Extracting large-scale knowledge bases from the web*

Ravi Kumar   Prabhakar Raghavan   Sridhar Rajagopalan   Andrew Tomkins

IBM Almaden Research Center

Abstract

The subject of this paper is the creation of knowledge bases by enumerating and organizing all web occurrences of certain subgraphs. We focus on subgraphs that are signatures of web phenomena such as tightly-focused topic communities, webrings, taxonomy trees, keiretsus, etc. For instance, the signature of a webring is a central page with bidirectional links to a number of other pages. We develop novel algorithms for such enumeration problems. A key technical contribution is the development of a model for the evolution of the web graph, based on experimental observations derived from a snapshot of the web. We argue that our algorithms run efficiently in this model, and use the model to explain some statistical phenomena on the web that emerged during our experiments. Finally, we describe the design and implementation of Campfire, a knowledge base of over one hundred thousand web communities.

1 Overview

The subject of this paper is the creation of knowledge bases by enumerating and organizing all web occurrences of chosen subgraphs. For example, consider enumerating all *cliques* of size four or more, where a clique is a set of web pages each of which links to all the others. Each such clique could represent an alliance between the creators of these pages: business partners, *keiretsus*, members of a family, etc. If we could enumerate all cliques on the web, then organize and annotate them into a usable structure, we would have created a knowledge base of all such alliances—patent as well as latent—as evident in the link structure of the web. More generally, we consider the creation of knowledge bases from an analysis of the web graph using the following paradigm: (1) identify a *signature* subgraph that is likely to arise in every element to be represented in the knowledge base; (2) devise a method for enumerating every instance of this subgraph in the web graph; (3) reconstruct, from each enumerated subgraph, the associated element of the knowledge base; and (4) annotate and index the elements to make the knowledge base usable. In the example above, step (1) could consist of identifying a clique of size four as likely to be present in every keiretsu (say); step (2) would require an efficient method for enumerating cliques of size 4; step (3) would require assembling all pages in a keiretsu given the portion represented by the 4-clique; and step (4) could consist of extracting and indexing statistically significant keywords from the assembled pages.

While the first of these steps is specific to the elements that will populate the knowledge base, the other three share some common challenges that will be our focus here. Foremost among these challenges: enumerating subgraphs on large graphs is, in the worst case, infeasible—from a complexity-theoretic as well as practical standpoint. Clearly, we must exploit the fact that the web is not a "worst-case" graph. To this end, we develop a stochastic model of the web graph that exhibits good agreement with statistics from the web, and show that a traditional random graph model could not exhibit such agreement. We develop an algorithmic paradigm for subgraph enumeration problems, run a concrete instance on the web, and show that its good performance is predicted by our web graph model. We describe the ongoing Campfire project, in which we enumerate, annotate, and index over 100,000 web communities generated using our methods.

Locally dense regions and communities. We begin with the following motivating example:

Example 1 *Consider the set of web pages that point to both* `www.boeing.com` *and* `www.airbus.com`. *There are more than two thousand such pages on today's web, including personal pages, aircraft museums, vendors, legal services, and so forth. Almost all of these pages represent some type of resource list of airplane manufacturers. The subgraph induced by these thousand pages, and the pages they point to, has a specific form: a number of resource lists*

*IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120. Email: {ravi, pragh, sridhar, tomkins}@almaden.ibm.com

Proceedings of the 25th VLDB Conference, Edinburgh, Scotland, 1999.

all point to some subset of a set of resources, in this case Boeing, Airbus, and dozens of other aircraft manufacturers. Moreover, pages within the subgraph frequently—but not always—cross-reference each other. (*See Figure 1.*)

A knowledge base containing all structures such as the one in Figure 1—aircraft manufacturers, long distance phone companies, US national parks, etc.—would clearly be of tremendous value. This motivates the definition of structures we call *bipartite cores:* a bipartite core in a graph consists of two (not necessarily disjoint) sets of nodes L and R, such that every node in L links *to* every node in R. Note that links from R to L, or within R or L, may or may not be present.

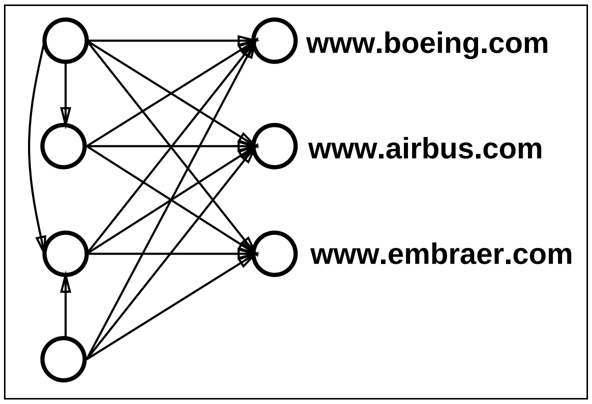

Figure 1: A bipartite core.

Indeed, one may envision building knowledge bases from enumerations of many different interesting structures—bipartite cores, cliques, *webrings* (which manifest themselves as star-shaped graphs with bidirectional links on the spokes), pages in a hierarchically organized website, or newsgroups and newsgroup discussion threads (which manifest themselves as bidirectional paths). The reasons for doing so include: (1) Such knowledge bases represent a better starting point for deeper analyses and mining than raw web data. Indeed, this is the goal of the Campfire project. (2) Structure can be used more effectively for searching and navigation. For instance, an agent responsible for searching a database containing dense bipartite graphs could pay more attention to text surrounding the relevant links, for their annotative value. (3) Fine-grained structures provide a basis for targeted market segmentation. (4) Studying these enumerated structures over time gives us insight into the sociological evolution of the web.

Challenges and approaches. From an algorithmic perspective, the naive "search" algorithm for enumeration suffers from two fatal problems. First, the size of the search space is far too large—using the naive algorithm to enumerate all bipartite cores with two web pages pointing to three pages would require examining approximately 10^{40} possibilities on a graph such as the web with 10^8 nodes. Second, and more practically, the algorithm requires random access to edges in the graph, which implies that a large fraction of the graph must effectively reside in main memory to avoid the overhead of seeking a disk on every edge access.

Although these obstacles appear insurmountable, we exploit additional structure latent in the web graph. For instance, the average number of links out of a page is small. However, the web is not a *random* sparse graph—it is a sparse graph containing many structures (*cliques* for instance) that arise only in far denser random graphs. The reason the web contains these structures is that, despite its overall sparseness, *local regions of the web are dense*. We propose novel algorithms that exploit this structure of the web graph to overcome these challenges.

To understand the performance of our algorithms, we develop a stochastic model for "web-like" graphs. The model exhibits two desirable properties: (1) it agrees with a number of statistical observations about the web graph, and (2) it is a useful tool in algorithm design, suggesting both explanations for the performance of our algorithms, and future directions for other efficient web analyses.

Here is a preview of a key technical element of our stochastic web graph model: intuitively, *new pages are created by borrowing random fragments of existing pages.*

Guided tour of this paper. In Section 2 we detail a number of measurements we have made of a snapshot of the web graph; these include the distribution of in- and out-degrees of nodes, and the numbers of bipartite cores. In Section 3 we motivate and develop our stochastic model for the web graph. We give analyses showing that our model—besides being a plausible high-level process for the creation of the web graph—explains our measurements from Section 2 in ways that traditional random graph models could not. Section 4 describes our three main algorithms: the elimination/generation paradigm for subgraph enumeration, an extension of a link-based web search algorithm for extending bipartite cores into communities, and an index extraction algorithm. In Section 5 we give some results of the Campfire project on organizing web communities.

1.1 Related previous work

Link analysis. A number of web search projects have used links to enhance the quality and reliability of the search results; see for instance HITS [19], its variants [5, 9, 10], and Google [6]. The connectivity server [4] also provides a fast index to linkage information on the web. Dean and Henzinger [12] combine heuristic improvements from [5] and [10] and apply these to the problem of finding related pages on the web.

Sociometrics. Statistical analysis of the structure of the academic citation graph has been the subject of much work in the Sociometrics community. As we discuss below, Zipf distributions seem to characterize web citation frequency. Interestingly, the same distributions have also been observed for citations in the academic literature. This fact, known as *Lotka's law*, was demonstrated by Lotka in 1926 [23]. Gilbert [16] presents a probabilistic model explaining Lotka's law, which is similar in spirit to our proposal, though different in details and application.

Data mining. Traditional data mining research (see for instance Agrawal and Srikant [1]) focuses largely on algorithms for finding association rules and related statistical correlation measures in a given dataset. However, efficient methods such as *a priori* [1] or even more general methodologies such as query flocks [29], do not scale to the numbers of "items" (pages) in the web dataset. This number is currently around four hundred million, which is two to three orders of magnitude more than the number of items in a typical market basket analysis. Further, the graph-theoretic structures we seek could correspond to association rules with very small support and confidence. Our conviction that these structures are interesting comes from additional insight about the web graph, rather than from traditional support and confidence measures.

In the case of bipartite cores, the relation we are interested in is co-citation. Co-citation is effectively the join of the web citation relation with its transpose, the web "cited by" relation. The size of this relation is potentially much larger than the original citation relation. Thus, we need methods that work with the original, without explicitly computing the co-citation relation.

The work of Mendelzon and Wood [25] is an instance of structural methods in mining. They argue that the traditional SQL query interface to databases is inadequate in its power to specify several structural queries that are interesting in the context of the web. We note that this is not the case here. The signature graphs we are interested in are relatively easy to specify in SQL.

Enumerating bipartite cores. In recent work [20] we reported enumerating over 200,000 bipartite cores from a snapshot of the web. The contributions reported in that paper were: (1) a special case of the algorithmic paradigm introduced here; (2) preliminary statistical observations about the web that we extend and explain here, using the web graph model developed in this paper; and (3) A random sampling experiment showing that barely 5% of the cores arose coincidentally. The large scale and high quality of such enumerated structures provides a compelling basis for building knowledge bases out of them. A number of these results, including Kleinberg's HITS algorithm [19], are discussed in the survey paper [21].

Henzinger *et. al.* [17] study algorithmic and memory bottleneck issues in related graph computations from a theoretical perspective, primarily to derive impossibility results. For a survey of database techniques on the web and the relationships between them, see Florescu, Levy, and Mendelzon [13].

2 Measurements of the web graph

In this section we describe a set of measurements generated from a crawl of the web from 1997, provided by Alexa, Inc. As our primary focus is the linkage patterns between pages,

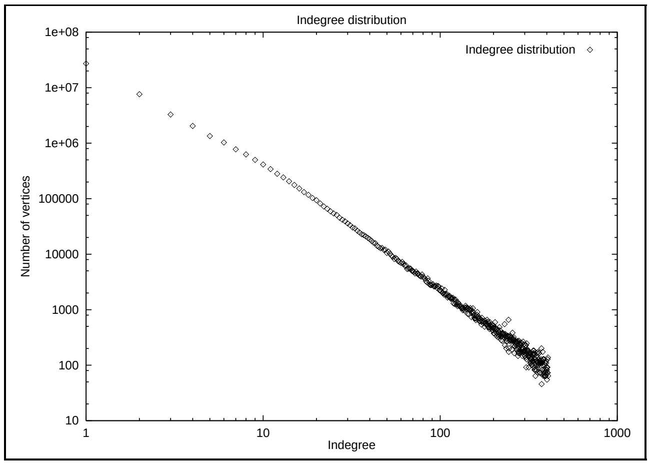

Figure 2: In-degree distribution.

we consider only the interconnection patterns of hyperlinks, and abstract away the textual content of each page. We view each page as a node of a directed graph, and each hyperlink as a directed edge. These measurements provide crucial insights for our web graph model (Section 3) as well as in our efficient enumeration algorithms (Section 4).

In-degree. We say that a hyperlink is an *out-link* of its source page, and an *in-link* of its destination page. We call the number of out-links and in-links of a page the *out-degree* and *in-degree* of the page, respectively. We begin by counting the in-degree of each page. Since the average out-degree is around eight, and since each edge contributes equally to the total in- and out-degree, the average in-degree must also be around eight. Our interest is in understanding not just the average, but the complete distribution of in-degrees. Figure 2 shows a log-log plot of the number of pages that have in-degree i as a function of i. The linearity of the curve indicates that the distribution is an inverse polynomial. Fitting an inverse polynomial to the data we find the probability that a page has i in-links to be roughly proportional to $i^{-2.1}$. We will also refer to inverse polynomial distributions as *Zipfian distributions* [30].

An important characteristic of Zipfian distributions is that they have high probability of deviating significantly from the mean. Thus, although the mean in-degree of a page is about 8, there is a significant probability that a page will have 1000 in-links (for example, approximately 100K pages on the web have ≥ 1000 in-links).

Out-degree. Our next observation concerns the roughly Zipfian distribution of out-degrees in the web graph. Figure 3 shows a log-log distribution of i versus the number of pages that have out-degree i. This curve also follows a Zipfian distribution. Fitting an inverse polynomial to the data, we note that a random web page has out-degree i with probability approximately $i^{-2.38}$.

$C_{i,j}$ **counts.**

In Section 1 we described bipartite cores, which consist of two sets of pages L and R, such that every page in L links *to* every page in R. If L contains i pages, and R contains j pages, we refer to the core as a $C_{i,j}$.

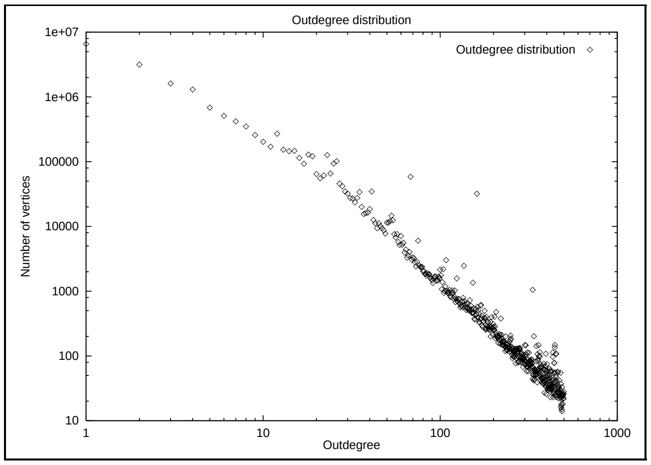

Figure 3: Out-Degree distribution.

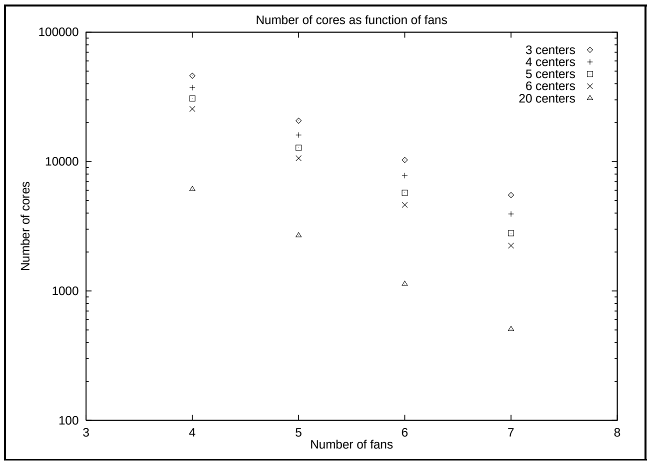

Figure 4: Number of $C_{i,j}$'s as a function of i.

During the exhaustive enumeration [20] of bipartite cores, we created a subgraph of the web consisting of all pages remaining after the algorithm prunes away all nodes of degree less than four. We enumerated all $C_{i,j}$'s in the resulting graph, for all $i \in \{2, \ldots, 7\}$ and $j \in \{3, \ldots, 20\}$. We now analyze the trends in that dataset. We begin by looking at the dependence of the number of $C_{i,j}$'s on i. Figure 4 shows a number of curves representing fixed values of j and four values of i—we display only counts for $i \geq 4$ since the pruning removed nodes of degree less than 4. To avoid clutter, we show values of j ranging from 3 to 6, and 20—intermediate values have the same form. As the figure shows, the number of $C_{i,j}$'s drops exponentially as a function of i, in this range.

Next, Figure 5 shows the curves representing fixed values of i for various values of j. The graph shows a log-log plot with linear behavior. Fitting an inverse polynomial to the data, the number of $C_{i,j}$'s as a function of j drops as j^{-c}, where the value of c is between 1.09 and 1.4.

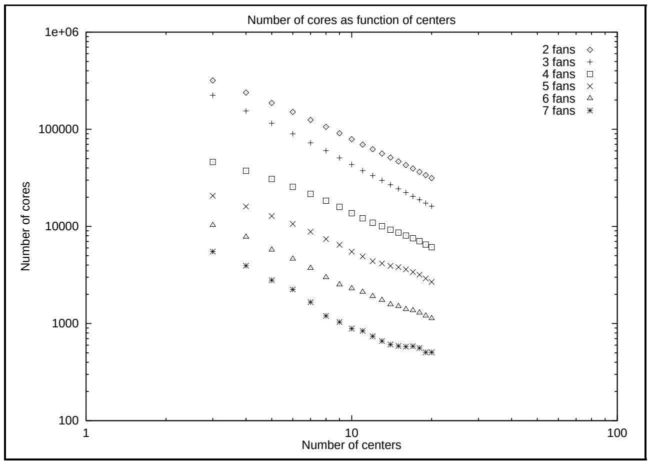

Figure 5: Number of $C_{i,j}$'s as a function of j.

3 Models for web-like graphs

In this section we present our model for web-like graphs, motivated by the following goals:

1. Understand various structural properties of the web graph—in- and out-degrees, neighborhood structures, etc. Of particular interest to us is the distribution of web structures that are signatures of the components of a knowledge base (e.g., the bipartite cores we are interested in).

2. Perform a more realistic analysis of algorithms on the web graph—this is of particular interest because worst case analysis of many algorithms is particularly pessimistic and unrealistic when applied to the web graph.

3. Predict properties of the web graph based on the model—this is of interest because it can lead to better algorithmic and structural insight.

We seek to model the linkage structure of the web graph. In particular, our models do not describe textual content

We begin with a discussion of the properties such a model should have, and then present a general framework for a class of graph models called *copying models*. Next, we give a simple concrete instance of a copying model, and analyze the concrete model to predict the parameters measured in Section 2. We show that measurements and predicted behavior exhibit strong agreement.

3.1 Desiderata for a web graph model.

1. *Simplicity*: It should have a succinct and fairly natural description.

2. *Plausibility*: It should be rooted in a plausible macro-level process for the creation of content on the web. We cannot hope to model the detailed behavior of the many users creating web content. Instead, we only desire that the aggregate formation of web structure be captured well by our graph model. Thus, while the

model is described as a stochastic process for the creation of individual pages, we are really only concerned with the aggregate consequence of these individual actions. Therefore we seek a model that is plausible at this aggregate level.

3. *Topics and communities*: It should provide structure corresponding to the strongly-linked "topics" that have emerged on the actual web, *but* it should not do so by requiring some *a priori* static set of topics that are part of the model description—the evolution of interesting topics and communities should instead be an emergent feature of the model.[1] Such a model has several advantages:

   - *Viability*: It is extremely difficult to characterize the set of topics on the web; thus it would be useful to draw statistical conclusions without such a characterization.
   - *Dynamism*: The set of topics reflected in web content has proven to be fairly dynamic. Thus, the shifting landscape of actual topics will need to be addressed in any topic-aware model of time-dependent growth.

4. *Statistics*: We would like the model to reflect many of the structural phenomena we have observed in the web graph. These include:

   (a) *Zipfian distributions*: A Zipfian distribution [30] for the number of links *to* a web page; in particular, the number of pages with i in-links is well-approximated [20] by $1/i^2$. A similar phenomenon has been observed in the study of scholarly citations [11, 23], and is the basis of studies in the sociology of scientific communities [16].

   (b) *Locally dense globally sparse structure*: Although the web graph is relatively sparse (the average number of links out of a page is roughly 7.2), the graph contains well over one hundred thousand bipartite cores with at least six nodes in them, even after mirrors are deleted. This is because the web graph, though globally sparse, has many locally dense regions.

5. *Evolution:* We should capture the phenomenon that the web graph has nodes and edges appearing and disappearing with time.

Interestingly and unfortunately, Items 3, 5, and all the criteria of Item 4, fail to hold for traditional models [7] of random graph theory. For instance, traditional random graph models would predict in- and out-degrees that are Poisson distributed, rather than the significantly heavier-tailed Zipfian distributions we have observed. It is easy to extend traditional random graph models to capture evolution, but this has not been addressed primarily because in standard models it is unlikely that interesting mathematical phenomena arise from evolution.

[1] In particular, we avoid models of the form "Assume each node is some combination of topics, and add an edge from one page to another with probability dependent on some function of their respective combinations."

3.2 Intuition for our models

Items 1 and 2 of the desiderata listed above ask that our model encapsulate a simple, plausible notion of content creation on the web. Our notion is based on the following intuition:

- some page creators on today's web may create content and link to other sites without regard to the topics that are already represented on the web, but
- *many* page creators will be drawn to existing topics of interest to them, and will link to pages within some of these existing topics.

Consider a user intent on creating a page about recreational sailing. Like most content creators, this user would probably wish to incorporate some links that would be of interest to potential visitors to the page. In order to gather together such links, the user would probably begin to browse around, perhaps using the many excellent resource lists[2] already available about recreational sailing, and choosing links based on his or her particular preferences within the topic. In the end, the resulting page would be another resource list about the topic, albeit one with a new personal spin.

We draw two lessons from this example: first, if a number of users have created links to a page, a new user will be more likely to link to that page than to a random page (partly because the page is probably of higher quality, and partly because the page is easier to find). And second, a user who has added a link to a sailing page is more likely to add another sailing link than an arbitrary link. Or said another way, if a user links to a page, and some existing resource list also links to that page, the user may link to something else appearing on the resource list.

Rephrasing these observations in purely graph-theoretic terms:

1. A new page is more likely to link to pages with higher in-degree.

2. A new page is more likely to link to two pages that co-occur on some resource list than to two random pages.

We therefore propose the following intuition for a simple process of link creation, which results in behavior obeying the previous observations: *A new page adds links by picking an existing page, and copying some links from that page to itself.*

[2] by *Resource lists* we mean pages that collect links on one or more topics, ranging in scope from a carefully-researched node of Yahoo to the small personal page of an individual who has collected several links about lateen-rigged sailing vessels.

We reiterate that this process is *not* meant to reflect individual user behavior on the web; rather, it is a local procedure which in aggregate works well in describing page creation on the web. The model also implicitly captures topic creation as follows: First, a few scattered pages begin to appear about the topic. Then, as users interested in the topic reach critical mass, they begin linking these scattered pages together, and other interested users are able to discover and link to the topic more easily. This creates a "locally dense" subgraph around the topic of interest. Furthermore, copying is a powerful mechanism for giving rise to bipartite cores: an author creates a resource list, and others create pages pointing to many pages on this resource list.

This intuitive view summarizes the process from a page-creator's standpoint; we now recast this formulation in terms of a random graph model.

3.3 A class of graph models

We model the web as an evolving graph, in which nodes and edges appear and disappear with time. Our models are described by four stochastic processes—*creation* processes $\mathcal{C}_v$ and $\mathcal{C}_e$ for node- and edge-creation, and *deletion* processes $\mathcal{D}_v$ and $\mathcal{D}_e$ for node- and edge-deletion. These processes are discrete-time processes. Each process is a function of the time step, and of the current graph.

Consider, for instance, the following node creation and deletion process: at time step t, independent of all earlier events, create a node with probability $\alpha_c(t)$. We could have a similar model with parameter $\alpha_d(t)$ for node deletion. Deleting a node also deletes all its incident edges. Clearly, we would tailor these probabilities to reflect the growth rates of the web and the half-life of pages respectively.

We present edge processes ranging from simple to complex to model the web with increasing fidelity. At this stage, we state a complex model we believe to be largely realistic. In Section 3.4, we show that even a greatly simplified process induces a Zipf distribution on in-degrees.

At each step we choose (possibly by random sampling) a node v to add edges out of, and a number of edges k that will be added to v. With probability β we add k edges from v to nodes chosen independently and uniformly at random. With probability $1-\beta$, we choose another vertex u, at random, and *copy* k edges from u to v. That is, after choosing a node u at random, create k (directed) edges (v, w) such that (u, w) is a random edge incident at u. One might reasonably expect that much of the time, u will not have out-degree larger than k; if the out-degree of u exceeds k we pick a random subset of size k. If on the other hand the out-degree of u is less than k we first copy the edges out of u, then pick another random node u' to copy from, and so on until we have enough edges. Such a copying process is not unnatural, and consistent with the qualitative intuition at the beginning of this section.

As a simple example of an edge deletion process, at time t, delete a random edge with probability $\delta(t)$.

3.4 The α and (α, β) models

We illustrate these ideas with a very simple special case that captures destination copying. We show that even in this simple case, the induced in-degree distribution is Zipfian. A node is created at every step. Nodes and edges are never deleted, so the graph keeps on growing. This corresponds to setting $\alpha_c(t)$ to 1, and $\alpha_d(t)$ and $\delta(t)$ to 0.

We now concentrate on the edge creation process. As each node comes into being, with probability $\alpha \in (0, 1)$ it adds an edge to itself, and with probability $1-\alpha$ it picks a random edge and copies the edge onto itself, i.e., choose a random edge (u, w) created earlier and add the edge (v, w).

We now argue that the in-degree distribution of nodes in the α model is Zipfian.

Let $p_{i,t}$ be the fraction of nodes at time t with in-degree i. Assume t is sufficiently large that $p_{i,t} = p_{i,t+1}$. Then at time t there are $t \cdot p_{i,t}$ nodes with in-degree i, and at $t+1$ there are $(t+1) \cdot p_{i,t+1}$ such nodes. Therefore the probability that an additional node has in-degree i at time $t+1$ is $(t+1)p_{i,t} - tp_{i,t} = p_{i,t}$. The probability that a node with in-degree $i-1$ garners the single new edge is $(1-\alpha)(i-1)tp_{i-1,t}$, and the probability that a node with in-degree i gains the new edge and therefore becomes in-degree $i+1$ is $(1-\alpha)(it)p_{i,t}$. It must be the case that:

$$\begin{aligned}
(1-\alpha)(i-1)tp_{i-1,t} - (1-\alpha)(it)p_{i,t} &= p_{i,t} \\
(1-\alpha)i(p_{i-1,t} - p_{i,t}) &= p_{i,t} \\
(1-\alpha)i\frac{dp_{i,t}}{di} &= -p_{i,t} \\
(1-\alpha)\frac{dp_{i,t}}{p_{i,t}} &= -\frac{di}{i} \\
(1-\alpha)\ln p_{i,t} &= -\ln i \\
p_{i,t} &= 1/i^{\frac{1}{(1-\alpha)}}.
\end{aligned}$$

As an example, set $\alpha = 1/2$. In this process, each new edge flips a fair coin, and on heads points to the newest page, but on tails points to the destination of a uniformly-chosen random in-link. This process will then have an in-degree distribution given by $1/i^2$, following our observations of the web.

Now, consider an extension to the α model yielding the (α, β) model. Each time a new page arrives, a single edge is added as follows. The process flips two independent coins with probabilities α and β of coming up heads. The new edge (u, v) is built as follows:

The "destination" v is set based on the α coin: if it comes up heads, v is the newest page, and otherwise, v is the destination of a random link. The "source" u is set based on the β coin: if it comes up heads, u is the newest page, and otherwise, u is the source of a random link.

Since the original α model chooses edge destinations without reference to edge sources, the in-degree distribution of the (α, β) model will also be given by $p_{i,t} = 1/i^{\frac{1}{(1-\alpha)}}$. And since the same analysis applies if we flip the definition of "source" and "destination," the out-degree distribution q is given by $q_{i,t} = 1/i^{\frac{1}{(1-\beta)}}$ for large enough t.

For example, setting $\alpha = .52$ and $\beta = .58$, the model generates a graph with the following properties:

1. $\Pr[\text{A node has in-degree } i] \longrightarrow i^{-2.1}$.
2. $\Pr[\text{A node has out-degree } i] \longrightarrow i^{-2.38}$.

Both of these values match the web.

3.5 Resource list copying for bipartite cores

The (α, β) model of Section 3.4 induces Zipfian in- and out-degrees, but edges are added one at a time. In the general framework of Section 3.3, a model with more topic focus would copy several links from the same resource list. In this section, we explore the impact of copying multiple links on $C_{i,j}$ formation.

Recall that a $C_{i,j}$ is a bipartite core whose set L contains i nodes, and whose set R contains j nodes. We will refer to the elements of L as *fans*, and the elements of R as *centers*. In [20] it is shown that the web contains over 133,000 $C_{3,3}$ cores. We consider first a traditional random graph model, showing that such a model will not predict this large number of $C_{3,3}$'s. Next, we show that our model will in fact predict such a large number.

Example 2 *Let $n = 100,000,000 = 10^8$, and consider a traditional random graph model where every link is independently present with probability 10^{-7} (so that the average out-degree is 10, somewhat higher than reality). A 6-tuple of nodes $L = \{a, b, c\}$ and $R = \{x, y, z\}$ forms a $C_{3,3}$ if all nine edges from L to R are present, an event occurring with probability $(10^{-7})^9 = 10^{-63}$. There are approximately $n^6/720 = 10^{47}/72$ ways of choosing such 6-tuples. Thus the expected number of $C_{3,3}$'s is approximately*

$$10^{-63} \times 10^{47}/72 < 10^{-17} \ll 1,$$

Turning this calculation around, we can compute how large an out-degree an average page must have in the traditional random graph model, in order for 133,000 $C_{3,3}$'s to arise in the web: roughly 1200, a number that is again inconsistent with the web.

A similar calculation with our model yields an expected number of roughly 100,000 $C_{3,3}$'s. Consider the model of Section 3.3, with no deletion processes; thus we allow the copying of a random subset of links from a resource list. We omit the details, but the key idea is the following: the probability that a and b both copy the same 3 links from a single resource list c is $\Omega(1/n^2)$ even though the out-degree of a node is a small constant independent of n. When they both copy the same 3 links from the same source, a $C_{3,3}$ results. There are some technicalities that stand in the way of this being a mathematically provable statement; chief of these is the fact that in the early stages of copying, we may attempt to copy from a resource list that does not have 3 links to copy. However, in the "steady state" this should not be a significant effect; proving this rigorously remains an interesting open direction in random graph theory.

4 Algorithms

We identify three steps in the construction of a knowledge base of communities on the web: (i) efficiently enumerating the subgraphs (i.e., cores) of interest; (ii) collecting additional web pages related to each enumerated core to build a community; and (iii) extracting statistically significant information from each such community, leading to a searchable index of these communities. Accordingly in this section we describe three algorithms that we view as central to each of these steps: (i) the *elimination/generation* paradigm for efficient subgraph enumeration on web-scale graphs, using a relatively lean computational infrastructure; (ii) an extension of Kleinberg's HITS algorithm [19] that can take as input only a set of URL's present in a core (and no query terms), to produce authoritative web pages on the community centered around this core; and (iii) a method for extracting statistically significant index terms with which to index such communities. Note that the first and the third steps are generic to building indexable knowledge bases of communities from subgraph enumeration. The second is specified in terms of bipartite cores for concreteness, although the reader may readily see its extension to other subgraphs.

4.1 Enumerating cores

An algorithm in the elimination/generation paradigm performs a number of sequential passes over the web graph. The graph is stored as a binary relation on disk, and thus is not available for random access. During each pass, the algorithm writes a modified version of the dataset to disk for the next pass. It also collects some metadata in main memory which serves as state during the next pass. Passes over the data are interleaved with sort operations, which change the order in which the data is scanned, and constitute the bulk of the processing cost. During each pass over the data, *elimination* and *generation* operations are interleaved. The details are given below:

Elimination. There are often easy necessary (though not sufficient) conditions that have to be satisfied in order for a node to participate in a subgraph of interest to us. Consider for instance the problem of enumerating cliques of size four. We can prune any node whose in-degree or out-degree—*at any stage of the process*—is less than 4 (the significance of this will become apparent below). Consider next the example of $C_{4,4}$'s. Any node with in-degree 3 or smaller cannot participate on the right side of a $C_{4,4}$. Thus, edges which are directed into such nodes and the nodes themselves can be pruned from the graph. Likewise, nodes with out-degree 3 or smaller cannot participate on the left side of a $C_{4,4}$. We refer to these necessary conditions as *elimination filters*.

Generation. Generation is a counterpoint to elimination. Nodes that barely qualify for potential membership in an interesting subgraph can easily be established as either belonging to such a subgraph or not. Consider again the example of a $C_{4,4}$. Let u be a node of in-degree exactly

4. Then, u can belong to a $C_{4,4}$ *if and only if* the 4 nodes that point to it have a neighborhood intersection of size at least 4. It is possible to test this property relatively cheaply. We define a *generation filter* to be a procedure that identifies barely-qualifying nodes, and for all such nodes, either outputs a subgraph or proves that such a subgraph cannot exist.

If the test embodied in the generation filter is successful, we have identified an interesting subgraph. Furthermore, regardless of the outcome, this node can be marked for pruning since all potential interesting subgraphs containing it have already been enumerated.

Similar schemes can be developed for other structures.

Example 3 *Consider enumerating all subgraphs in which four web pages all point to one another. Then, any node with fewer than three links out of it can be eliminated. Likewise, any node with fewer than three links into it can be eliminated. Thus, the elimination filter for such an enumeration finds nodes with fewer than three in-links or out-links. The generation filter finds nodes with exactly three in-links or out-links, and checks to see whether the resulting set of four nodes, namely the original and the three adjacent nodes, is a clique. Notice that the generation filter is substantially cheaper in this case since there is no exhaustive enumeration of subsets of size* 4 *required as would have been the case if the in and out degrees had each been substantially larger than* 3.

Note that if edges appear in an arbitrary order, it is not clear that the elimination filter can be easily applied. If, however, the edges are sorted by source (resp. destination), it is clear that the elimination filter can be applied to the out-links (resp. in-links) in a single scan.

The generation filter is slightly more complicated to implement efficiently. We first explain how this can be done in 3 sub-passes over the data, in the setting of Example 3 above. In the first sub-pass, we construct a list of nodes with in-degree or out-degree 3, along with their in- or out-neighborhoods. We assume that this list fits in main memory during the second sub-pass; if not we can break the second sub-pass into several phases, each processing a main memory-sized chunk of candidates. In the second sub-pass, we verify that each node in the neighborhood points to all other nodes in the neighborhood. At the end of the second sub-pass we output all the interesting subgraphs, and in the third sub-pass, we delete all the candidate nodes.

Notice that the first and the third passes can be overlapped with the preceding and subsequent elimination filtering pass. Thus, the effective cost of the generation filter is just one pass over the data. Also, note that even the first round of the elimination/generation paradigm will eliminate a large fraction of the destination nodes in the graph (see Example 4 for details).

After one elimination/generation phase, the remaining nodes have fewer neighbors than before in the residual graph, which may present new opportunities during the next pass. We can continue to iterate until we do not make significant progress. Depending on the filters, one of two things could happen: either we repeatedly remove nodes from the graph until nothing is left or after several passes, the benefits of elimination/generation "tail off" as fewer and fewer nodes are deleted at each phase. If for instance our elimination filter were "delete all nodes with fewer than 100 in-links from `.com` pages", we will eliminate almost all pages on the web. On the other hand if the elimination filter were "delete all nodes with fewer than 3 out-links", we will reach a fixed point at which repeated elimination passes do not reduce the size of the residual graph substantially (though this may not happen after the first such pass).

Why should such algorithms run fast? We make a number of observations about their behavior:

1. The in/out-degree of every node drops monotonically during each elimination/generation phase.

2. During each generation test, we either remove a node u from further consideration (by developing a proof that it can belong to no instance of the subgraph of interest), or we output a subgraph that contains u. Thus, the total work in generation is linear in the size of the graph plus the number of subgraphs enumerated, assuming that each generation test runs in constant time. This is the case if we are enumerating a constant-sized subgraph; thus our algorithm is *output sensitive*.

3. As shown in Example 4 below, elimination phases rapidly eliminate most nodes in the web graph. A complete mathematical analysis of iterated elimination is beyond current techniques, but the example below shows that just the first elimination phase by itself is quite powerful.

Example 4 *Consider the enumeration of* $C_{3,3}$*'s. By the inverse quadratic law for in-degrees, eliminating nodes on the basis of in-degree prunes away any node with in-degree* ≤ 2. *As described in Section 2, out-degrees also have a (different) Zipfian distribution, and we can prune away any node with out-degree* ≤ 2. *Finally, applying a generation phase means that we also remove nodes with in/out-degree* equal *to 3. From simple calculations with the corresponding probabilities (noting for instance that* $\sum_i 1/i^2 = \pi^2/6$*), we determine that these elimination/generation steps would account for nearly 90% of the nodes in just the first iteration. Further iterations are less dramatic—both in theory and in practice.*

It is thus clear that the insights from our measurements and model drive the efficiency of the elimination/generation paradigm. A more mathematically precise analysis of the running time of elimination/generation in our model appears to be beyond the reach of current random graph theory, and is a worthwhile goal.

4.2 From cores to communities

We now describe an extension of Kleinberg's HITS [19], to expand a core $C_{i,j}$ into its surrounding community. For brevity we only detail the novel aspects of our extension; the reader is referred to [19] for details on the basic HITS algorithm. To make our description succinct, we use the notation $p \rightarrow q$ to denote that a web page p has a hyperlink to a web page q.

Given a query, the basic HITS algorithm assembles a small set of pages $\mathcal{R}$ (the *root set*) using a traditional text-based search engine and then constructs $\mathcal{R}'$ (the *expanded set*) by adding all pages that points to or is pointed to pages in the root set. I.e., $\mathcal{R}' = \mathcal{R} \cup \{p \mid p \rightarrow q, q \in \mathcal{R}\} \cup \{q \mid p \rightarrow q, p \in \mathcal{R}\}$. Each page p is associated with a pair of scores $h(p), a(p)$ which are initially set to 1. The algorithm iteratively updates these scores as $h(p) = \sum_{p \rightarrow q} a(q)$ and $a(p) = \sum_{q \rightarrow p} h(q)$, (after appropriate normalization). The top-scoring pages based on $a(\cdot)$ (resp. $h(\cdot)$) are termed "authorities" (resp. "hubs").

In our case, we have no text query; instead we have a core $C_{i,j}$. Since there is no query, the root set $\mathcal{R}$ is constructed using the following rule: the root set consists of (i) the core $C_{i,j}$, (ii) all pages pointed to by the nodes in L and (iii) all pages that point to at least two nodes in R. I.e.,

$$\mathcal{R} = L \cup \{p \mid q \rightarrow p, q \in L\} \cup$$

$$R \cup \{p \mid p \rightarrow q_1, p \rightarrow q_2, q_1, q_2 \in R\}.$$

We apply the basic HITS algorithm to this root set (making the assumption that the community around the core overlaps significantly with the root set). The result is a set of authoritative sources of information for that community, together with a set of hubs that collect together and annotate these authoritative pages.

4.3 Extracting index terms

Having enumerated the cores and run the generalized HITS algorithm above to expand each core into a full community, we must extract from each community index terms to build the knowledge base, and a summary that can be used to identify the contents of the community.

We make the following observation: the title of a page is likely to contain terms describing the page. Using this heuristic, we implement the following algorithm for identifying keywords that are useful for both indexing and summarization:

1. Extract the titles of all web pages in a community (which comprises the top 50 hubs and 50 authorities as returned by the algorithm of Section 4.2)
2. Eliminate stop words (e.g., articles, "html", "home", "page", "web", etc.)
3. Rank the resulting terms by frequency across the entire set of pages
4. Return the top ten most frequent words

We use the resulting set of 10 terms to index the community, and build a search utility against this index.

We are investigating more sophisticated strategies for the future. For instance, anchortext—the text in the vicinity of the "href" tags at the tails of hyperlinks—often represents a description of the contents of the linked-to page. Thus, anchortext of good hubs around links pointing to good authorities may be important for generation of index terms and summaries. We also plan to use phrase extraction techniques instead of simple keyword extraction.

5 Campfire: a knowledge base of communities

In this section we describe Campfire, a knowledge base of communities. We spell out the details of our experimental setup and give some examples of the communities indexed in Campfire.

5.1 Experimental setup

The communities in the Campfire knowledge base were constructed from a 1997 crawl of the web. In addition to extracting cores from the tape, we also extracted the state of Yahoo at the time of the crawl. As a reference point, Yahoo contained slightly over 16,000 topics in this crawl.

The trawling algorithms were run on a PC with a 333 MHz Intel Pentium II processor. The machine had 256M of main memory and a SCSI chain with multiple disk drives. The cost of running the trawling algorithm has two major components: (i) the data cleaning portion, and (ii) the pruning-based mining algorithm.

Since 2 mirrored copies of a page would generate a spurious $C_{3,j}$, we removed mirrors using the shingling method of Broder et al [8]. Our algorithm shingles 3100 pages/second. This implies that the entire set of potential fan pages (2 Million pages in our case) are shingled in about 10 minutes. Eliminating duplicates (pages which have the same shingle) runs at about the same speed (about 3000 pages/second). Thus, initial data cleaning takes only about half an hour. The major cost in the pruning step is in sorting the edge data. There are 60M edges in the dataset after shingling. The time to sort the edge set is 5 minutes per pass (on average). Our algorithms make 30 passes over the data (for i, j = $3, 3$) in the worst case. The total time to trawl all $C_{3,3}$ cores was about 1.5 hours.

Having enumerated the cores, we then expanded them into communities using the algorithm of Section 4.2. To construct the root set, we augmented the fans and centers by following links on today's web. However, since the cores date back to 1997, and the half-life of a web page is of the order of a few weeks, many of these fans and centers no longer exist. Therefore, we only expanded those cores at least half of whose fans were still alive on today's web. For example, in the case of $C_{4,j}$ cores, 41% passed this test. Interestingly, the fraction of centers that are still alive is around 54%, suggesting that centers might have a higher half-life than fans. Running HITS typically expands a core into a community of between 100 and 4000 pages, with the average around 1300. The number of hyperlinks between these pages varies from 200 to 15000, with the average around 3500. Parsing each page takes a few milliseconds, but running the expansion algorithm takes between 2 and 10 seconds.

The final task is to index these communities. To accomplish this, we run the index term extraction algorithm on each of these communities. Given that we already have a parsed version of these pages, running this algorithm takes under a millisecond for each page. Using the keywords extracted, we index each community into Campfire.

5.2 Sample communities

We now provide 5 sample communities automatically generated by expanding $C_{4,j}$ cores. We present the top five hubs and authorities of each community along with indexing keywords, and a brief (manually-generated) annotation for the benefit of the reader. A more detailed list of pages can be found at `www.almaden.ibm.com/cs/k53/campfire.html`

To validate our conclusions further, we conducted the following experiment on the extracted communities. We took a random sample of about 100 communities and manually applied our resource compilation tools [10] to generate, for each community, a high-quality set of relevant pages. We then computed the overlap between these carefully-compiled resource lists and their counterparts in Campfire. The average overlap of sites was around 60%, suggesting that the quality and purity of communities extracted by our algorithm is fairly high.

6 Further directions

Our work raises a number of interesting directions and questions.

1. What are other instances of subgraph structures (possibly in conjunction with text patterns) that can be used to build valuable knowledge bases?

2. Our web graph model leads to very different areas for further investigation:

   (a) What are the dominant modes of content creation that different kinds of users exhibit, and how can these be reconciled (in an aggregate sense) with our model?

   (b) We have only scratched the surface of the rigorous mathematical analysis of our graph model (in contrast to the analysis of simpler, more traditional random graph models). Explicating the structure of graphs generated by our model is far more challenging than these traditional models, but a worthwhile goal in being able to predict the evolution of the web graph.

3. We have given illustrative examples and some general principles for devising elimination/generation algorithms for several interesting subgraph structures. Are there systematic ways for developing elimination/generation algorithms for any web subgraph enumeration problem? How do we exploit the interplay between the nature of the web graph and the elimination/generation phases? What are systematic techniques for tuning this methodology for given resource constraints?

4. In Campfire, we have a knowledge base built from web communities by extending the cores we enumerate from the web. We have not elicited significant semantic content (beyond extracting title words and indexing them for text search). Are there new paradigms for annotating and organizing these knowledge atoms in ways that are more valuable to users?

References

[1] R. Agrawal and R. Srikant. Fast Algorithms for mining Association rules. *Proceedings of VLDB,* Sept, 1994, Santiago, Chile.

[2] G.O. Arocena, A.O. Mendelzon, G.A. Mihaila, Applications of a Web query language, *Proc. 6th International World Wide Web Conference*, 1997.

[3] A.E. Bayer, J.C. Smart, G.W. McLaughlin, Mapping intellectual structure of scientific subfields through author co-citations, *J. American Soc. Info. Sci.*, 41(1990), pp. 444–452.

[4] K. Bharat, A. Broder, M.R. Henzinger, P. Kumar, and S. Venkatasubramanian. The connectivity server: fast access to linkage information on the web. *Proceedings of WWW7,* Brisbane, Australia, April, 1998.

[5] K. Bharat and M.R. Henzinger. Improved Algorithms for Topic Distillation in a Hyperlinked Environment. *Proceedings of ACM SIGIR*, 1998.

[6] S. Brin and L. Page, The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Proceedings of the 7th World-wide web conference (WWW7),* 1998.

[7] B. Bollobás, *Random Graphs,* Academic Press, 1985.

[8] A. Broder, S. Glassman, M. Manasse and G. Zweig. Syntactic clustering of the Web. In *Proceedings of the Sixth International World Wide Web Conference*, April 1997, pages 391-404.

[9] S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, P. Raghavan and S. Rajagopalan. Automatic Resource Compilation by Analyzing Hyperlink Structure and Associated Text, *Proceedings of the 7th World-wide web conference (WWW7),* 1998.

[10] S. Chakrabarti, B. Dom, S. Ravi Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Experiments in Topic Distillation. SIGIR workshop on hypertext information retrieval, 1998.

[11] H.T. Davis. The Analysis of economic time series., Principia press, 1941.

[12] J. Dean and M.R. Henzinger. Finding related pages in the World Wide Web. *Proceedings of the 8th WWW conference*, 1999.

[13] D. Florescu, A. Levy and A. Mendelzon. Database Techniques for the World-Wide Web: A Survey. SIGMOD Record 27(3): 59-74 (1998).

[14] E. Garfield, Citation analysis as a tool in journal evaluation, *Science*, 178(1972), pp. 471–479.

[15] E. Garfield, The impact factor, *Current Contents*, June 20, 1994.

[16] N. Gilbert. A simulation of the structure of academic science. *Sociological Research Online*, 2(2), 1997.

[17] M.R. Henzinger, P. Raghavan, and S. Rajagopalan. Computing on data streams. *AMS-DIMACS series,* special issue on computing on very large datasets, 1998.

[18] M.M. Kessler, Bibliographic coupling between scientific papers, *American Documentation*, 14(1963), pp. 10–25.

[19] J. Kleinberg, Authoritative sources in a hyperlinked environment, *Proc. ACM-SIAM Symposium on Discrete Algorithms*, 1998. Also appears as IBM Research Report RJ 10076(91892) May 1997.

[20] S. Ravi Kumar, P. Raghavan, S. Rajagopalan and A. Tomkins. Trawling emerging cyber-communities automatically. In *Proceedings of the 8th World-wide web conference,* 1999.

[21] J.M. Kleinberg, S. Ravi Kumar, P. Raghavan, S. Rajagopalan and A. Tomkins. The web as a graph: measurements, models and methods. Invited paper in *Proceedings of the Fifth Annual International Computing and Combinatorics Conference,* 1999.

[22] R. Larson, Bibliometrics of the World Wide Web: An exploratory analysis of the intellectual structure of cyberspace, *Ann. Meeting of the American Soc. Info. Sci.*, 1996.

[23] A.J. Lotka. The frequency distribution of scientific productivity. Journal of the Washington Academy of Sciences. 16, p.317, 1926.

[24] M. Marchiori, The Quest for Correct Information on the Web: Hyper Search Engines, *The 6th International World Wide Web Conference (WWW6),* 1997.

[25] A. Mendelzon and P. Wood. Finding Regular Simple Paths in Graph Databases. SIAM J. Comp. 24(6), 1995, pp. 1235-1258.

[26] P. Pirolli, J. Pitkow, R. Rao, Silk from a sow's ear: Extracting usable structures from the Web, *Proc. ACM SIGCHI Conference on Human Factors in Computing*, 1996.

[27] E. Rivlin, R. Botafogo, B. Shneiderman, Navigating in hyperspace: designing a structure-based toolbox, *Communications of the ACM*, 37(2), 1994, pp. 87–96.

[28] E. Spertus, ParaSite: Mining structural information on the Web, *Proc. 6th International World Wide Web Conference*, 1997.

[29] D. Tsur, J. Ullman, S. Abiteboul, C. Clifton, R. Motwani, S. Nestorov, and A. Rosenthal. Query Flocks: A Generalization of Association Rule Mining. *Proceedings of ACM SIGMOD*, 1998.

[30] G.K. Zipf. Human behaviour and the principle of least effort. New York: Hafner, 1949.

AUTHORITIES	HUBS
www.midnightbeach.com/hs	user.pa.net/∼rtregl/schools.html
www.home-school.com	larch-www.lcs.mit.edu:8001/∼raymie/linklist.html
www.comenius.org/chnpage.htm	www.phonet.com/bsimon/educ-nat.html
users.aol.com/WERHSFAM/humor.html	home.sol.no/∼hunwww/Elinker.htm
www.sound.net/∼ejcol/confer.html	www.thefamily.net/educational.html
indexing keywords: *homeschool homeschooling school education resource*	

Homeschooling.

AUTHORITIES	HUBS
www.netimages.com/∼chile	www.chetbacon.com/hotlinks.html
www.webring.org/cgi-bin/webring?list&ring=chile	bigsun.wbs.net/homepages/g/g/h/gghosey/links.html
www.firegirl.com	www.bizkid.com/food.htm
www.azstarnet.com/∼coriel	www.catechnologies.com/cuisinesites.html
www.chilegod.com	allison.clark.net/pub/yoda
indexing keywords: *hot food sauce chile cooking recipes spicy peppers*	

Hot and spicy foods.

AUTHORITIES	HUBS
manufacture.com.tw	www.aunet.com.tw/steel.htm
www.rack.com.tw	www.commerce.com.tw/c/metal
www.mold.net.tw	www.acer.net/search/ee.htm
www.pack.com.tw	www.hinet.net/source/business/steel/3.htm
www.or.com.tw	www.hinet.net/source/business/steel/1.htm
indexing keywords: *taiwan corporation products international machinery*	

Taiwanese machine shops.

AUTHORITIES	HUBS
www.eren.doe.gov	www.physic.ut.ee/∼janro/ab.html
www.doe.gov	www.csmi.com/oaatweb/links.htm
www.epri.com	www.azstarnet.com/∼dcat/Rec_list.htm
www.epa.gov	www.beasley.com.au/news.htm
www.ashrae.org	wwwvms.utexas.edu/∼whcii/energy/bldglnks.htm
indexing keywords: *energy wind renewable sustainable*	

Sustainable energy resources.

AUTHORITIES	HUBS
cancer.med.upenn.edu	www.allny.com/health/oncology.html
wwwicic.nci.nih.gov	micf.mic.ki.se/Diseases/c4.html
www.cancer.org	www.medmark.org/onco/onco.html
www.nci.nih.gov	www.meds.com/cancerlinks.html
www.nlm.nih.gov	www.mic.ki.se/Diseases/c4.html
indexing keywords: *cancer breast oncology*	

Cancer.