On the Intrinsic Locality Properties of Web Reference Streams*

BUCS-TR-2002-022

Rodrigo Fonseca[‡] Virgílio Almeida[‡] Mark Crovella[§] Bruno Abrahão[‡]

[‡] Department of Computer Science
Federal University of Minas Gerais
Brazil

[§] Department of Computer Science
Boston University
USA

August 13, 2002

Abstract

There has been considerable work done in the study of Web reference streams: sequences of requests for Web objects. In particular, many studies have looked at the locality properties of such streams, because of the impact of locality on the design and performance of caching and prefetching systems. However, a general framework for understanding *why* reference streams exhibit given locality properties has not yet emerged.

In this work we take a first step in this direction, based on viewing the Web as a set of reference streams that are transformed by Web components (clients, servers, and intermediaries). We propose a graph-based framework for describing this collection of streams and components. We identify three basic stream transformations that occur at nodes of the graph: aggregation, disaggregation and filtering, and we show how these transformations can be used to abstract the effects of different Web components on their associated reference streams. This view allows a structured approach to the analysis of why reference streams show given properties at different points in the Web.

Applying this approach to the study of locality requires good metrics for locality. These metrics must meet three criteria: 1) they must accurately capture temporal locality; 2) they must be independent of trace artifacts such as trace length; and 3) they must not involve manual procedures or model-based assumptions. We describe two metrics meeting these criteria that each capture a different kind of temporal locality in reference streams. The popularity component of temporal locality is captured by entropy, while the correlation component is captured by interreference coefficient of variation. We argue that these metrics are more natural and more useful than previously proposed metrics for temporal locality.

We use this framework to analyze a diverse set of Web reference traces. We find that this framework can shed light on how and why locality properties vary across different locations in the Web topology. For example, we find that filtering and aggregation have opposing effects on the popularity component of the temporal locality, which helps to explain why multilevel caching can be effective in the Web. Furthermore, we find that all transformations tend to diminish the correlation component of temporal locality, which has implications for the utility of different cache replacement policies at different points in the Web.

*Supported in part by NSF grants ANI-9986397 and ANI-0095988; Virgílio Almeida is partially supported by a research grant from CNPq-Brazil.

1 Introduction

The Web is a globally distributed collection of information-oriented services accessible through the Internet. Web protocols follow a client-server model, and so three basic kinds of Web components can be identified: *clients* who request information; *servers* who provide information; and *intermediaries* who generate both requests and replies on behalf of other components. Through the exchange of requests and replies these components are connected and organized into a complex topology. We use the term *Web topology* to refer to the collection of components (clients, servers and intermediaries) as well as the connecting paths through which requests and responses are transmitted. In principle a sequence of requests can be captured from any point in this topology over a given period of time. This work is concerned with understanding the properties of these sequences, which we refer to as "request" or "reference" streams.

Considerable effort has gone into the study of Web reference streams. Ever since the first research on the Web appeared, a major line of study has focused on their properties. This intense focus has been driven by the practical importance of understanding the nature of Web workloads, since it directly affects significant engineering and economic activity. This research has yielded important characterizations and insights, and driven many of the design decisions in the development of the Web. For example, the nature of Web request streams has informed the development of cache replacement policies [40, 12], inter-cache coordination protocols [19], prefetching algorithms [9], and load distribution on server clusters [3]. It has even become possible to contemplate the cost-optimal design of Web caches [27].

Nonetheless, the vast majority of the work so far on Web reference streams has dealt with individual streams *in isolation*, overlooking the fact that the Web is a system of clients, intermediaries, and servers, *each* of which may simultaneously emit or consume Web reference streams. We seek to advance the view of the Web as a system of components emitting and consuming reference streams; we believe that this view will enable a new level of Web characterization and system design.

Thus, the first contribution of our work is to propose and evaluate a shift in perspective—taking a *stream-centric* view of the Web, placing focus on how the characteristics of reference streams change as they pass through the Web. This view of the Web emphasizes three fundamental transformations that streams experience: aggregation, disaggregation, and filtering. We argue that the different components of the Web – clients, servers, intermediaries – can be seen as points where combinations of these basic transformations are applied to streams. Aggregation occurs when streams from multiple sources are merged; a typical example is the interleaving of requests from multiple sources arriving at a server or proxy. Disaggregation occurs when streams are split into disjoint sub-streams; a example is the way in which user requests are directed to different servers or proxies based on the requests' targets. Filtering occurs when a subset of a stream is absorbed and the remainder is passed on, as when a proxy cache serves some requests (hits) and passes on the rest (misses).

To evaluate this perspective, we take some initial steps toward an understanding of how each of these transformations affects the properties of Web request streams that are important for system design. In this work we focus on *temporal locality* as the first and most important of such proper-

ties. The presence of temporal locality is particularly important because of its implications for the success of demand-driven caching.

With the steady increases in Web size and traffic, a complex infrastructure has emerged to improve the Web's scalability and performance. Caches are widely deployed in many points of the Web topology. There are caches located in user browser agents, at business and different institutions, in ISPs, in backbones, and even at specialized points, such as in Web search engines [35]. There are regional and country wide caches, and these are often chained hierarchically. Many companies now offer the services of content delivery networks [28], which is a distributed collection of servers used to increase the scalability of content providers by replicating or caching content closer to users. On the server side, many popular servers use clusters of servers that distribute the load among themselves and appear to the rest of the network as a single server. Configuring and deploying these various infrastructure components involve many design decisions that can be difficult to address.

Typical questions without good answers include: Where should caching proxies be located? How should cache hierarchies be organized (how deep, in what relationship, and with how much local storage)? How does the choice of cache replacement policy depend on whether the cache is near to clients, near to servers, or distant from both? What are good strategies for distributing load in a cluster-based server?

Answers to many of these questions require an understanding of how temporal locality is affected by each kind of stream transformation. To analyze the temporal locality of streams in this setting, we find that the collection of existing metrics for Web reference characterization is insufficient. Previous metrics used to assess temporal locality are tuned to the problem of studying individual agents in isolation, or are better suited to generating representative workloads, instead of explaining what causes the observed characteristics.

Thus, the second contribution of this work is the introduction of a set of metrics that effectively and accurately capture temporal locality, and yet are simple enough to lend themselves to reasoning about the three kinds of stream transformations. These metrics are well adapted to this task because 1) they accurately capture temporal locality; 2) they are independent of trace artifacts such as trace length; and 3) they do not involve manual procedures or model-based assumptions.

We use this framework and these metrics to analyze a diverse set of Web reference traces, and we find that this approach can in fact shed light on how and why locality properties vary across different locations in the Web topology. For example, we find that filtering and aggregation have opposing effects on the popularity component of the temporal locality, which helps to explain why multilevel caching can be effective in the Web. Furthermore, we find that all transformations tend to diminish the correlation component of temporal locality, which has implications for the utility of different cache replacement policies at different points in the Web. These results suggest that our approach can allow a more principled understanding of how components of the Web interact, leading to better insight into the entire Web as a system.

The remainder of the text is organized as follows. In Section 2, we introduce and motivate the metrics we use to capture temporal locality. In Section 3, we describe our general stream transformations framework for the analysis of Web systems. In Section 4 we formalize the definition of temporal locality that we use, and describe its two independent components: popularity and corre-

lation. We then motivate and define the two metrics that we use to assess these two components. They are validated through a set of experiments performed on real Web traces, and with theoretical discussions. In Section 5, we put the pieces all together and analyze a large collection of Web reference traces under each kind of transformation. Finally, Section 6 presents our conclusions and points to future research directions.

2 Background and Related Work

The notion of temporal locality was first recognized by Belady [7] and given a precise definition by Denning [15] in terms of the *working set*, which is the set of unique references contained within some fixed number of past references. These and other early results were in the domain of program memory references, as opposed to Web requests. Throughout this paper we refer simply to *object* references in situations where the distinction is not significant, and also use *virtual time*(following [16]), meaning that time is discrete and advances by one unit with each object reference.

Increased temporal locality generally improves cache performance. A demand-driven cache is designed to store some subset of the objects previously accessed; a good cache has the property that the objects held in the cache have a higher likelihood of being accessed in the near future than would a random collection of objects. When reference streams show temporal locality, then recency of reference is a useful metric upon which to base cache management policies.

Despite the extensive studies of temporal locality, it can sometimes be defined rather loosely, leading to differing approaches in its analysis. Two competing definitions tend to appear:

Definition 1 "An object just referenced has a *high* probability of being referenced in the near future" (*e.g.*, [34]);

Definition 2 "An object just referenced has an *increased* probability of being referenced in the near future" (*e.g.*, [29]).

While Definition 2 would seem to more accurately capture the spirit of temporal locality, Definition 1 better captures the importance of temporal locality for caching systems. For this reason, both definitions appear in the literature and the same term appears applied to both.

To make the distinction more precise, we will use a specific formalization for temporal locality. Consider a reference stream over a set of objects I, with $|I| = N$. We will use $p_i(k)$ to denote the probability that, following a reference to object i, the next reference to object i occurs within k references. We say that a particular object shows temporal locality of reference if there exists a $k > 0$ such that $p_i(k) > 1 - (1 - 1/N)^k$. That is, object i shows temporal locality if there is some distance k over which two references to i are more likely than if references to i were independent with probability $1/N$.[1]

[1]This formalizes the notion of Definition 1; in contrast, Definition 2 can be formalized (as discussed in [29]) as the property of decreasing probability of reference to an object with increasing time since its last reference.

This formalization helps us see that temporal locality can arise in two ways: First, it can exist because an object is simply more popular than its peers, as when $p_i(k) > 1 - (1 - 1/N)^k$ for all k. Second, it can exist when an object's references occur in a correlated manner, as when $p_i(k) > 1 - (1 - 1/N)^k$ for only certain k.

To see this intuitively, we use an example from [25]. Temporal locality arising from popularity can be seen in the reference sequence

$$\ldots XAXBXCXDXEXF\ldots,$$

where the temporal locality comes from the popularity of X; its inter-reference times are much shorter than what would be expected were the objects equally popular. Temporal locality arising from correlation can be seen in the reference sequence

$$\ldots GGHHIIJJKKLL\ldots,$$

in which the temporal locality comes from the nearby occurrence of each letter. Note that in a random permutation of the stream the first property is preserved, whereas the second is not.

Recently, a number of authors have focused on these two ways in which temporal locality can arise. The separation of temporal locality into these two effects was first suggested by Jin and Bestavros [23] and by Mahanti, Eager, and Williamson [29]. Jin and Bestavros termed these two effects *popularity* and *temporal correlations*, while Mahanti, Eager, and Williamson use the term *concentration* instead of popularity; in this paper we consider them to be two kinds of temporal locality which we distinguish as *popularity* and *correlation*.

2.1 Popularity

Probably the best-known characteristic of Web reference streams is their highly skewed popularity distributions, which are usually characterized by the term *Zipf's Law* [21, 14, 1, 11]. Zipf's Law states that the popularity of the n^{th} most popular object is proportional to $1/n$. More generally, "Zipf-like" distributions have been found to approximate many Web reference streams well. In such a distribution:

$$P[O_n] \propto n^{-\alpha} \tag{1}$$

in which $P[O_n]$ is the probability of a reference to the n^{th} most popular object; typically, $\alpha \leq 1$.

The practical implication of Zipf-like distributions for reference streams is that most references are concentrated among a small fraction of all of the objects referenced. Returning to our definition of temporal locality above, this implies that for a small subset of objects, $p_i(k) \gg 1-(1-1/N)^k$ for all k. That is, Zipf's Law results in strong temporal locality because the probability of referencing certain objects is much greater than $1/N$.

Based on the near-ubiquity of Zipf-like distributions in the Web, many authors have used the value of α as a metric for capturing popularity skew [23, 11, 6]. However, for a number of reasons, we take a different approach. First, note that what is important in terms of temporal locality is the deviation from $1/N$ of the probability of referencing some object i (denoted p_i); this deviation is

not directly captured by the Zipf exponent α. Rather, a direct measure of such deviation is available: *entropy.* The entropy of a random variable X taking on N possible values with probabilities p_i is simply:

$$H(X) = -\sum_{i=1}^{N} p_i \log_2 p_i. \tag{2}$$

The properties of $H(X)$ are exactly what we desire: it measures the deviation of X's distribution from the uniform distribution. It takes on its maximum value ($\log_2(N)$) in the case where all realizations of X are equally likely (*i.e.,* $p_i = 1/N,\ i = 1, \ldots, N$.) It takes on its minimum value (zero) in the case where only one observation can occur, that is, when the outcome is deterministic.

The second reason for preferring entropy over the Zipf exponent α is that real data often does not fit a Zipf-like distribution perfectly. As a result, measurement of the Zipf exponent can be subjective, as we will show in Section 4.1. The strength of the entropy metric is that it requires no underlying modeling assumption about the data, and so captures popularity skew equally well whether the trace adheres to a Zipf-like distribution or not.

2.2 Correlation

Attempts to characterize correlation in reference streams focus on the way in which references to a given object are separated by references to other objects. A widely used approach is the *stack distance* model [30]. For any given reference to object i, the corresponding stack distance is the number of unique references since the last reference to i (the first reference to i has undefined stack distance).

The motivation for this metric derives from its relationship to the behavior of a cache managed with an LRU replacement policy. For a reference stream whose corresponding sequence of stack distances is treated as samples of a random variable X, then $P[X > s]$ is exactly the miss rate of that stream when all objects are of equal size, and the stream is applied to an LRU cache capable of holding s objects. This property makes it possible to get performance data for a given stream on all cache sizes simultaneously, with only one pass through the reference stream. The LRU Stack Distance model has been used to model program behavior, and to generate synthetic reference streams with behavior that approximates that of real programs [37]. Similarly, many studies have used stack distance to capture and characterize temporal locality in Web request streams [1, 29, 13, 4] and other authors have used the stack distance transformation as a tool for cache sizing [27] and workload generation [6].

However, stack distance does not fit our needs because it cannot distinguish the causes of temporal locality directly. As already noted, the authors in [29] consider both popularity and correlation, and capture the difference between these two kinds of temporal locality using a modified stack distance model. They propose that stack distance be normalized to factor out the effects of long-term popularity, allowing it to be characterized separately. Cherkasova and Ciardo [13] propose a similar approach based on normalizing stack distance.

The approach we take here is to characterize correlation more directly, using a simpler and more intuitive metric than stack distance. Our approach starts from a simpler metric: the inter-reference distance. Instead of considering *unique* intervening references between two references to object i,

we consider the total number of intervening references including those that appear multiple times. The advantage of this metric is that it does not intermingle the two kinds of temporal locality in the way that stack distance does. To see this, consider a sequence of symbols from a fixed alphabet. If we change the popularity of a symbol by replacing all occurrences of one symbol with another, then the stack distance properties of all symbols are potentially affected — even though the correlation properties of other symbols have not changed. This is not true for the inter-reference distance metric; it is purely a measure of correlation. A side, but important, benefit is that inter-reference distance is much faster and simpler to compute than stack distance.

Inter-reference distance was first used as a measure of temporal locality in [34]. More recently, inter-reference distance was used by Jin and Bestavros [23] as a measure of correlation; our work follows their lead in this regard. However we add to their approach in two ways: first, the measure of request correlation used in [23] is the distribution of inter-reference distance for *equally popular* objects, which may mix together objects that have varying distributions of inter-request times; we separate the inter-request distances for each object. Second, rather than fitting a line to the slope of the distribution of inter-request distances on log-log scale, which is fairly sensitive to noise, we summarize it in a simpler metric: coefficient of variation. This metric captures the essential properties of inter-request distance, as shown in Section 4.2.

After developing these natural and precise metrics for capturing temporal locality, we show how these metrics are affected by the transformations on request streams that commonly occur in the Web: stream aggregation, disaggregation, and filtering. The first two have not been extensively studied before. The third transformation (the stream filtering effects of caches) has recently received some attention. Mahanti, Williamson and Eager [29] study how temporal locality changes at different levels in the caching hierarchy. They show that the concentration of references tends to diminish, and the tail of the Zipf-like distribution increases, as one goes up the caching hierarchy. This effect is also noted and characterized in [41, 18]. The authors in [24] also note that temporal locality properties are likely to be different depending on the location at which they are collected. These papers are consistent with a subset of our results; we put these effects (and others) into a larger framework. For example, while [41] examines filtering effects of caches, it does not separate the effects of filtering on the two kinds of temporal locality, nor does it consider transformations other than filtering. Another related effort is the GD* cache replacement algorithm which tunes itself to whatever popularity and correlation properties are present in the request stream [24].

3 A Framework for Analyzing Web Request Streams

In this section we present a stream centric view of the Web as a strategy to understand the interactions of the millions of individual components – clients, servers and intermediaries – of the Web architecture, and what the performance and design implications of the characteristics of the request streams in different points of the topology are. One of our main goals is to establish a common and general framework for analyzing the Web as a system of agents operating on streams. In the first part of the present section we give an introduction to the architecture of the Web, and then proceed to lay out the abstraction of the Web topology as a collection of nodes that transform

request streams in different ways.

3.1 Web Architecture

A client, according to the HTTP specification [20], is a program that establishes an HTTP connection for the purpose of sending requests. A server, on the other hand, is a program that accepts connections in order to provide responses to these requests. A *user agent* is a client which initiates a request, and an *origin server* is the server responsible for providing the content of a given resource. This distinction of these two special client and server is important because the Web architecture has a third class of components, called intermediaries, that can also act as clients and/or servers, midway between user agents and origin servers. Proxies are the most important type of intermediaries, and are pieces of software that act as both a server and a client, receiving requests from other clients, possibly reissuing these to other servers, and passing the response back to the original client. A wealth of functions may be performed by a proxy, such as anonymization, filtering of unwanted content, logging, provision of external network access, and caching of content. Another example of intermediaries are some types of load balancing devices used by cluster of servers. These clusters, used in heavily loaded Web sites, are a set of machines that work together in answering requests, and appear to the rest of the network as a single server. To divide the load among these backend servers, a type of intermediary may be installed that redirects the connections to the different servers, using different criteria. One of the main reasons for the presence of intermediaries is the maintenance of the scalability of the system as a whole.

Between the user agent and the origin server, a chain of requests and responses may be formed, through one or more intermediaries. For example, the following scenario illustrates several possible intermediaries that may be present in the Web architecture, and illustrates how easily the complexity of the system can grow if we consider the presence of millions of users and servers. In a home office with more than one machine, and one Internet connection, a proxy may be installed on the machine with the external connection, such that the other machines can have access, through this proxy, to Web resources outside the home network. This proxy could be configured to cache previously requested pages, for example, or to prevent requests to particular servers to go through. It could also perform anonymization for the machines inside the private network. This home network, together with other clients of the same Internet Service Provider (ISP), would probably share a communication link to an Internet backbone. The ISP could install a transparent proxy through which the traffic of all its clients has to go through. This proxy could just perform accounting and statistics tasks, but could also have associated a local store with a cache, with the objective of reducing the traffic between the ISP and the backbone. This caching proxy could be part of an hierarchy of caches, such that if a request would not be satisfied by its local cache, it could forward the request to other caches higher in the hierarchy, or directly to the origin server.

In this paper, we use the term Web *topology* to refer to the organization of the collection of components in the Web at a given moment in time—the clients, servers and intermediaries—together with the paths that the request and response messages take between these components. It is important to note that we are specifically dealing with messages transferred through the HTTP protocol, which is an application level protocol. In particular, when we talk about the path between the user

agent and the origin server, we refer to the request-response chain that is formed between pieces of software that generate and receive HTTP messages, that is, the Web components we have just described. We are not concerned, for example, with the routes that different packets that compose the HTTP messages take. The abstraction that the application layer has of the underlying network is simply that any one computer connected to the Internet is accessible from any other one.

Having had a general picture of the Web architecture and its components, how they are organized (and how complex this organization can get), and of how the streams of requests flow between the components, we are ready to discuss our abstraction of this architecture from the point of view of the request streams.

3.2 A Web of Streams

In order to be able to deal with the complexity of the interactions among the different components of the Web architecture, and the varying effect that these have on the characteristics of the request streams, we approach the Web from the point of view of the streams. We abstract the different components of the Web architecture as combinations of basic transformations of the request streams. The advantage of this approach is that we are able to first study these transformations individually and then possibly combined, to assess their effects on relevant properties of the request streams.

The important points of the Web topology to our analysis are those in which the request streams may be altered, together with the paths that connect these points, where the requests flow through. Together, these can be seen as a graph, as we detail below. We are interested in the fundamental transformations that request streams may be subject to, and have initially identified three such transformations:

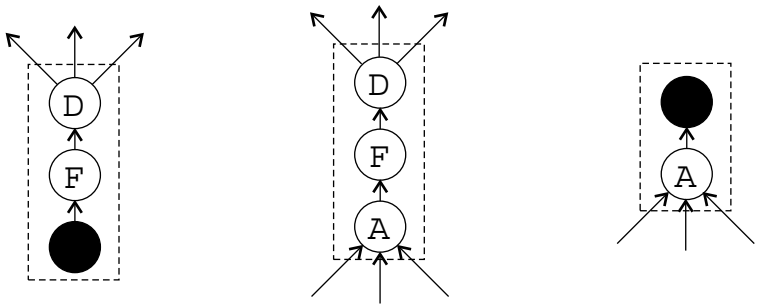
1. Multiple streams may be aggregated, meaning that they are interleaved in time according the arrival times of each request individually. In this case, at least two streams are combined and produce a single resulting stream.

2. A stream may be disaggregated, meaning that it is separated into disjoint sub-streams that follow different paths. The disaggregation is typically (but not always) based on the identities of the objects being requested.

3. A stream may be filtered, in which case some of the requests initially present in the stream are removed, while the remainder of the requests is sent past the filter.

Based on these three transformations, we define three types of nodes in the Web topology graph, Aggregators (A), Disaggregators (D) and Filters (F). We also define two additional endpoint nodes, which can generate and absorb requests; these are part of the user agents and origin servers, respectively. The edges are exactly the connections between these nodes, representing the flow of requests between the successive transformations.

The correspondence between the just defined nodes and the components of the Web topology – clients, servers and intermediaries – may vary, and we view this as a powerful feature of this abstraction. Most commonly, the components will consist of a set of chained transformation nodes.



(a) Client (b) Proxy Cache (c) End Server

Figure 1: Abstractions of typical components of the Web topology in the "ADF" graph. The node types are Aggregator (A), Disaggregator (D), Filter (F), and End Nodes (dark), and correspond to points in the topology at which the Web request streams can be altered.

For example within a client browser there is a stream of requests that is generated by user actions, which is passed to the browser cache. The request stream that is sent into the network originates from the miss stream of this cache. Thus, the browser cache can be represented as having a generator node (the user), followed by an (F) node that alters the original request stream by filtering it (the browser cache). After this, the same request stream is split into different connections, that go to different destinations. This can be abstracted by a disaggregator (D) node. Disaggregation will most commonly happen when requests are sent to different destinations, but may be artificially induced by content. For example, a page served by a content provider which has a contract with a CDN may have requests directed to varying locations. Figure 1(a) shows how a client just described might be represented in our model. The black nodes in the figure are the end nodes.

Another important example of how a component may be abstracted is that of a proxy cache, which is often located at some intermediate point of the topology, between many clients and many servers. When requests are received by a common cache server through different connections, they are aggregated in a single stream, so that the cache subsystem itself sees a single stream. This can be represented by an aggregator node in our graph. This aggregated stream is then processed by the cache, and the miss stream that leaves the cache is the result of a filtering operation by the cache. The next node is a disaggregator node, from which several edges leave, headed towards different servers. Of course, some of these may be other proxy servers, as well as end servers or content providers. Figure 1(b) shows one possible configuration of a proxy cache server in our graph.

A content provider, or an origin server, can also be represented in a similar fashion. Right before the processing by the Web server, which may even be a cluster of servers, there occurs an aggregation operation, and several streams coming upward are multiplexed into a single stream. Of course the details of this aggregation would depend on factors such as load balancing policies, which might be represented by several independent aggregators, or by a sequence of aggregators and disaggregators. A possible representation for a simple server is shown in Figure 1(c).

In this way the whole Web topology can be seen mapped onto this graph. Edges can represent

both the flow of request between different components, such as a client and a server, but also between internal parts of a given component. Furthermore, depending on the level of the abstraction desired, a whole caching hierarchy, for example, could be abstracted by a single filtering node, perhaps with an aggregator node before it, such that different hierarchical organizations could be compared, or, on the other hand, bounds on the gains of the entire hierarchy could be derived, no matter what its configuration, given its input streams.

Our framework for analyzing the Web request streams then consists of the identification of these three basic transformations, and the abstraction of the Web topology by the graph induced by the set of (A), (D), (F), and end nodes, and the paths that connect them. This representation yields a structured way to view the interaction of the different components of the Web, and gives an abstraction with which to study properties of request streams on different points of this topology. The different components are abstracted by sequential combinations of transformation nodes, that have as inputs and outputs one or more streams. Thus, the effects that these components have on the streams are combinations of the effects that the individual nodes have on the streams. To study a given intrinsic property of the request streams, we can then study the effects that each transformation has individually, and then combine these effects to understand what the effects of different components, or of collections of components, are.

As a first application of this abstraction, in Section 5 we analyze the temporal locality properties that request streams present at different points of the topology. Our analysis provides insight into what happens to the temporal locality properries of streams as we move our observation point from the client to the origin server. In order to enable this analysis, in the next section we develop temporal locality metrics that are appropriate for use in this framework.

4 Measuring Temporal Locality

Section 3 presented a view of the Web as a collection of reference streams that are transformed by a set of basic operations. The different components of the Web topology can each be represented by combinations of these basic operations. This view allows us to begin to systematically organize our understanding of how and why different streams show different locality properties. In order to do so, however, we need to define quantitative measures of temporal locality that are suitable for use in this framework.

Because we are analyzing streams abstractly, we need to avoid using metrics that are dependent on the properties of any particular system. This is in contrast to traditional metrics that have been applied to caching systems, (*e.g.,* hit ratio and byte hit ratio [10]) which are strongly dependent on the parameters of the cache itself (such as cache size and replacement policies) and only indirectly reflect the intrinsic properties of the stream.

Furthermore, although we are interested in analyzing these intrinsic properties of the streams, it is only possible to estimate them based on particular finite stream realizations — logs that record request sequences (*e.g.*, client, proxy, or server logs). Therefore we need to pay careful attention to issues of normalization for artifacts such as the particular number of requests or distinct objects in the log.

In the following two sections we define and motivate metrics we use for measuring these two effects; we discuss normalization issues; and we show that the resulting metrics are more general and robust than previously used metrics for temporal locality. The logs we use for some of the validations are described in Section 5.

4.1 Measuring Popularity: Entropy

In this section we define our metric for measuring the skew (degree of imbalance) in the relative popularity of different objects in a request stream, and discuss its strengths.

4.1.1 Definition

We use the entropy of the request stream [36, 38] to measure the degree of imbalance in the popularity of objects. That is, the stream is treated as independent samples of a random variable X, and we are concerned with the entropy of X, denoted $H(X)$.

In estimating $H(X)$ from a given log, we view the requests in the log as a sequence of symbols, which are the requested objects; we approximate the probability p_i of a given object i being referenced as the number of times it appears in the log, divided by the total references in the log. We thus obtain an empirical probability distribution over the set of objects in the log. Then the entropy $H(X)$ is defined as in Equation 2. We note that $H(X)$ only depends on the probabilities of occurrence of the different objects, and not on the relative order in which they occur.

The measure of entropy that we use here is not to be confused with the entropy rate of the process generating the stream, which is a measure of the predictability of a given symbol in the sequence given complete knowledge of all preceding symbols. By treating the subsequent symbols as independent, we are working with an upper bound on the entropy rate of the source. Furthermore, we assume that the universe of the objects is exactly the set of objects that appear in the log, and that is probably not the case.

However, these characteristics are exactly what we desire, for our goal for this metric is to obtain a measure of the popularity component of the temporal locality (which is by definition unaffected by correlations). We make use of the mathematical properties of Equation 2, and do not make direct use of the information theory concepts that are associated with the entropy rate. As an aside, we note that it is an important and challenging problem to estimate the true entropy rate of Web request streams, as that may yield a good predictor for the potential for the use prefetching on the stream [33].

As defined in Equation 2, $H(X)$ also bears a relation to the number of distinct objects that are referenced in the log. In particular, the upper bound on $H(X)$ is given by $H_0(N) = \log_2(N)$, and is attained when each object is equally likely to be referenced. It has been shown that the number of distinct references in a segment of a log increases with the size of the segment, even for very large logs [26]. Thus to be able to compare logs with different number of distinct references present, it is important to normalize the entropy measure. The appropriate normalization is based on the largest possible value of $H(X)$, namely $H_0(N)$. Therefore the metric for popularity we will

use is the normalized entropy:

$$H^n = H(X)/H_0(N) \tag{3}$$

where N is the number of distinct references in the log. The normalized entropy can be regarded as a measure of compressibility of the (scrambled) log. This metric has the same properties as the (non-normalized) entropy, but has an upper bound at 1. The closer it is to this upper bound, the less predictable the stream is, and the more it approaches a uniform distribution over the set of unique objects.

Finally, we note that in some of our later results, instead of working with H^n, we use the following transformation of H^n, which we called "scaled" normalized entropy:

$$H^s = -\log_{10}(1 - H^n) \tag{4}$$

We use H^s because H^n can often be quite close to 1, making it hard to distinguish on plots.

4.1.2 Validation

In this section we show that normalized entropy accurately captures the popularity component of temporal locality by relating normalized entropy to two commonly used measures of this property: hit ratio and Zipf exponent α. As discussed in Section 2 above, α is the most commonly used measure of the degree of imbalance in popularity of Web references. Here we show three key facts:

1. For any given number of unique references N, if reference popularity precisely follows a Zipf-like distribution, then there is a one-to-one relationship between the Zipf exponent α and $H(X)$;

2. In some cases, reference popularity does not precisely follow a Zipf-like distribution, making the estimation of α error-prone, but having no such effect on $H(X)$; and

3. Across a large set of traces and cache sizes, normalized entropy is strongly correlated with hit ratio.

Our first point is that in the case where popularity in fact follows a Zipf-like distribution, for a trace with N unique references, $H(X)$ has a one-to-one relationship with α. This means that, given $H(X)$ or H^n, one can determine the corresponding α — so there is no need to additionally measure it. The fact can be seen as follows: let $S_{\alpha,N} = \sum_{i=1}^{N} i^{-\alpha}$. We call this sum S where it does not impair understanding, to simplify notation.[2] If a trace obeys a Zipf-like distribution, $p_i = \frac{1}{S} i^{-\alpha}$. This comes from Equation 1, with the constant $1/S$ added so that the sum of the probabilities over all i is one. Then:

[2]There is a need to specify N because for $\alpha \leq 1$, this summation does not converge as N tends to infinity.

$$
\begin{aligned}
H_{zipf}^{\alpha,N}(X) &= \sum_{i=1}^{N} -p_i \log_2 p_i \\
&= -\sum_{i=1}^{N} \frac{1}{S} i^{-\alpha} \log_2 \frac{1}{S} i^{-\alpha} \\
&= -\sum_{i=1}^{N} \frac{1}{S} i^{-\alpha} \log_2 \frac{1}{S} + \sum_{i=1}^{N} \frac{1}{S} i^{-\alpha} \alpha \log_2 i \\
&= \log_2 S + \frac{\alpha}{S} \sum_{i=1}^{N} i^{-\alpha} \log_2 i \qquad (5)
\end{aligned}
$$

The relationship between $H(X)$ and α is intuitively clear: as α approaches 0 the popularity distribution becomes more uniform, and the entropy tends to $\log_2 N$. To see this, first note that $\lim_{\alpha \to 0} S_{\alpha,N} = N$, as all the terms tend to 1. From this limit, and from Equation 5, we can derive the limit

$$\lim_{\alpha \to 0} H_{zipf}^{\alpha,N} = \log_2 N.$$

On the other hand, when α grows, the popularity distribution becomes more imbalanced, and in the limit only one object is referenced so the entropy is 0. This can be seen if first take the limit $\lim_{\alpha \to \infty} S_{\alpha,N} = 1$ (all the terms in the sum but the first vanish). From this, and Equation 5, we can see that

$$\lim_{\alpha \to \infty} H_{zipf}^{\alpha,N} = 0.$$

This relationship is illustrated in Figure 2, plots (a) and (b). Most of the reported estimates of α lie in the range between 0.6 and 1 [11]. In this range, the Figure shows that normalized entropy H^n typically lies between approximately 1/2 and 1.

Our next point concerns the lack of generality and accuracy inherent in using the Zipf-like distribution as a model.

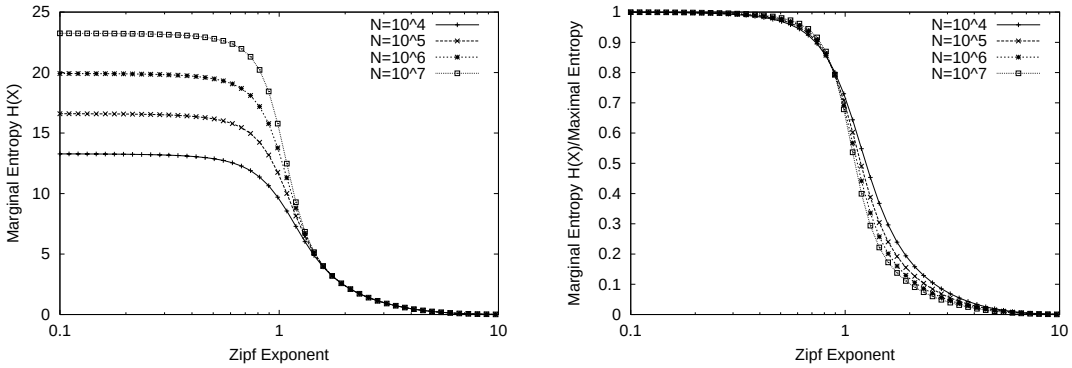
In Figure 3 we show, for two of the traces we studied (others, not shown, are similar), estimates of the Zipf exponent using two different regression techniques (lines marked 'zl' and 'ze'). The lines labeled 'data' plot, on a log-log scale, the number of references to an object versus its popularity rank. The figure shows how reasonable regression procedures can produce widely varying results. The line labeled 'zl' line was determined by a linear fit on the logarithm of the references versus rank data: we fit the logarithm of the popularity versus the logarithm of the number of references by the model

$$\log_{10}(\text{reference count}) = -\alpha \ \log_{10}(\text{rank}) + log_{10}\beta.$$

The 'ze' lines were obtained from a non-linear fit on the original data, using a model of the form

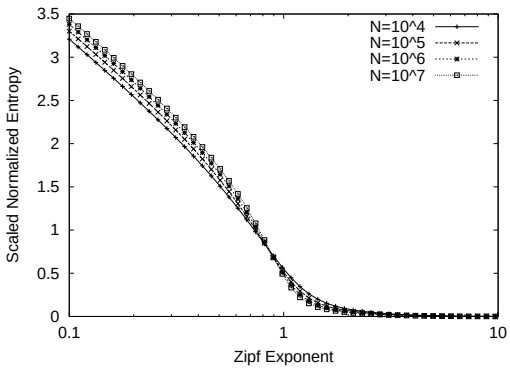
$$\text{reference count} = \beta \ \text{rank}^{-\alpha}.$$

We can see that the linear fit ('zl') emphasizes the points in the tail of the distribution, while the non-linear fit on the original data ('ze') emphasizes the highest ranked objects. For these



(a) Not Normalized: $H(X)$

(b) Normalized (H^n)



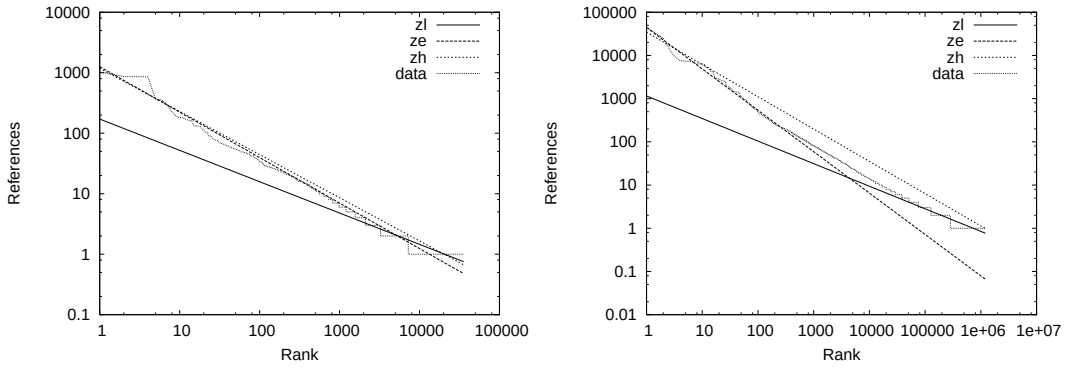
(c) (Scaled) Normalized

Figure 2: Entropy of ideal streams with Zipf-like popularity distribution of references, as a function of the Zipf exponent α.

approaches to α estimation, for example, a larger number of "one-timers" (references occurring only once in a trace) can lead the linear regression to underestimate the exponent, while a single very popular object can lead the non-linear regression to overestimate the exponent. In fact, the two plots of Figure 3 illustrate the impact of one-timers on the estimation of α. The NLANR traces exhibit a much higher percentage of one-timers than the Berkeley logs (see table 4). We notice from Figure 3(b) that the linear fit underestimates α for the NLANR trace. It is a matter of judgment to determine which points to use and which procedure to adopt, in order to correctly estimate the exponent.

In fact, a third estimate of α is possible: that based on inverting Equation (5). By first calculating $H(X)$, and then obtaining the corresponding α value, we obtain estimates of α that appear to be even *better* fits to the actual popularity distribution; these lines are marked 'zh' in Figure 3. To obtain this value, we first approximated the sums in Equation 5 by integrals, which can be simplified

to a closed form formula. We then used the software Derive [17], which is capable of performing approximate analytical calculations, to solve the resulting equation for α, given the values of H and N. In Appendix A we show our approximation in more detail. From these observations, we argue that $H(X)$ is a fundamentally more robust metric than is α.



(a) Berkeley Server Log

(b) NLANR SD Log

Figure 3: Determination of the Zipf parameter using the entropy ('zh' curve) and two standard regression techniques for the NLANR SD and Berkeley Server logs.

Cache size (%)	Correlation of H^n and HR	
	Scrambled Logs	Original Logs
1	-0.72	-0.61
2	-0.83	-0.77
5	-0.84	-0.86
10	-0.81	-0.84
14	-0.79	-0.81
20	-0.78	-0.79
29	-0.76	-0.76
50	-0.74	-0.74

Table 1: Correlation coefficient between H^n and hit ratio.

Our third and last point is that H^n has a strong effect on hit ratio which suggests that it provides a useful view of locality. To demonstrate this we show the results of cache simulations. We simulated an LRU cache and did not consider the size of the objects referenced, *i.e.*, we considered all objects to be the same size. While this is not an accurate measure of true hit ratio considering object sizes, it can provide valuable insight into the utility of the entropy metric.

Figure 4 shows the results when caches are sized to hold 5% of the number of unique references in each log. Logs were scrambled to remove all temporal correlations, leaving only the popularity component of the locality. The figure shows the clear and strong trend of decreasing hit ratio with increasing entropy. A wider range of results are summarized in Table 1. This table shows, for a

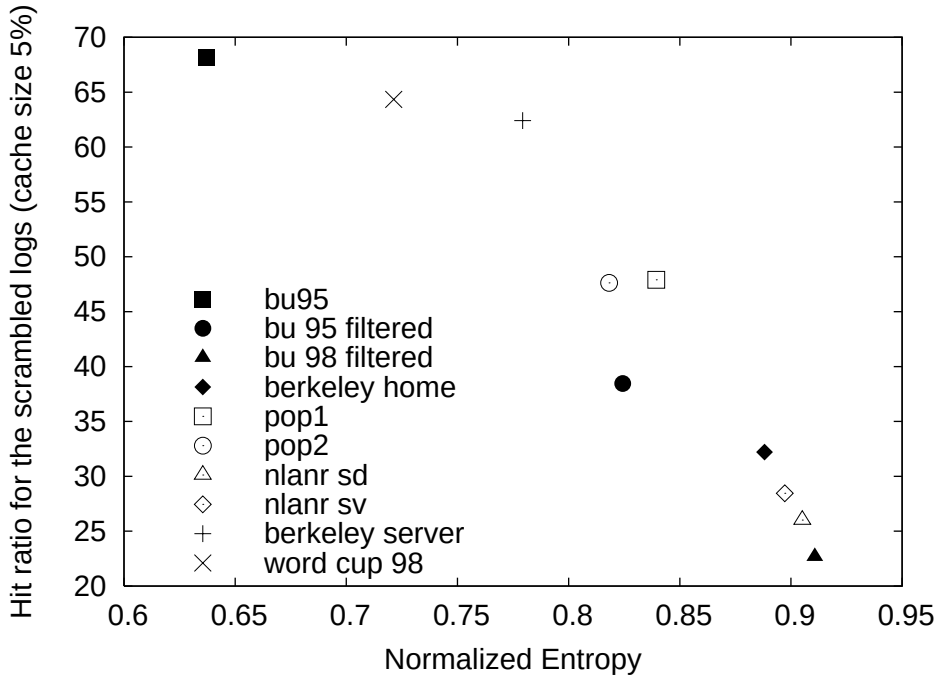


Figure 4: Hit ratio for the scrambled versions of the logs versus the normalized entropy of the logs. The cache uses LRU replacement policy, and has the capacity to hold 5% of the total objects in each log.

range of cache sizes, the correlation coefficient (R^2) relating the normalized entropy to the hit ratio obtained in the caching simulation for that size. The table shows that the correlation between hit ratio and normalized entropy is strong across a wide range of cache sizes.

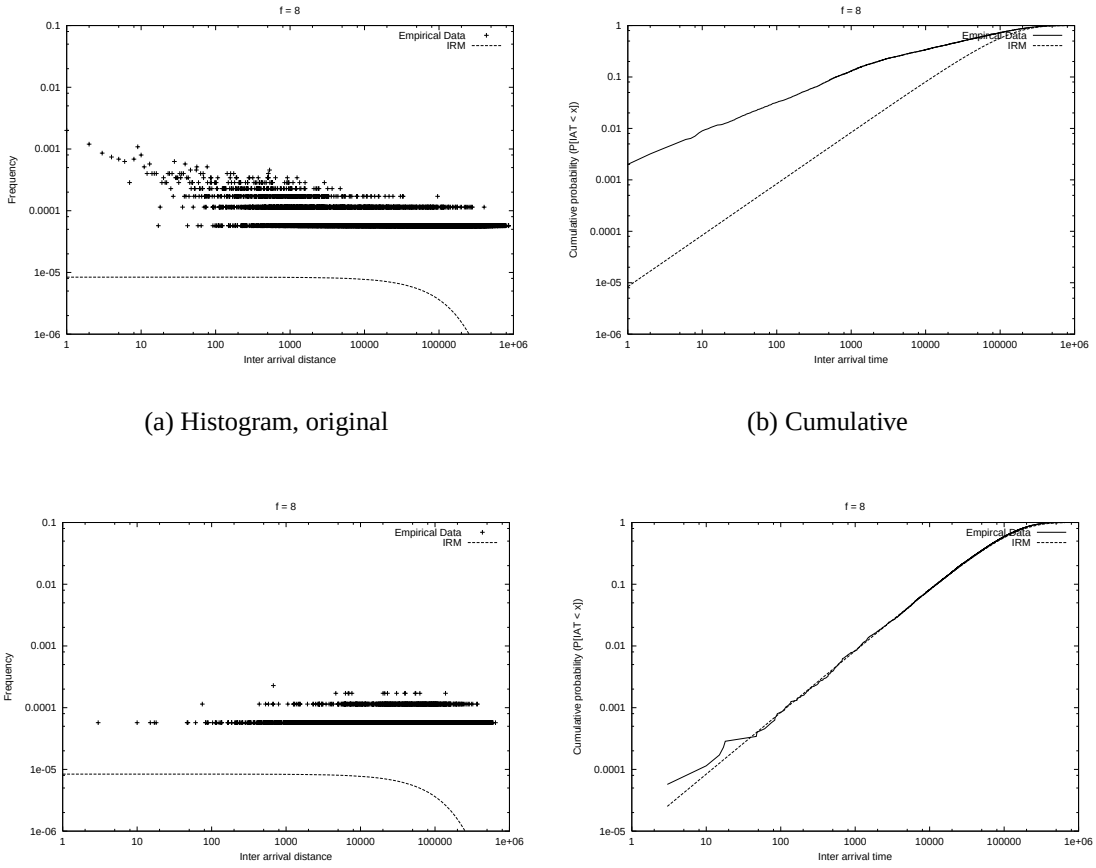
In fact, this result is supported by theory in [33], which establishes that entropy can be used to establish bounds on the miss rate of an optimal caching algorithm for a discrete memoryless source with finite alphabet. It is important to note however that they use the entropy rate of the source of the request stream, and that the measure of entropy that we use here is a first order approximation of the source entropy rate. For details, we refer the reader to [33, 36].

Finally, Table 1 also shows results for the original, non-scrambled versions of logs. It is interesting to note that entropy bears a strong (negative) correlation with the hit ratio for the original logs, especially for larger cache sizes. This may be understood because the larger the cache, the less important the impact of temporal correlation is on the cache performance.

4.2 Measuring Correlation: Coefficient of Variation

Having identified an appropriate metric for capturing popularity, in this section we describe our metric for the correlation component of temporal locality, and show its utility.

If all accesses to objects in a stream were completely uncorrelated, we could model the request generating process by the Independent Reference Model [16] (IRM). In the IRM, each object



(a) Histogram, original

(b) Cumulative

(c) Histogram, scrambled

(d) Cumulative

Figure 5: IAT distribution for documents of frequency 8 from the NLANR sv log.

has an associated probability of being referenced, and each reference is independent of any other reference. In this model the inter arrival time (IAT) for a given object (measured in number of references) follows a geometric distribution, which is a discrete memoryless distribution. For each object i with reference probability p_i, the mean of its associated IAT distribution is given by $1/p_i$.

If we form a random permutation of a log of references, thus removing all temporal correlation, we expect the IAT of each object to follow a geometric distribution, with mean determined by the relative frequency of appearance of the object. In the original log, however, if there is temporal correlation between the references to the same object, we expect to see deviation from this memoryless behavior.

As an initial example, to distinguish the presence of correlation from that of popularity, we proceed as the authors in [23], grouping equally popular objects and conducting separate analysis of these groups. In the absence of temporal correlation, we would have the IRM, as discussed above, and the distribution of the IAT's for any given object would depend solely on the probability

of accesses to that object. Thus, equally popular objects would have the same IAT distribution.

We examined the IAT distribution for groups of objects with the same number of accesses, and obtained similar results for all such groups. As an example, we show in Figure 5 the IAT distribution for one of these groups, that of all objects with 8 references each.

We plot the distribution and the cumulative distribution of the IAT's for the objects for the original log (plots (a) and (b)), and for a scrambled version of the same trace (plots (c) and (d)). Figure 5(a) shows, in the histogram for the original trace (a), a strong tendency of the IAT's to cluster around shorter values, and this tendency is not present in the same plot for the scrambled trace (c). This is confirmed in the cumulative distributions (b) and (d). Also plotted are curves labeled 'IRM', which represent the geometric distribution with the parameter adjusted to represent the same probability of occurrence as the objects considered. Notice that the actual distribution, in plots (a) and (c), does not fully approach the expected behavior because of the finite length of the trace: the lowest values of these curves correspond to one occurrence of that IAT divided by the total number of IAT's recorded, and is not as low as the predicted IRM values. In plots (b) and (d) we see, however, how close the scrambled version approaches the curve for the geometric distribution, whereas the curve for the original plot has a large increase in the lower values. The plots for all other frequencies show very similar behavior, and these results confirm those obtained (in a slightly different fashion) in [23]. These results present evidence that the temporal correlation indeed has an effect on the locality, since in the original, non-scrambled traces, the IAT's tend to be lower than what would be predicted by the IRM.

4.2.1 Definition

Motivated by the fact that temporal correlation for the accesses to the same object should cause a deviation from the geometric distribution, we introduce a metric that is very sensitive to this deviation. The *coefficient of variation* of a distribution is its standard deviation σ divided by its mean μ, as shown in Equation 6. Coefficient of variation (CV) is widely used in different settings, and is a simple measure of relative dispersion of a distribution. This metric is dimensionless and is simple to calculate and understand.

$$cv = \frac{\sigma}{\mu} \tag{6}$$

CV is also a convenient measure to use here because it has a natural reference point for the geometric distribution. Given a geometric distribution with mean given by $\mu = \frac{1}{p}$ and variance given by $\sigma^2 = (1-p)/p^2$, the CV is given simply by $\sqrt{1-p}$. In Web reference streams, in which even the most heavily accessed objects have a very low probability of reference (generally much less than 1%), the expected CV, in the case of no temporal correlation, is very close to 1. Thus, in this setting, CV values close to unity are associated with distributions close to the IRM, and thus little or no temporal correlation, while values larger than one represent a distribution with large relative variance.

In order to use the IAT-CV as a measure of correlation, we must decide how to apply it across the unique set of references in a trace. In [23] the authors form IAT distributions over the set of all objects with k references. This has the effect of mixing together objects with possibly varying

properties. In contrast we wish to consider each object separately and so form the IAT-CV for each unique reference in the trace. Furthermore, unlike [23], using IAT-CV as a summary of the distribution in this way does not require any assumption about distributional shape nor does it require any manual curve-fitting.

Each unique object in a trace has its own IAT-CV. Again, for each object we determine the distribution of its IATs. We consider that the trace repeats itself indefinitely, that is, this assumption has the effect of 'wrapping' the stream. This way, for each object that appears k times, with $k \geq 2$, we add another IAT value to the distribution, one that adds the number of references from the last reference of the object to the end of the trace and from the beginning of the trace to the first reference of the object. This is done to avoid some artifacts due to the arbitrary size of the log, since we are ultimately interested in characteristics of the stream, and not of the log, a finite realization of the stream itself. In order to summarize an entire trace we must combine these individual values. The method we use is based on the *per-reference* IAT-CV. That is, since we are concerned with temporal locality as it impacts caching systems, we seek a metric that is weighted on a per-reference basis (rather than, say, a per-unique-reference basis). Another reason for this choice is that the CV inherits from the mean and the standard deviation the characteristic that the confidence on the estimate decreases with the number of samples. The smaller the number of accesses an object has in the log, the smaller will the confidence be on its estimated CV value. By weighting the CV of each object with its number of references, we are implicitly assigning more relative importance to the measures of CV in which we can have more confidence. As a final remark, in Appendix B we derive an upper limit on the CV computed from n samples (in our case, this applies to the objects that appear n times) as being $\sqrt{n}$. This fact could cause a dependence of the CV on the number of accesses of an object, but we observed that the values for the CV of the different objects are seldom close to this upper limit, specially for objects that are more frequently referenced, and (ii) the weighting per-reference diminished this effect.

Interestingly, the distribution of per-reference IAT-CV shows a long tail, making the mean of this quantity an unstable metric. We find experimentally that the long tail of this distribution results in a mean IAT-CV that grows with the length of the trace considered. Since we want to normalize our metrics to be independent of trace length, we need a statistic that is insensitive to trace length. For this reason we use the distributional median, which is a very robust metric even for long-tailed distributions.

To sum up, the metric for correlation we use is calculated as follows. For each object in the trace, calculate its IAT-CV (the standard deviation of its IAT over its mean IAT). Now form the set consisting of the IAT-CV for each individual reference in the trace; an object with n references in the trace will have n copies of its IAT-CV in this set. The final metric is the median value of this set.

4.2.2 Validation

The presence of correlations in a reference stream can be measured by its effect on cache hit ratio. When a trace is scrambled, the resulting hit ratio tends to decrease due to the removal of correlation. To see this, consider the example streams in Section 2. If we scramble the second

trace, the correlation disappears.

Figure 6 shows this effect for four traces. In each case we plot the curve for the original trace and its scrambled version, which has all correlation removed. The curves represent the hit ratio as a function of cache size, and show that different traces incur varying changes in hit ratio when scrambled. The strongest variation happens to the "bu95" trace, while the "world cup 98" trace presents the weakest correlation. In all cases, to varying degrees, the effect of the temporal correlation is to improve hit ratio, as all original traces showed higher hit ratios than the corresponding scrambled traces, and these effects are magnified for small cache sizes.

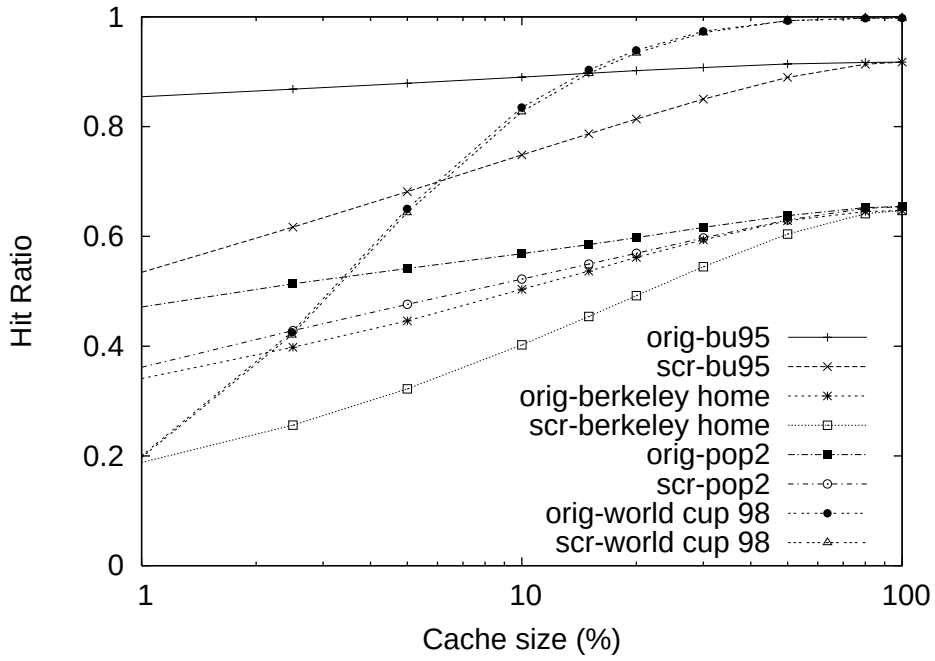


Figure 6: Hit ratio versus cache size, for the original and scrambled versions of some of the logs studied. A large difference indicates that there is a component of the temporal locality that depends on temporal correlations between the accesses.

This difference may be used to gauge the utility of our IAT-CV metric. In Figure 7 we show the relationship between the difference in hit ratio due to scrambling, and the IAT-CV, for all of the logs we studied. In this case all caches were sized to hold 5% of the unique references in each trace. The difference, for each log, is exactly the hit ratio obtained in the cache simulation for the original trace minus the hit ratio for the same trace scrambled. The figure shows the clear relationship between improvement in hit ratio and IAT-CV. Larger IAT-CVs are indicative of stronger contributions of correlation to the overall hit ratio of the trace in a cache.

These results are generalized to a range of cache sizes in Table 2. The table shows that even for large cache sizes (capable of holding half of the unique references in a trace), that the improvement in hit ratio is correlated with IAT-CV; and for small caches, this correlation is quite strong.

Cache size (%)	Correlation CV X HR diff.
1	0.78
2	0.72
5	0.65
10	0.53
14	0.44
20	0.38
29	0.31
50	0.32

Table 2: Correlation coefficient between CV and the HR difference (original - scrambled), for different cache sizes.

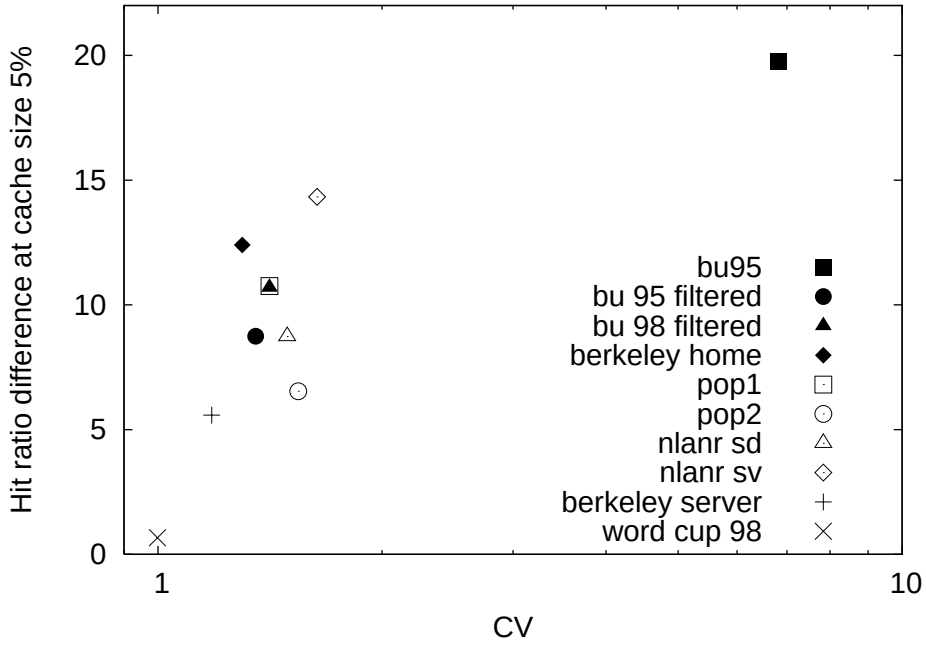


Figure 7: Difference in hit ratio between the original and the scrambled logs, for an LRU cache with relative size of 5%.

5 Effects of the Transformations on Request Streams

In this section we show how locality is affected by the transformations on request streams that commonly occur in the Web: stream aggregation, disaggregation, and filtering. We proceed through a set of experiments on logs taken from different points of the topology, using the metrics introduced in Section 4. We begin with a description of the logs we used, and present the results of the experiments used to assess the changes in the locality properties induced by the individual transformations.

5.1 Description of the Logs used

We have tried to obtain traces from varying points in the Web topology, from close to the client to deep intermediary nodes, to the origin servers. We notice that it is increasingly difficult to obtain suitable traces. Companies are often reluctant in making their access logs available, as potentially valuable information could be inferred from them, such as the sheer popularity of their Web sites to the volume of sales through some e-commerce portal. On the side of the client, it is difficult to obtain the complete stream of requests generated by a large population of clients, because that requires that the requests be recorded before the users' browser caches. We only obtained one log in this fashion. Other approaches have been used to collect client traces, such as the use of cache-busting proxies, that serve all the content to the clients as being uncacheable. This approach has been used in [26], for example.

We broadly divide the logs we used into 3 groups, according to their location in the topology of the Web: client, proxy, and server logs, and this distinction is useful in some of the discussions that follow. The client logs present a subdivision: the BU95 trace is the only one collected before browser caches, while the other client traces are actually collected in the first proxy after the streams have left the client machines. This explains some of the differences that arise when analyzing its temporal locality properties. In the paragraphs that follow, we briefly discuss the logs, ordered by their proximity to the clients. Most of these can be obtained from the Internet Traffic Archive (ITA) [22], or from the W3C Web Characterization Repository (WCA) [39].

The **bu95** trace we use is part of a trace collected and made available by the Computer Science Department of the Boston University. The trace and the data collection process are thoroughly described in [14]. This trace was collected from instrumented versions of the Mosaic browser at the computing facilities of the CS Department of the Boston University. It is the only one that we had available that was collected before the browser caches, thus registering all the requests that the users generated. We have reasons to believe that this is the main cause of this log presenting temporal locality properties very different from the other traces. We used a portion of two months from this log, starting on January 1st, 1995, for a total of 558,263 requests. The 'bu95' trace is the only true Client trace that we analyzed.

From the same trace, we generated a second one, which we call **bu95flt**, or bu95 filtered. There is information in the trace to inform when a request was served from the browser's internal cache, and when it was fetched from the network. The **bu95flt** trace is equivalent to the miss stream of the **bu95** trace, i.e., it presents only those requests that were *not* served by the browser's local cache. We consider this trace to be part of the 'Client Proxy' group, as it might have been collected from a hypothetical proxy serving all of the client machines used in the data collection.

The **bu98flt** trace we use is derived from a trace collected 1998 in the Computer Science Department of the Boston University. It is described in [5], and records accesses from 306 users on 29 machines. The data is collected by a set of HTTP proxies external to the client browsers, located in the client machines, and is later aggregated to form a single trace, and we also consider it to be a 'Client Proxy' trace. The trace we used spans approximately one and a half months, for a total of 67,629 requests.

The last trace from the 'Client Proxy' group that we analyzed is the **bk-home-ip** trace, derived

from data collected from a gateway computer through which all the traffic from the clients connected to the HomeIP service of the University of California ad Berkeley passed. These are home users connected via dial-up connections, so that the data collected corresponds to all of their requests, except for those that were satisfied by the browser caches. The data was collected by means of a packet sniffer that only recorded HTTP data connections on port 80. We used a period of 3 days, starting November, 6th, 1996, and comprising 1.7 million requests. It can be downloaded from the ITA [22].

We now describe the logs which we call 'Network Proxy' logs. These are collected from proxies that are more "deeply" located in the Network. The logs we call **POP1** and **POP2** are the proxy logs for the caches located at the POP-MG, an ISP located at the Universidade Federal de Minas Gerais, that provides Internet access to some businesses and universities in the state of Minas Gerais, Brazil. The caches are organized in a two level hierarchy. The **POP1** (POP-MG level 1) trace is collected at two machines that serve requests coming from end users of client organizations that do not have their own caches. The **POP2** (POP-MG level 2) cache only answers requests from the level 1 caches, which include the **POP1** machines. We used two days of data for both caches, October 18th and 19th, 2001, resulting respectively in 1,949,490 and 2,308,411 requests. We removed requests that returned error codes (4xx and 5xx), and also those that did not use the method GET and the HTTP protocol.

The second set of Network Proxy logs we used is from the NLANR cache hierarchy, which is a set of top level caches that are part of the IRCache project [31]. Currently, logs from the last seven days are available upon request. They are free of charge for academic use. The caches answer requests from other caches only, except for the sd server, to which end users can connect. We used 1 week of logs from two of the NLANR caches, namely the sd and sv caches, starting on November 13th, 2001, for respective totals of 3,950,198 and 5,357,077 requests. We refer to these logs as **NLANR sd** and **NLANR sv**, and have applied the same cleaning procedure used for the **POP1** and **POP2** logs.

Finally, for the server logs, we analyzed logs from two sources. The first log, **bk-server**, records requests for the the University of California at Berkeley Computer Science Department Web Server, and can be obtained from [8]. We used the logs corresponding to the month of December, 2001, which correspond to a total of 6,011,564 requests.

The last trace we use in our experiments is part of the data collected from the 1998 FIFA World Cup Web servers, and can be obtained from the ITA. We refer to it as the **wc98** trace, and it corresponds to all of the accesses to the World Cup servers on May 29th, 1998, comprising a total of 2,223,141 requests. This log is thoroughly analyzed in [2].

A short description of these logs is given in Tables 3 and 4. All of these, except for the POP1 and POP2 traces, are publicly available and can be found in [22] or [39], unless otherwise noted.

We also utilize scrambled versions of the same logs, that are similar to the original logs, but with have correlations removed. These scrambled logs allow us to separate the effects of the popularity and of correlation.

Name	Short Description	Period
Client, before browser cache		
bu95	'95 Boston University	01/01 - 02/28/95
Client Proxy, after browser cache		
bu95flt	'95 Boston University	01/01 - 02/28/95
bu98ftl	'98 Boston University	04/06 - 05/21/98
bk-homeIP	Berkeley Home IP Service	11/06 - 11/09/96
Network Proxy		
pop-1	POP-MG Level 1 Cache	10/18 - 10/19/01
pop-2	POP-MG Level 2 Cache	10/18 - 10/19/01
nlanr-sd	NLANR SD Root Cache	11/13 - 11/19/01
nlanr-sv	NLANR SV Root Cache	11/13 - 11/19/01
End Server		
bk-server	Berkeley CS Dept.	12/01 - 12/31/01
wc	'98 World Cup Web Site	05/29/98

Table 3: High level description of the logs used

5.2 Filtering

Using these traces, we start by assessing the effects of filtering on reference streams. Because the goal of the experiment is not to analyze replacement policies, but to understand the effects of caching on reference streams, the simulations assume a simple LRU policy.

When analyzing the filtering transformation, we want to understand how entropy and CV capture the temporal locality properties of the output stream, and what is the relation to these properties to those of the input stream. As an example, the expected behavior of a cache is to emit a miss stream with increased, or even maximized, entropy as compared to the inputs stream (since the outcome of an ideal cache is to only leave one reference to each unique object). Regarding the temporal correlation, the requests that are present in the miss stream may have different causes:

Name	Requests	Objects	% 1-Timers
bu95	558,263	48,532	43.64
bu95flt	128,077	47,502	72.14
bu98flt	67,629	35,646	42.02
bk-homeIP	1,703,835	600,940	66.18
pop-1	1,949,490	464,795	69.53
pop-2	2,308,411	734,015	76.26
nlanr-sd	3,950,198	1,785,884	18.17
nlanr-sv	5,357,077	1,544,956	35.38
bk-server	6,011,564	268,018	1.69
wc	2,223,141	4,829	0.04

Table 4: Some important statistics of the logs used

(i) the first reference to each object must generate a miss, for the cache could not have a copy of an object it has not seen; (ii) capacity misses are generated when the cache replacement algorithm chooses to substitute an object in the cache for a new object; and (iii) other issues such as expiration or invalidation may also generate misses. None of these processes are expected to bear much correlation to the process(es) that generate the requests that arrive to the cache, and therefore, should work to decrease the temporal correlation between requests.

The two following questions help us to understand the effects of this transformation: 1) how do the locality metrics of the output stream vary as a function of the cache size? and 2) how do the filtering effects vary according to the topological position of the caching in the Web?

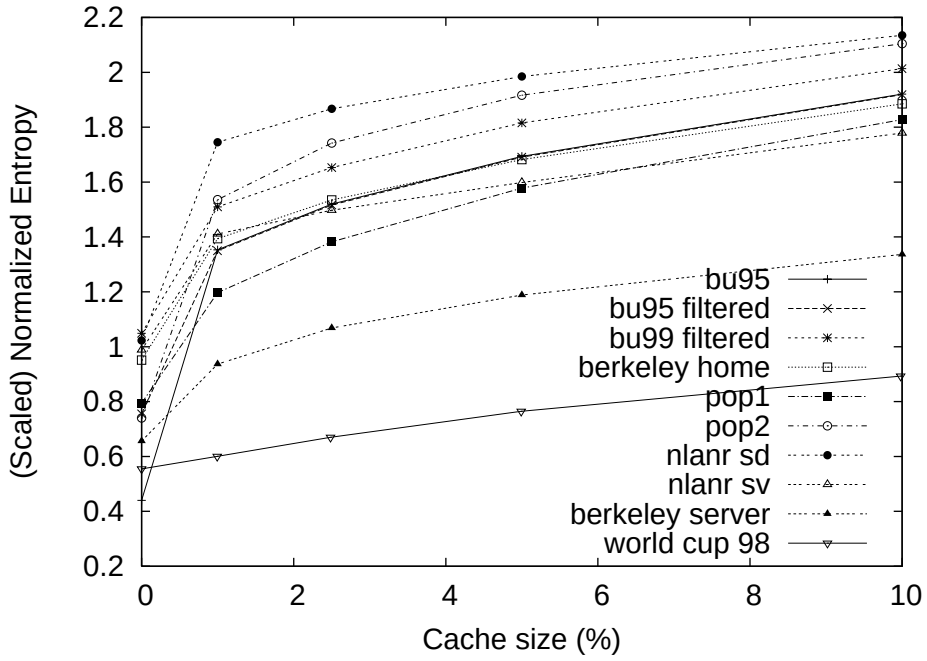


Figure 8: (Scaled) normalized entropy of the miss stream versus cache size. Cache size is measured as a fraction of the total number of objects in the log. The point for cache size of 0 corresponds to the original log.

Intuitively, we expect that filtering absorbs part of the temporal locality of a reference stream and generates a miss stream consisting of evenly distributed references to fairly popular objects. The graph of Figure 8 shows the variation of the normalized entropy as a function of the cache size. In the graph, note that points corresponding to cache size equals to 0 refer to the entropy of the input stream. The figure shows that the entropy of the output stream (i.e., miss stream) increases with the cache size, indicating that more popular references are eliminated from the stream as we increase the size of the cache, making the object distribution of the output stream more uniform.

The figure also shows that the deeper the cache in the Web system, the lower the magnitude of the entropy difference between input and output streams. The explanation for that stems from the different percentage of one-timers in the reference streams. As indicated in Table IV, one-timers

in the two traces near to clients constitute 42% and 66% of references. In the World Cup and Berkeley servers logs, the one- timers represent 0.04% and 1.69%. In a popularity versus rank plot, the curves for the server logs would have a concentration of points at the top left portion of the curve while the client logs would have a concentration of points at the tail of the curve. Williamson [41] shows graphically that the primary impact of filtering is to truncate and flatten the top left portion of popularity curves. Therefore, entropy variation is magnified by the presence of a large percentage of one-timers. Williamson [41] also shows that deeper levels of caching produce little change in the object popularity profile, which agrees with the results shown here.

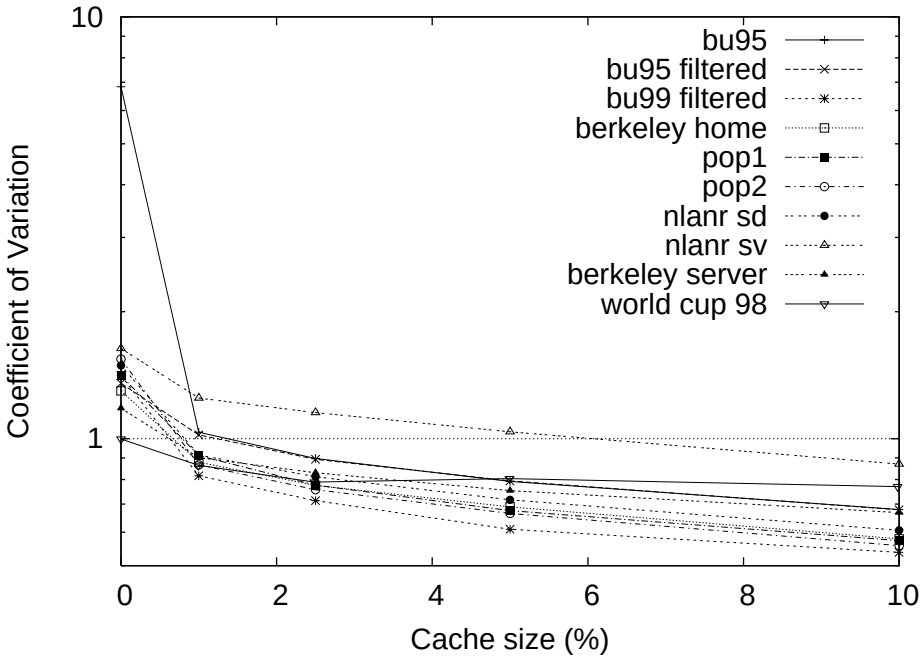


Figure 9: CV of the miss stream versus cache size.

Figure 9 displays the IAT-CV as a function of cache size. As the cache size increases, more repetitions are eliminated from the output stream, decreasing the temporal correlation of the miss stream. As a matter of fact, the miss stream should exhibit a low coefficient of variation, for the temporal correlations are almost eliminated by the LRU policy of the caching. For larger cache sizes, since we use a cache simulator that only has capacity misses, and no expiration misses, the number of requests in the miss streams begins to diminish quickly, and so does the number of repetitions. This makes the number of objects that actually contribute to the value of IAT-CV to also diminish, and the metric loses precision for these larger caches.

In summary, we find that the effects of caching are to remove both sources of temporal locality from reference streams. This suggests that the output stream from a cache would typically be a poor candidate for sending to another cache, since the subsequent “downstream” cache would observe little temporal locality in its input stream. Nonetheless, we observe that multilevel caching is common in the Web, from client caches to proxy caches to server accelerators (caches in front of

servers). Why are such multilevel caching schemes effective? The answer is provided by turning again to the ADF framework, which we do in the next section.

5.3 Aggregation and Disaggregation

In order to analyze the effects of the aggregation and disaggregation transformations on the streams, we analyze different logs by separating them into sub-logs corresponding, in turn, to the sources of the requests, and to the destinations of the requests. We then study the distribution of H^n and IAT-CV across these sub-logs. Note that we do not use any filtering between the two transformations, for we want to isolate their individual effects. The plots in Figure 10 show the cumulative distribution *per reference*, i.e., weighted by the number of requests in each sub-log, of H^n and the CV, for a sample log. The full vertical lines in each plot shows the value of the corresponding metric in the full log.

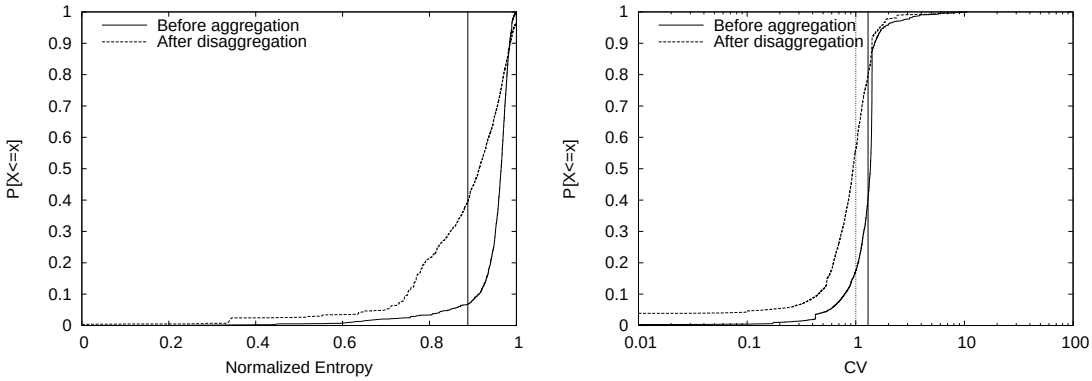


Figure 10: Cumulative distributions of H^n and CV for the Berkeley Home IP log, for the aggregation and disaggregation sub-logs. The full vertical line is the respective value for the full log.

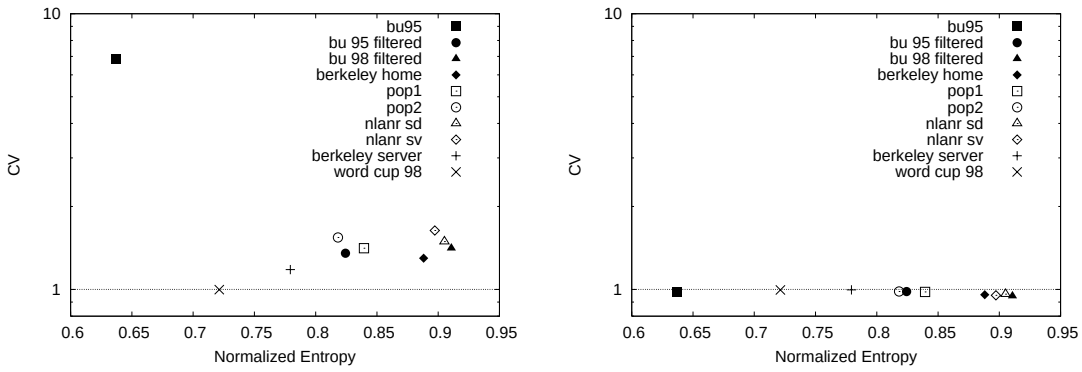
We first examine the results for normalized entropy. Across all logs we studied, the percentage of requests from incoming sub-logs with H^n greater than that of the full log is consistently above 50%. This shows that aggregation tends to *decrease* the normalized entropy of its output stream compared to its input. That is, aggregation acts to increase the popularity component of temporal locality. This provides an answer to the question posed at the end of the last section: higher level caches tend to be effective despite lower level caching because they generally involve considerable stream aggregation (as shown in Figure 1(b)). This is an example of how we can use the ADF framework to enable better engineering of the Web system, since we can decide to place caches at locations in which high levels of aggregation occur.

For the case of the disaggregation, although the median of the distribution is close to the reference value for the full log, one can notice a significant increase in the number of requests belonging to outgoing sub-streams with lower normalized entropy. This is also intuitive, because the streams that servers receive have a more limited set of objects, and the requests can be drawn from several different incoming streams, the repetitions of which were not filtered together before.

Concerning the IAT-CV, we have found for all logs we studied that the distribution per reference of the IAT-CV decreases when comparing the incoming sub-streams for the aggregation to the outgoing sub-streams for the disaggregation. This can be seen for the second plot in Figure 10. This finding is in line with the intuitive notion that the temporal correlations should decrease when aggregating, because the separate processes that generate the requests (for example, two different users) are most likely independent time series. The same observation applies to the disaggregation transformation.

Thus we find that aggregation and disaggregation shed considerable light on the Web system: both aggregation and disaggregation tend to increase the popularity component of temporal locality, while tending to somewhat decrease the correlation component of temporal locality. These conclusions are supported by looking at the properties of traces at different levels in the hierarchy, which we do in the next section.

5.4 Locality Properties at Different Points of the Topology



(a) Original Logs

(b) Scrambled Logs

Figure 11: CV versus Normalized Entropy for logs studied. Logs from locations close to clients are shown with filled symbols, logs from proxy servers are shown with hollow symbols, and the logs from servers are shown with line drawn symbols.

Having studied the effects of the transformations individually, we now present, in Figure 11, results combining both metrics for each of the ten logs we studied, which come from different points in the topology. In the figure, plot (a) shows the results for the original logs, and plot (b) shows the results for the scrambled logs. The first observation is that the CV of the scrambled logs indeed tends to the value predicted by the IRM, *i.e.*, it is very close close to 1 for all logs.

We can readily notice three different groups of points. The first is that composed of one point, **bu95**. This is the only trace that is collected before browser caches, and has not been filtered in any way. It presents a high degree of temporal correlation, that comes from the correlations in user surfing patterns, and also the lowest relative entropy, since no repetitions have been filtered out

yet. Then we move to the 'proxy region', in which the streams present high H^n and a much lower CV. These characteristics come from the fact that these proxies generally receive streams that have been filtered by lower level caches. It is interesting to notice that even proxies that are close to clients exhibit these characteristics. The third region in the plot is the 'servers' region. These present the lowest temporal correlation, and also lower entropies than the proxies. This is in line with our findings that aggregation and disaggregation both increase popularity imbalance and that all transformations tend to decrease temporal correlation. It has also been verified (for example, in [32]) that the Zipf exponent for the popularity in servers is generally much higher than in proxies. This is strongly in line with the fact that the entropy of the server logs that we examined is lower than that of the proxies.

Other interesting facts can be noted from the figure. The point for **bu95** may seem to be an outlier, but confidence on the validity of the numbers comes from the fact that the **bu95flt** point shows similar characteristics to those of other proxy logs. Again, this log is obtained directly from the **bu95** log, by removing the objects that were cached locally in the browsers. Also, the relation between **bu95flt** and **bu98flt** confirms previous results. These logs are compared in [5], and it is found that the popularity distribution of objects in the 1998 logs is less concentrated than it is in the 1995 trace. In our plot, although they exhibit similar IAT-CV, there is a considerable difference in the entropy, that confirms this observation. The authors conjecture that this is due to an improvement in the client caches. That makes sense, since, as we have seen, better caches further increase the entropy of the miss streams.

These findings show that the correlation and popularity components of locality behave differently as streams pass through the Web system; while popularity imbalance can rise or fall as a result of stream transformations, correlation seems to be intrinsically generated by clients and generally declines as streams are transformed. The previous analysis of the individual effects of the basic transformations on the streams proved valuable in providing insights for the observed results.

6 Conclusions and Future Work

In this work we have presented a shift in perspective for Web characterization, by proposing a stream centric view of its architecture. We identify three fundamental transformations to which streams are subject—aggregation, disaggregation and filtering—and abstract the components of the Web architecture as combinations of these basic transformations. This allows us to gain a better understanding of what happens to intrinsic properties of the streams as they pass through these components. Using this framework, we analyze the characteristics of temporal locality present in streams in different points of the topology.

To this end, we have proposed new metrics to capture the two causes of temporal locality: popularity and correlation. Entropy was defined as a natural metric for measuring the skew in the relative popularity of different objects in a request stream, and we argue that it is better for representing this property than is the Zipf's law exponent. We also show that normalized entropy accurately captures the popularity component of temporal locality by relating normalized entropy to two commonly used measures of this property: hit ratio and Zipf exponent. The Coefficient of

Variation of the IAT distribution was used as a metric for temporal correlation, motivated by the fact that the presence of such a correlation between accesses to the same object should cause a deviation from the geometric distribution. We validated these metrics using a wide collection of logs from different points in the Web topology, and showed that they are intuitive and effective.

We then showed how these metrics can give valuable insights when applied to streams under the transformation framework. For example, they yield an important observation that concerns the effectiveness of caching hierarchies: while caches themselves *decrease* temporal locality, aggregation and disaggregation can *increase* temporal locality. These observations provide guidance for overall design of the Web system: *e.g.*, in suggesting that stream aggregation (perhaps more than network topology) is a key factor in optimizing cache placement.

This point is refined by our separate measurement of the two components of locality. While the popularity component can sometimes increase (leading to the effect just described), all three transformations were found to diminish the correlation component, and this effect was confirmed by the consistent decline in correlation found in streams moving up the hierarchy from clients toward servers.

Considering directions for further study, we note that the framework we propose can be used with metrics that assess other properties of the streams. These might include inter-object reference correlation (analogous to spatial locality) and the distributions of size of the objects. Furthermore, considering real time instead of virtual time suggests using the framework as a basis to study the characteristics of request arrival processes.

In this paper, we did not tackle the problem of measuring the inter-object access correlations. However, as noted in [33], we believe that a good estimator for the actual entropy rate present in the request streams can yield a measure of the prefetching potential of the stream, which is a result of inter-object correlation.

Finally, regarding the metrics for temporal locality we developed and their use under the proposed framework, we believe that in some cases, bounds on the values of the resulting streams can be determined (given the parameters of the input request streams and of the transformation node). For example, we have had some progress in determining the upper bound on the (non-normalized) entropy of the output stream of an aggregation, given that there are no common objects between the streams.

Despite the need for these additional studies, we conclude that the metrics and framework we propose appear to be important tools for understanding and better designing the Web as a system of multiple interacting components that create, transform and absorb streams of requests.

References

[1] Virgílio Almeida, Azer Bestavros, Mark Crovella, and Adriana de Oliveira. Characterizing reference locality in the WWW. In *Proceedings of the Fourth International Conference on Parallel and Distributed Information Systems (PDIS96)*, December 1996.

[2] M. Arlitt and T. Jin. Workload Characterization of the 1998 World Cup Web Site. *IEEE Network, Special Issue on Web Performance*, 14(3):30–37, May–June 2000.

[3] Mohit Aron, Darren Sanders, Peter Druschel, and Willy Zwaenepoel. Scalable content-aware request distribution in cluster-based network servers. In *Proceedings of the 2000 Annual Usenix Technical Conference*, pages 323–336, 2000.

[4] Abdullah Balamash and Marwan Krunz. Application of multifractals in the characterization of WWW traffic. In *Proceedings of the ICC 2002 Conference – Symposium on High-Speed Networks*, April 2002.

[5] Paul Barford, Azer Bestavros, Adam Bradley, and Mark Crovella. Changes in Web client access patterns: Characteristics and caching implications. *World Wide Web*, 2:15–28, 1999. Special Issue on Characterization and Performance Evaluation.

[6] Paul Barford and Mark Crovella. Generating representative web workloads for network and server performance evaluation. In *Proceedings of the ACM SIGMETRICS Conference*, pages 151–160, June 1998.

[7] Les A. Belady. A study of replacement algorithms for a virtual-storage computer. *IBM Systems Journal*, 5(2):78–101, 1966.

[8] Berkeley Computer Science Department Access Logs. `http://www.cs.berkeley.edu/logs/`.

[9] Azer Bestavros. Using speculation to reduce server load and service time on the www. In *Proceedings of CIKM'95*, Baltimore, Maryland, November 1995.

[10] Azer Bestavros, Robert L. Carter, Mark E. Crovella, Carlos R. Cunha, Abdelsalam Heddaya, and Sulaiman A. Mirdad. Application-level document caching in the Internet. In *Proceedings of IEEE SDNE '95)*, 1995.

[11] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web Caching and Zipf-like Distributions: Evidence and Implications. In *Proceedings of IEEE Infocom*, April 1999.

[12] Pei Cao and Sandy Irani. Cost-aware WWW proxy caching algorithms. In *Proceedings of the 1997 Usenix Symposium on Internet Technologies and Systems (USITS-97)*, Monterey, CA, 1997.

[13] Ludmila Cherkasova and Gianfranco Ciardo. Characterizing temporal locality and its impact on web server performance. In *Proceedings of IEEE ICCCN*, pages 434–441, October 2000.

[14] Carlos A. Cunha, Azer Bestavros, and Mark E. Crovella. Characteristics of WWW client-based traces. Technical Report TR-95-010, Boston University Department of Computer Science, April 1995.

[15] Peter J. Denning. The working set model for program behavior. *Communications of the ACM*, 11(5):323–333, May 1968.

[16] Peter J. Denning and Stuart C. Schwartz. Properties of the working-set model. *Communications of the ACM*, 15(3):191–198, March 1972.

[17] Derive 5. `http://www.derive.com`.

[18] Ron Doyle, Jeff Chase, Syam Gadde, and Amin Vahdat. The trickle-down effect: Web caching and server request distribution. In *Proceedings of the Sixth International Workshop on Web Caching and Content Delivery*, pages 1–18, June 2001.

[19] Li Fan, Pei Cao, Jussara Almeida, and Andrei Z. Broder. Summary cache: a scalable wide-area Web cache sharing protocol. *IEEE / ACM Transactions on Networking*, 8(3):281–293, 2000.

[20] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. RFC 2616: Hypertext transfer protocol—HTTP/1.1, June 1999.

[21] Steven Glassman. A caching relay for the World Wide Web. In *Proceedings of the First International World Wide Web Conference*, pages 69–76, 1994.

[22] The Internet Traffic Archive. `http://ita.ee.lbl.gov/`.

[23] Shudong Jin and Azer Bestavros. Sources and Characteristics of Web Temporal Locality. In *Proceedings of the 8th MASCOTS*. IEEE Computer Society Press, August 2000.

[24] Shudong Jin and Azer Bestavros. GreedyDual* Web Caching Algorithm: Exploiting the Two Sources of Temporal Locality in Web Request Streams. *International Journal on Computer Communications*, 24(2):174–183, February 2001.

[25] Shudong Jin and Azer Bestavros. Temporal locality in web request streams. Technical Report BUCS-TR-1999-014, Boston University Computer Science Department, July 2002.

[26] Terence Kelly. Thin-client web access patterns: Measurements from a cache-busting proxy. In *Proceedings of the 6th Web Caching Workshop*, Boston, MA, 2001.

[27] Terence Kelly and Daniel Reeves. Optimal Web cache sizing: Scalable methods for exact solutions. In *Proceedings of the 5th Web Caching Workshop*, May 2000.

[28] B. Krishnamurthy, C. Wills, and Y. Zhang. On the use and performance of Content Distribution Networks. In *Proceedings of SIGCOMM Internet Measurements Workshop*, pages 169–182, November 2001.

[29] A. Mahanti, D. Eager, and C. Williamson. Temporal locality and its impact on web proxy cache performance. *Performance Evaluation Journal: Special Issue on Internet Performance Modelling*, 42(2/3):187–203, September 2000.

[30] R. L. Mattson, J. Gecsei, D. R. Slutz, and I. L. Traiger. Evaluation techniques for storage hierarchies. *IBM Systems Journal*, 9(2):78–117, 1970.

[31] NLANR IRCache Project. `http://www.ircache.net/`.

[32] Venkata N. Padmanabhan and Lili Qui. The content and access dynamics of a busy web site: findings and implicatins. In *SIGCOMM*, pages 111–123, 2000.

[33] Gopal Pandurangan and Eli Upfal. Can entropy characterize performance of online algorithms? In *Symposium on Discrete Algorithms*, pages 727–734, 2001.

[34] V. Phalke and B. Gopinath. An interreference gap model for temporal locality in program behavior. In *Proceedings of the 1995 ACM SIGMETRICS Conference*, pages 291–300, 1995.

[35] Patrícia Saraiva, Edleno Moura, Rodrigo Fonseca, Wagner Meira Jr., Berthier Ribeiro-Neto, and Nivio Ziviani. Rank-preserving two-level caching for scalable search engines. In *Proceedings of the 24th ACM SIGIR*, pages 51–58, September 2001.

[36] Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423 and 623–656, July and October 1948.

[37] Jeffrey R. Spirn. Distance string models for program behavior. *Computer*, 9(11):14–20, November 1976.

[38] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley and Sons, 1991.

[39] W3C Web Characterization Repository. `http://repository.cs.vt.edu/`.

[40] Jia Wang. A survey of Web caching schemes for the Internet. *ACM Computer Communication Review*, 25(9):36–46, 1999.

[41] Carey Williamson. On filter effects in web caching hierarchies. *ACM Transactions on Internet Technology*, 2(1):47–77, February 2002.

A Approximating the Entropy of a Zipf Distribution

In determining the α exponent for a Zipf-like distribution given a value for the entropy and the number of objects, we find it useful to replace the sums in Equation 5 with integrals, such that we can get a closed form for the equation. We reproduce the equation here for reference:

$$H_{zipf}^{\alpha,N}(X) = \log_2 S + \frac{\alpha}{S} \sum_{i=1}^{N} i^{-\alpha} \log_2 i$$

We have two sums that we need to approximate, given by Equations (7) and (8) below.

$$S = \sum_{i=1}^{n} i^{-\alpha} \tag{7}$$

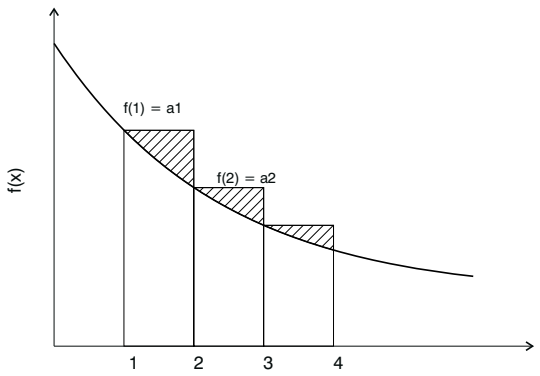


Figure 12: Relation between the integral and the discrete sum of $f(x)$

$$\sum_{i=1}^{n} i^{-\alpha} \log_2 i \tag{8}$$

We detail the approximation to Equation (7); the process for Equation (8) is analogous, and we only show the final result for this equation. The approach we take is to find an integral that approximates the sum with a bounded error, and find an approximation for this error.

First, let $f(x)$ be a continuous function, and call $f(i) = a_i$. The n-th partial sum is given by

$$s_n = \sum_{i=1}^{n} a_i = \sum_{i=1}^{n} f(i).$$

We use the integral of $f(x)$ from 1 to $n+1$ to approximate the sum. The relationship between both can be understood by looking at Figure 12. The error of the approximation for a given n, given by the difference $\sum_{i=1}^{n} f(i) - \int_1^{n+1} f(i)\,\mathrm{d}i$, is exactly the sum of the shaded areas. Under some circumstances, this error converges, even if the sum does not. For example, when $\alpha = 1$ Equation (7) is the harmonic series, which is a divergent series. However, the limit of the difference, when $n \to \infty$, is the famous Euler-Mascheroni constant, 0.5772156...

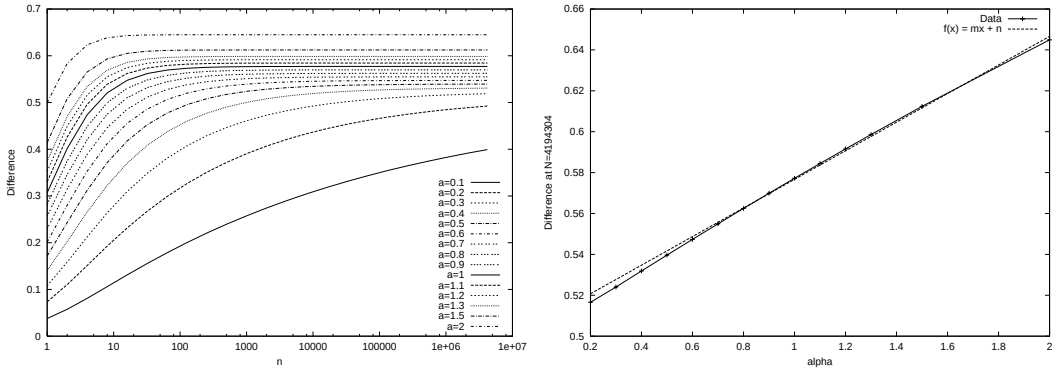
We do not set out to prove the convergence of the error here, but simply to determine it empirically for the range of values that we are interested in. We inspect the space in which α varies between 0.1 and 2, and look at values of n up to about 4×10^6.

Our first approximation to equations 7 and 8 is then given by using the integrals (for $\alpha \neq 1$)

$$S = \sum_{i=1}^{n} i^{-\alpha} \approx \int_1^{n+1} i^{-\alpha}\,\mathrm{d}i = \frac{(n+1)^{1-\alpha} - 1}{1-\alpha}, \text{ and}$$

$$\sum_{i=1}^{n} i^{-\alpha} \log_2 i \approx \frac{1}{\ln 2} \int_1^{n+1} i^{-\alpha} \ln i\,\mathrm{d}i = \frac{1 - (n+1)^{1-\alpha}[(\alpha-1)\ln(n+1) + 1]}{\ln 2(\alpha-1)^2}$$

By examining the difference between the integral and the sum, we obtain the curves plotted in Figure 13(a). We observed that the differences seem to converge to values that depend approximately linearly on the value of α. This observation was also valid for Equation (8), although the convergence is slower, specially for values of α smaller than 0.5.



(a) Difference versus n for different values of α

(b) Difference versus α and linear approximation

Figure 13: Approximating the sum in Equation (7) with an integral

Equation	p	q
7	0.0699	0.506737
8	0.014941	0.120051

Table 5: Parameters for correcting the errors in the approximations of the entropy of a Zipf distribution

We performed a linear fit for the values of the error at large value of n, for both equations using the the model $\epsilon = p \cdot \alpha + q$. The result for the first equation is shown in Figure 13(b), and the values for p and q are shown in Table 5, for both equations.

Thus, we use the following approximations for the sums:

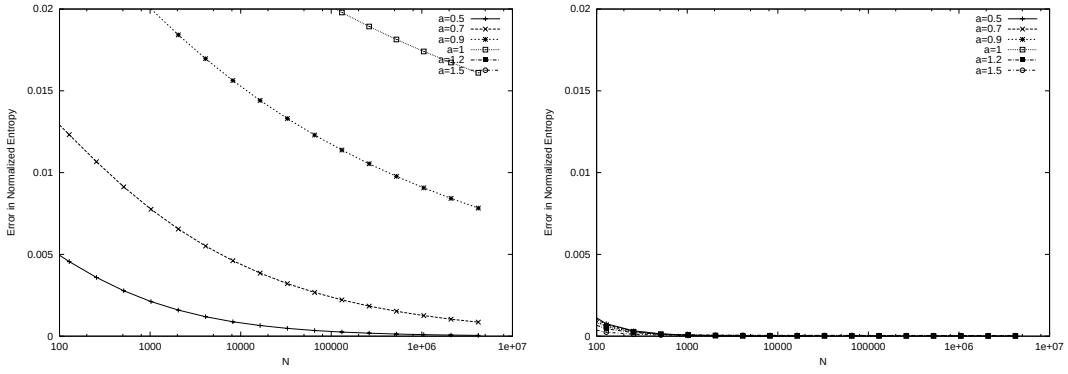
$$S = \sum_{i=1}^{n} i^{-\alpha} \approx \frac{(n+1)^{1-\alpha} - 1}{1-\alpha} + (0.0699 \cdot \alpha + 0.506737), \text{ and}$$

$$\sum_{i=1}^{n} i^{-\alpha} \log_2 i \approx \frac{1 - (n+1)^{1-\alpha}[(\alpha-1)\ln(n+1) + 1]}{\ln 2(\alpha-1)^2} + (0.014941 \cdot \alpha + 0.120051).$$

With these values we were able to obtain very good approximations of the entropy. In Figure 14 we show the error for the normalized entropy when using the integrals only (a), and using the integrals corrected (b). The complete equation for $H_{zipf}^{\alpha,N}(X)$, replacing the sums with the approximations above, was input into the program Derive [17] to find the numeric solutions for α, given N and H, as we mention in Section 4.1.

B Limit on the CV

In this section we present the proof to the conjecture that the CV for an object i that has n references in the log is at most $\sqrt{n}$. Since we are dealing with strictly non-negative quantities, we



(a) Integrals without the corrections

(b) Corrected equations

Figure 14: Errors in the approximation for the normalized entropy using integrals, with and without the corrections

set out to show the equivalent result that the value of the CV squared, when this is computed from n samples, cannot exceed n.

As defined previously in Equation 6, the CV as we use it is the standard deviation divided by the mean. We want to show that

$$\frac{\sigma^2}{\mu^2} \leq n.$$

For each object i that has n references in the log, the calculation of cv_i uses n IAT samples.

We use the unbiased estimator for the variance of a population, given by Equation 9. The limits in the summations range from 1 to n, unless otherwise specified, and have been omitted in some equations, to improve readability.

$$\sigma^2 = var(X) = \frac{n\sum x_i^2 - (\sum x_i)^2}{n(n-1)} \tag{9}$$

The empirical CV for n samples is then the variance (9), divided by the average of the x_i squared, and is given by

$$\mathrm{CV}^2 = \left(\frac{n}{n-1}\right)\frac{n\sum x_i^2 - (\sum x_i)^2}{(\sum x_i)^2}\ . \tag{10}$$

Lemma 1. *For all $x_i > 0$, and for $n \geq 1$,*

$$\sum_{i=1}^{n} x_i^2 \leq (\sum_{i=1}^{n} x_i)^2.$$

Proof. This is equivalent to

$$\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2 \leq 0\ .$$

The proof is by induction on n.

Base: Trivial for $n = 1$, the two summations have the exact same value: $x_1^2 - x_1^2 = 0$.

Induction Step: For $n + 1$,

$$\sum_{i=1}^{n+1} x_i^2 - \left(\sum_{i=1}^{n+1} x_i\right)^2 =$$

$$\left(\sum_{i=1}^{n} x_i^2 + x_{n+1}^2\right) - \left[\left(\sum_{i=1}^{n} x_i\right)^2 + 2x_{n+1}\sum_{i=1}^{n} x_i + x_{n+1}^2\right] =$$

$$\left[\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2\right] - 2x_{n+1}\sum_{i=1}^{n} x_i. \tag{11}$$

By the induction hypothesis, $\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2 \leq 0$. Since $x_i > 0\ \forall i$, $-2x_{n+1}\sum_{i=1}^{n} x_i \leq 0$, and if follows that (11) is always ≤ 0.

□

Theorem 1. *For $x_i > 0,\ \forall i$; for $n > 1$; and CV² as defined in (10), CV² $\leq n$.*

Proof. From (10),

$$\begin{aligned}\left(\frac{n}{n-1}\right)\frac{n\sum x_i^2 - (\sum x_i)^2}{(\sum x_i)^2} &\leq n \\ = \frac{n\sum x_i^2 - (\sum x_i)^2}{(\sum x_i)^2} &\leq (n-1) \\ = \sum x_i^2 &\leq (\sum x_i)^2\end{aligned}$$

The proof thus follows directly from Lemma 1. □