of the 17th annual ACM symposium on User interface software and technology*, pages 175–184, 2004.

[67] E. Gabrilovich, S. Dumais, and E. Horvitz. Newsjunkie: providing personalized newsfeeds via analysis of information novelty. *Proceedings of the 13th international conference on World Wide Web*, pages 482–490, 2004.

[68] D. Gibson, K. Punera, and A. Tomkins. The volume and evolution of web page templates. *International World Wide Web Conference*, pages 830–839, 2005.

[69] J. Ginsberg, M.H. Mohebbi, R.S. Patel, L. Brammer, M.S. Smolinski, and L. Brilliant. Detecting influenza epidemics using search engine query data. *Nature*, 457(7232):1012–1014, 2008.

[70] Google. http://google.com/trends, 2009.

[71] F. Grandi. An Annotated Bibliography on Temporal and Evolution Aspects in the World Wide Web. Technical report, TimeCenter Technical Report, 2003.

[72] F. Grandi. Introducing an annotated bibliography on temporal and evolution aspects in the world wide web. *SIGMOD Rec.*, 33(2):84–86, 2004.

[73] S. Greenberg and M. Boyle. Generating custom notification histories by tracking visual differences between web page visits. *Proceedings of the 2006 conference on Graphics interface*, pages 227–234, 2006.

[74] S. Greenberg and M. Boyle. Generating custom notification histories by tracking visual differences between web page visits. In *GI '06: Proceedings of Graphics Interface 2006*, pages 227–234, Toronto, Ont., Canada, Canada, 2006. Canadian Information Processing Society.

[75] S. Greenberg and A. Cockburn. Getting back to back: Alternate behaviors for a web browser's back button. *Proceedings of the 5th Annual Human Factors and the Web Conference*, 1999.

[76] D. Gruhl, R. Guha, R. Kumar, J. Novak, and A. Tomkins. The predictive power of online chatter. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 78–87. ACM New York, NY, USA, 2005.

[77] S. Harabagiu and C. A. Bejan. An answer bank for temporal inference. In *Proceedings of LREC*, 2006.

[78] B. Hartmann, L. Wu, K. Collins, and S. Klemmer. Programming by a sample: Rapidly prototyping web applications with d. mix. *In Proceeding of the 20th Symp. on User Interface Software and Technology (UIST07). Newport, RI, USA*, 2007.

[79] S. Havre, E. Hetzler, P. Whitney, and L. Nowell. ThemeRiver: Visualizing Thematic Changes in Large Document Collections. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):9–20, 2002.

[80] E. Herder. Characterizations of user web revisit behavior. *Proceedings of Workshop on Adaptivity and User Modeling in Interactive Systems*, 2005.

[81] M. Herscovici, R. Lempel, and S. Yogev. Efficient indexing of versioned document sequences. *Proc. of the 29th European Conf. on Information Retrieval*, 2007.

[82] D. Huynh, S. Mazzocchi, and D. Karger. Piggy bank: Experience the semantic web inside your web browser. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(1):16–27, 2007.

[83] D. F. Huynh, D. R. Karger, and R. C. Miller. Exhibit: lightweight structured data publishing. *Proceedings of the 16th international conference on World Wide Web*, pages 737–746, 2007.

[84] D. F. Huynh, R. C. Miller, and D. R. Karger. Enabling web browsers to augment web sites' filtering and sorting functionalities. *Proceedings of the 19th annual ACM symposium on User interface software and technology*, pages 125–134, 2006.

[85] D. F. Huynh, R. C. Miller, and D. R. Karger. Potluck: Data mash-up tool for casual users. *Lecture Notes in Computer Science*, 4825:239, 2007.

[86] K. Ito and Y. Tanaka. A visual environment for dynamic web application composition. *Proceedings of the fourteenth ACM conference on Hypertext and hypermedia*, pages 184–193, 2003.

[87] A. Jatowt and M. Ishizuka. Temporal multi-page summarization. *Web Intelligence and Agent Systems*, 4(2):163–180, 2006.

[88] A. Jatowt, Y. Kawai, H. Ohshima, and K. Tanaka. What can history tell us?: towards different models of interaction with document histories, 2008.

[89] A. Jatowt, Y. Kawai, and K. Tanaka. Temporal ranking of search engine results. *Proceedings of the The Fifth International Conference on Web Information Systems Engineering (WISE2005)*, page 4352, 2005.

[90] A. Jatowt, Y. Kawai, and K. Tanaka. Visualizing historical content of web pages. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 1221–1222, New York, NY, USA, 2008. ACM.

[91] W. Jones, S. Dumais, and H. Bruce. Once found, what then? a study of "keeping" behaviors in the personal use of web information. *Proceedings of the American Society for Information Science and Technology*, 39(1):391–402, 2002.

[92] S. Kaasten and S. Greenberg. Designing an integrated bookmark/history system for web browsing. *Western Computer Graphics Symposium*, 2000.

[93] G. Karypis. CLUTO-a clustering toolkit, 2003.

[94] M. Kellar, C. Watters, and K. M. Inkpen. An exploration of web-based monitoring: implications for design. *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 377–386, 2007.

[95] M. Kellar, C. Watters, and M. Shepherd. A goal-based classification of web information tasks. *Proceedings of ASIST06*, 2006.

[96] E. Keogh, J. Lin, and A. Fu. Hot sax: Efficiently finding the most unusual time series subsequence. In *Fifth IEEE International Conference on Data Mining*, page 8, 2005.

[97] E. J. Keogh and M. J. Pazzani. Derivative dynamic time warping. In *First SIAM International Conference on Data Mining (SDM2001)*, 2001.

[98] S. J. Kim and S. H. Lee. An empirical study on the change of web pages. *the 7th Asia Pacific Web Conference*, pages 632–642, 2005.

[99] A.W. Klein, P. J. Sloan, A. Finkelstein, and M.F. Cohen. Stylized video cubes. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 15–22, New York, NY, USA, 2002. ACM Press.

[100] J. Kleinberg. Bursty and hierarchical structure in streams. *Data Mining and Knowledge Discovery*, 7(4):373–397, 2003.

[101] J. Kleinberg. Temporal dynamics of on-line information streams. *Data Stream Management: Processing High-Speed Data Streams, Springer*, 2005.

[102] W. Koehler. Web page change and persistence–a four-year longitudinal study. *Journal of the American Society for Information Science and Technology*, 53(2):162–171, 2002.

[103] W. Koehler. A longitudinal study of web pages continued: a consideration of document persistence. *Information Research*, 9(2):9–2, 2004.

[104] N. Kushmerick. *Wrapper Induction for Information Extraction*. PhD thesis, University of Washington, 1997.

[105] N. Kushmerick, D. S. Weld, and R. B. Doorenbos. Wrapper induction for information extraction. *IJCAI*, pages 729–737, 1997.

[106] Microsoft Live Labs. Accelerating search in academic research, 2006.

[107] A. H. F. Laender, B. A. Ribeiro-Neto, A. S. da Silva, and J. S. Teixeira. A brief survey of web data extraction tools. *ACM SIGMOD Record*, 31(2):84–93, 2002.

[108] V. Lavrenko, M. Schmill, D. Lawrie, P. Ogilvie, D. Jensen, and J. Allan. Mining of concurrent text and time series. In *KDD-2000 Workshop on Text Mining*, 2000.

[109] Facebook Lexicon. http://www.facebook.com/lexicon/, 2009.

[110] J. Lin, E. Keogh, and S. Lonardi. Visualizing and discovering non-trivial patterns in large time series databases. *Information visualization*, 4(2):61–82, 2005.

[111] G. Little, T. A. Lau, A. Cypher, J. Lin, E. M. Haber, and E. Kandogan. Koala: capture, share, automate, personalize business processes on the web. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 943–946. ACM Press New York, NY, USA, 2007.

[112] B. Liu, K. Zhao, and L. Yi. Visualizing web site comparisons. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 693–703, New York, NY, USA, 2002. ACM Press.

[113] L. Liu, W. Tang, D. Buttler, and C. Pu. Information monitoring on the web: A scalable solution. *World Wide Web*, 5(4):263–304, 2002.

[114] Ling Liu, Calton Pu, and Wei Tang. WebCQ-detecting and delivering information changes on the web. In *CIKM '00: Proceedings of the ninth international conference on Information and knowledge management*, pages 512–519, New York, NY, USA, 2000. ACM Press.

[115] M. Livny, R. Ramakrishnan, K. Beyer, G. Chen, D. Donjerkovic, S. Lawande, J. Myllymaki, and K. Wenger. Devise: integrated querying and visual exploration of large datasets. *SIGMOD Rec.*, 26(2):301–312, 1997.

[116] A. Lunzer and K. Hornbk. Recipesheet: creating, combining and controlling information processors. *Proceedings of the 19th annual ACM symposium on User interface software and technology*, pages 145–154, 2006.

[117] I. Mani, B. Schiffman, and J. Zhang. Inferring temporal ordering of events in news. *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: companion volume of the Proceedings of HLT-NAACL 2003–short papers-Volume 2*, pages 55–57, 2003.

[118] I. Mani and G. Wilson. Temporal granularity and temporal tagging of text. *AAAI-2000 Workshop on Spatial and Temporal Granularity, Austin*, 2000.

[119] A. Marian, S. Abiteboul, G. Cobena, and L. Mignet. Change-centric management of versions in an xml warehouse. *Proceedings of VLDB 2001*, page 581590, 2001.

[120] N. Milic-Frayling, R. Jones, K. Rodden, G. Smyth, A. Blackwell, and R. Sommerer. Smartback: supporting users in back navigation. *Proceedings of the 13th international conference on World Wide Web*, pages 63–71, 2004.

[121] R. C. Miller and B. A. Myers. Creating Dynamic World Wide Web Pages By Demonstration. Technical report, School of Computer Science, Carnegie Mellon University, 1997.

[122] J. B. Morrison, P. Pirolli, and S. K. Card. A taxonomic analysis of what world wide web activities significantly impact people's decisions and actions. *Conference on Human Factors in Computing Systems*, pages 163–164, 2001.

[123] C. S. Myers and L. R. Rabiner. A comparative study of several dynamic time-warping algorithms for connected word recognition. *The Bell System Technical Journal*, 60(7):1389–1409, 1981.

[124] A. Nadamoto and K. Tanaka. A comparative web browser (CWB) for browsing and comparing web pages. *Proceedings of the 12th international conference on World Wide Web*, pages 727–735, 2003.

[125] K. Nørvåg. Algorithms for temporal query operators in XML databases. *Proceedings of Workshop on XML-Based Data Management (XMLDM)(in conjunction with EDBT2002)*, 2002.

[126] A. Ntoulas, J. Cho, and C. Olston. What's new on the web?: the evolution of the web from a search engine perspective. *Proceedings of the 13th international conference on World Wide Web*, pages 1–12, 2004.

[127] H. Obendorf, H. Weinreich, E. Herder, and M. Mayer. Web page revisitation revisited: implications of a long-term click-stream study of browser usage. *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 597–606, 2007.

[128] C. Olston and S. Pandey. Recrawl scheduling based on information longevity. *Proceedings of WWW 2008, April 2125, 2008, Beijing, China.*, 2008.

[129] OneRiot. http://www.oneriot.com, 2009.

[130] N. Pandrangi, J. Jacob, A. Sanka, and S. Chakravarthy. WebVigiL: User Profile-Based Change Detection for HTML/XML Documents. *Twentieth British National Conference on Databases*, 2003.

[131] G. Pass, A. Chowdhury, and C. Torgeson. A picture of search. In *Proceedings of the 1st international conference on Scalable information systems*. ACM New York, NY, USA, 2006.

[132] Perpetually.com. http://www.perpetually.com, 2009.

[133] T. A. Phelps and R. Wilensky. Robust hyperlinks and locations. *D-Lib Magazine*, 6(7/8), July/August 2000.

[134] J. Pitkow and P. Pirolli. *Life, death, and lawfulness on the electronic frontier*. ACM New York, NY, USA, 1997.

[135] B. Pollak and W. Gatterbauer. Creating Permanent Test Collections of Web Pages for Information Extraction Research. In *Proceedings of the 33nd International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2007)*, volume II. ICS AS CR (Institute of Computer Science, Academy of Sciences of the Czech Republic), January 20–26, 2007.

[136] Open Directory Project. Open directory project, http://www.dmoz.org, 2008.

[137] M. Qiang, S. Miyazaki, and K. Tanaka. Webscan: Discovering and notifying important changes of web sites. *Lecture Notes in Computer Science*, pages 587–598, 2001.

[138] E. Rennison. Galaxy of news: an approach to visualizing and understanding expansive news landscapes. In *UIST '94: Proceedings of the 7th annual ACM symposium on User interface software and technology*, pages 3–12, New York, NY, USA, 1994. ACM Press.

[139] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *Readings in Speech Recognition*, page 159, 1990.

[140] M. Sanderson and S. Dumais. Examining repetition in user search behavior. *Lecture Notes in Computer Science*, 4425:597, 2007.

[141] N. Sato, M. Euhara, and Y. Sakai. Ftf-idf scoring for fresh information retrieval. *AINA 2004: Proceedings of the 18th International Conference on Advanced Information Networking and Applications*, 1, 2004.

[142] m.c. schraefel, Y. Zhu, D. Modjeska, D. Wigdor, and S. Zhao. Hunter gatherer: interaction support for the creation and management of within-web-page collections. *Proceedings of the 11th international conference on World Wide Web*, pages 172–181, 2002.

[143] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys (CSUR)*, 34(1):1–47, 2002.

[144] A. J. Sellen, R. Murphy, and K. L. Shaw. How knowledge workers use the web. *Proceedings of the SIGCHI conference on Human factors in computing systems: Changing our world, changing ourselves*, pages 227–234, 2002.

[145] C. Stolte, D. Tang, and P. Hanrahan. Polaris: A System for Query, Analysis, and Visualization of Multidimensional Relational Databases. In *IEEE Transactions on Visualization and Computer Graphics*, 2002.

[146] A. Sugiura and Y. Koseki. Internet scrapbook: automating web browsing tasks by demonstration. In *UIST '98: Proceedings of the 11th annual ACM symposium on User interface software and technology*, pages 9–18, New York, NY, USA, 1998. ACM Press.

[147] H. Takano and T. Winograd. Dynamic bookmarks for the WWW. *Proceedings of the ninth ACM conference on Hypertext and hypermedia: links, objects, time and space—structure in hypermedia systems: links, objects, time and space—structure in hypermedia systems*, pages 297–298, 1998.

[148] Y. Tanaka, K. Ito, and J. Fujima. Meme media for clipping and combining web resources. *World Wide Web*, 9(2):117–142, 2006.

[149] Y. Tanaka, K. Ito, and D. Kurosaki. Meme media architectures for re-editing and redistributing intellectual assets over the web. *International Journal of Human-Computer Studies*, 60(4):489–526, 2004.

[150] L. Tauscher and S. Greenberg. How people revisit web pages: empirical findings and implications for the design of history systems. *International Journal of Human Computer Studies*, 47:97–138, 1997.

[151] J. Teevan. The re:search engine: simultaneous support for finding and re-finding. In *UIST '07: Proceedings of the 20th annual ACM symposium on User interface software and technology*, pages 23–32, New York, NY, USA, 2007. ACM.

[152] J. Teevan, E. Adar, R. Jones, and M. A. S. Potts. Information re-retrieval: repeat queries in yahoo's logs. *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 151–158, 2007.

[153] J. Teevan, S. T. Dumais, D. J. Liebling, and R. L. Hughes. Changing how people view changes on the web. In *UIST '09: Proceedings of the 22nd annual ACM symposium on User interface software and technology*, pages 237–246, New York, NY, USA, 2009. ACM.

[154] Microsoft Bing Real time Twitter Search. http://www.bing.com/twitter, 2009.

[155] R. Tuchinda, P. Szekely, and C. A. Knoblock. Building data integration queries by demonstration. *Proceedings of the 12th international conference on Intelligent user interfaces*, pages 170–179, 2007.

[156] R. Tuchinda, P. Szekely, and C. A. Knoblock. Building mashups by example. *Proceedings of the 13th international conference on Intelligent user interfaces*, pages 139–148, 2008.

[157] E. R. Tufte. *Beautiful evidence*. Graphics Press Cheshire, Conn., 2006.

[158] TweetMeme. http://www.tweetmeme.com/, 2009.

[159] M. Uehara and N. Sato. Information retrieval based on temporal attributes in WWW archives. *Proceedings of the 11th International Conference on Parallel and Distributed Systems*, 1, 2005.

[160] J. J. Van Wijk and E. R. Van Selow. Cluster and calendar based visualization of time series data. *InfoVis'99: Proceedings of the 1999 IEEE Symposium on Information Visualization*, pages 4–9, 1999.

[161] F. B. Viegas, M. Wattenberg, F. van Ham, J. Kriss, and M. McKeon. ManyEyes: a Site for Visualization at Internet Scale. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1121–1128, 2007.

[162] F. Vitali. Versioning hypermedia. *ACM Computing Surveys (CSUR)*, 31(4es), 1999.

[163] M. Vlachos, C. Meek, Z. Vagena, and D. Gunopulos. Identifying similarities, periodicities and bursts for online search queries. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 131–142. ACM New York, NY, USA, 2004.

[164] F. Wang, X. Zhou, and C. Zaniolo. Bridging relational database history and the web: the XML approach. *Proceedings of the eighth ACM international workshop on Web information and data management*, pages 3–10, 2006.

[165] J. Wang and F. H. Lochovsky. Data extraction and label assignment for web databases. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 187–196, New York, NY, USA, 2003. ACM.

[166] M. Weber, M. Alexa, and W. Muller. Visualizing time-series on spirals. *INFOVIS 2001: Proceedings of the IEEE Symposium on Information Visualization*, pages 7–13, 2001.

[167] J. R. Wen, J. Y. Nie, and H. J. Zhang. Query clustering using user logs. *ACM Transactions on Information Systems*, 20(1):59–81, 2002.

[168] R. W. White and S. M. Drucker. Investigating behavioral variability in web search. *Proceedings of the 16th international conference on World Wide Web*, pages 21–30, 2007.

[169] W. Willett, J. Heer, and M. Agrawala. Scented widgets: Improving navigation cues with embedded visualizations. *IEEE Transactions on Visualization and Computer Graphics*, pages 1129–1136, 2007.

[170] A. Witkin. Scale-space filtering. In *8th Int. Joint Conf. Artificial Intelligence*, volume 2, pages 1019–1022, 1983.

[171] J. Wong and J. I. Hong. Making mashups with marmite: towards end-user programming for the web. *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1435–1444, 2007.

[172] J. Zhang and T. Suel. Efficient search in large textual collections with redundancy. *Proceedings of the 16th international conference on World Wide Web*, pages 411–420, 2007.

Appendix A

Structural Analysis

One particular difficulty of understanding Web pages over extended periods is the lack of techniques to scalably analyze structural changes over many versions of the same page. Structural changes represent an important way a page can change. A page with exactly the same textual content as a previous version can still be very different because that content is organized and presented in a different way. Such changes frequently break extractors that rely on structural stability. Additionally, there are a number of tasks that require being able to simultaneously compare many versions of the same page (e.g., calculating the change rate of an element, determining the path blocks of data on the page may move, etc.).

To do so we develop a simple algorithm that serializes an HTML document in such a way that different forms of lexicographic sorting and aggregation can answer these questions. A limitation of this approach is that the algorithm is unable to detect the amount of change in content directly (e.g., minor changes in the textual content of a particular block of HTML are treated the same as a large change). These type of changes require a more in-depth analysis of the tree and the text. A feature of the algorithm is that it may identify situations in which a more traditional tree-comparison algorithm may work better. Thus, the technique is still beneficial in processing the common cases. Despite these limitations, our technique has the significant advantage of being implementable as a basic map-reduce style problem and scaled to many copies of the same page.

More concretely, to generate the serialized version, the algorithm proceeds by progressing

in a depth first parse of the HTML tree of the document through a standard HTML parser (HTML documents are "cleaned" with the *tidy* tool to ensure that they are valid XML). When the system reaches the start of a new HTML element (e.g., "`<div>`" or "`<h1>`") the algorithm notes this in a stack. Textual content that is encountered during the parsing is also pushed onto a stack. When an ending element tag is reached (e.g., "`</b>`" or "`</div>`") the system emits a hash value of the text within the tags (as well as a hash of all the text in children of that tag). The system will also output the XPath used to arrive at the nodes in both set and direct access notation. For example, given `<a>foo <b>bar</b></a>` at time k, the system will output a stream of two lines:

full_path	*type_path*	*.idx*	*node_hash*	*subtree_hash*	*version*
`/a[0]/b[0]`	`(/a/b)`	`[0]`	HASH("`bar`")	HASH("`bar`")	k
`/a[0]`	`(/a)`	`[0]`	HASH("`foo`")	HASH("`foo bar`")	k

Each line corresponds to a temporally-annotated tuple: {*full_path*, *type_path*, *node_hash*, *subtree_hash*, *version*}. The type path reflects the full typing of the node based on its position in the hierarchy. Although the two are equivalent, we find that having two versions of the path, and in particular the type path, useful for detecting motion of elements between tree siblings. Similarly, having both a hash of the content of the node as well as the subtree allows us to isolate the cause of a change. We create this output for every version of the document. For example, if the $k+1$ version of the document contained the text `<a>foo <b>baz</b></a>` our stream would be:

full_path	*type_path*	*.idx*	*node_hash*	*subtree_hash*	*version*
`/a[0]/b[0]`	`(/a/b)`	`[0]`	HASH("`baz`")	HASH("`baz`")	$k+1$
`/a[0]`	`(/a)`	`[0]`	HASH("`foo`")	HASH("`foo baz`")	$k+1$
`/a[0]/b[0]`	`(/a/b)`	`[0]`	HASH("`bar`")	HASH("`bar`")	k
`/a[0]`	`(/a)`	`[0]`	HASH("`foo`")	HASH("`foo bar`")	k

We may additionally keep track of other features for later filtering (e.g., the length of the text, hash of tag attributes, etc.).

At present we simply utilize an MD5 hashing algorithm, but the use of shingles as a hash value [17] is worth exploring in the future as we can take advantage of our algorithm to rapidly eliminate exact matches, and testing approximate ones when necessary.

A.1 SORT and REDUCE Operations

We define two operations on this dataset: SORT(sorting variables), which outputs a sorted stream, and REDUCE(reduction variables), which outputs a set of sets. The SORT function corresponds to "map" but for our purposes is a simple lexicographic sort of the serialized stream in the order of the sorting variables (e.g., SORT(*node_hash*,*version*) would first sort by the *node_hash* and then by *version*). For convenience, a − (i.e., negation) in front of the variable indicates a reverse sort (e.g., SORT(−*version*) sorts in descending order). The REDUCE operation groups sorted data based on some matching parameters. For example, REDUCE(*version*) would group lines into a set (with sort order maintained) based on the equivalence of the *version* column: $\{\{k\},\{k+1\},\{k+n\}\}$. Thus, the first set contains all lines from version k.

By sorting on different columns in different orders and linearly processing the files, we are able to track the movement and lifespan of a specific piece of text in the hierarchy, track the number and lifespan of pieces of text at a specific location, and combine these operations in various ways.

The benefit of our approach is that it is a) linear in the number of tree nodes and versions, and b) can be parallelized by mapping different sub-trees to different processors.

A.2 Tracking DOM Element Lifespan

Using the mechanism described above we are able to calculate the lifespan of various DOM elements within our crawled pages over time. We achieve this by applying:

Algorithm 2 $lifespan$

1: SORT($full\_path$,$version$)
2: $S \leftarrow$ REDUCE($full\_path$)
3: **for** each s in S **do**
4: calculate the difference between the minimum version id and last reported id (i.e., $s[s.length-1].version - s[0].version$)

Letting S be the set of sets, s a subset within S. This algorithm is somewhat of a simplification as, in reality, we only test for the vanishing point for those paths present in the original crawl. Additionally, we ignore nodes that have not survived the first hour since these tend to be unrelated to document structure and frequently contain simple formatting.

A.3 Analyzing Blocks

Extending the ideas above, we can further calculate the persistence of any content anywhere within the document. This is particularly interesting for situations where content may move (as in the case of blogs or news sites) but also informs us about how long we might expect certain content to last on the page before vanishing. To calculate this we can execute the following (further defining t to be a tuple within s, and t.version, t.type_path, etc. to be the tuples values for those variables):

Algorithm 3 *blockAnalysis*

1: SORT($subtree\_hash, full\_path, version$)
2: $S \leftarrow$ REDUCE($subtree\_hash$)
3: **for** each s in S **do**
4: find the last version in which the hash appears, calculate for each t in S the time difference between the tuple and the last time, marking the DOM element with this count for later averaging.

Again the algorithm is a slight simplification for readability as we only consider the initial time that the content entered the node. The output of this analysis is a measure of how long we can expect a piece of text that starts at a given location to survive.

Figure A.1 shows a simple visual rendering of four sample pages from our crawl processed using this approach. The bright red regions are those that change the most rapidly, and the white regions are the most stable. Navigation and other persistent elements (such as search bars) tend to have a high expectation of survival, whereas text ads may not survive between versions. Notice, for example, the navigation bar on the left side of the amazon.com page

Figure A.1: Renderings of the lifespan of elements on a number of pages (darker red blocks are shorter life spans) including a) boston.com, b) televisionwithoutpity.com (note the groups of similarly colored content), c) the DVD bestseller list on Amazon, d) gas prices in various cities on GasBuddy.com, and e) a list of earthquakes at the USGS. Not all blocks marked.

in Figure A.1c which contains highly survivable content as well as the #1 bestseller (at the top) which persisted much longer than other elements in the list.

This algorithm can be adapted to group different nodes with similar average life spans (e.g., when a new article is posted on the top of a blog, generally all posts below move, thus the average change rate of all posts is matched). By additionally considering the rendered location of the DOM element, it may be possible to find those elements that are visually close and have similar lifespan characteristics. Blocks can also be identified by measuring and grouping by the times new content is detected within a DOM element.

Metrics for structural changes are frequently task specific. By creating a flexible serialized processing scheme we are able to rapidly test and measure structural change in different ways. We are presently continuing to expand this technique, tying it more closely in with

notions of content change.

Now that we have explored Web change on both a content and structural level, we turn to a discussion of ways the models and algorithms we have developed can be used to help people better interact with dynamic Web content.

A.4 Content Motion

As a final example we consider the motion of content within a document. This simple example relies on some of the algorithms we consider above and is again most likely appropriate for content that is changing in a fairly consistent manner. Each DOM element is represented as a node with a weighted edge connecting two DOM elements reflecting the transition of content from one node to another). Our implementation utilizes a simple sparse matrix to represent the complete graph between each DOM element in the document where $G[path_i, path_j]$ is a the edge weight between the node $path_i$ and $path_j$. Our algorithm is therefore:

Algorithm 4 *contentMotion*

1: SORT$(node_hash, version)$
2: $S \leftarrow$ REDUCE$(node_hash)$
3: **for** each s in S **do**
4: foreach sequential pair of tuples (t_i and t_{i+1}), increment $G[t_i.full\_path, t_{i+1}.full_path]$ by 1
5: $G'[p_i, p_j] \leftarrow \frac{G[p_i,p_j]}{\sum_k G[p_i,p_k]}$

G' represents the various transition probabilities and can be used to track the motion of content throughout the document. As a minor modification, we might chose to only consider the maximum, non-self loop transition to diagrammatically label the most likely path of the content on departing the particular DOM node.

Vita

Eytan Adar received his B.Sc. and M.Eng. degrees in Computer Science from MIT in 1997 and 1998, respectively. He worked as a member of the research staff at Xerox PARC from 1998 to 2001 in the Quantitative Content Analysis group and Information Dynamics Lab. During his time at PARC, Eytan helped spin out a personalized search company (Outride) where he was co-founder, and VP of Business Development for a short time. From 2001 to 2005, he worked as a researcher at Hewlett-Packard Laboratories. In 2005, Eytan entered graduate school at the University of Washington in the Computer Science department, working with Daniel Weld. He was awarded an NSF, departmental, and ARCS fellowships to support his research. He received his M.Sc. in 2007 and Ph.D. in 2009.