

1 Graph structure in the web

Here, we investigate the graph structure of the *Web-graph*, the directed graph whose nodes are *static pages* and edges are *links* between these pages.

Broder et al. [1] looked at some basic properties of this graph. They found that the web-graph has one large weakly connected component consisting of 91% of pages. This is not surprising since there is only *one* world wide web. This fact remains intact even when all nodes of degree more than 5 are removed: there still is a large component with more than 59 million pages. This shows that there are many different paths between pairs of nodes and the web will not be partitioned even if certain *important* pages were removed. The web was found to contain one strongly connected component (SCC) of approximately 56 million pages that were *mutually reachable*.

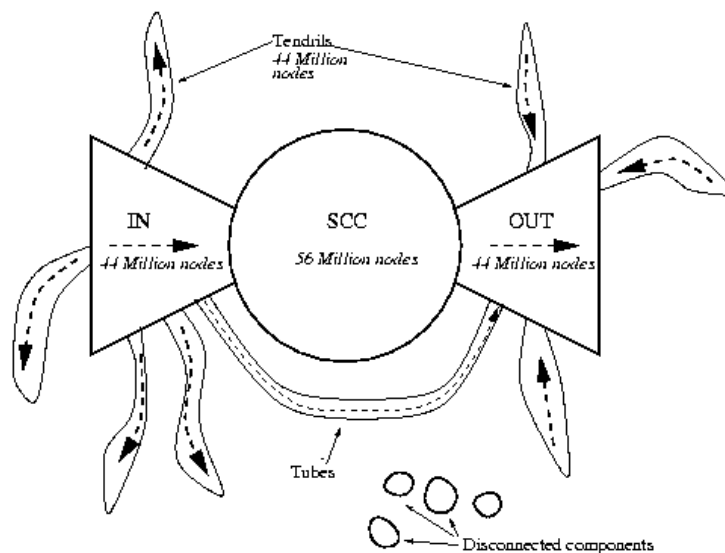


Figure 1: Connectivity of the web

Figure 1 gives an overview of the structure. One can pass from any node of IN through SCC to any node of OUT. Hanging off IN and OUT are TENDRILS containing nodes that are reachable from portions of IN, or that can reach portions of OUT, without passage through SCC. It is possible for a TENDRIL hanging off from IN to be hooked into a TENDRIL leading into OUT, forming a TUBE – a passage from a portion of IN to a portion of OUT without touching SCC. They found that apart from the roughly 59 million pages in SCC, IN and OUT each had roughly 44 million pages. The TENDRILS and TUBES comprised of another roughly 44 million pages.

As a result, if we select two nodes at random, there is a directed path from one to the other with probability roughly 30%. For this happens either when both nodes are in SCC, or one is in IN and the other in SCC or OUT, or the first one in SCC and the second in OUT. (In addition, there are some more less

frequent cases.) Looking at the sizes of the components then shows that the probability of such selections is about 30%, contrary to the once popular belief that almost all nodes are connected to each other via directed paths.

The above structure and relative sizes can be expected to be quite stable, even if new web pages are discovered. The reason is that any paths from OUT to IN, from SCC to IN, or from OUT to SCC, would already have been discovered if they existed. So these components will not be contracted.

Calculating the Diameter: The diameter can be calculated by running the BFS algorithm and calculating the longest shortest path. This takes $O(mn)$ time, which is too slow for the web graph, where m and n are of the order of billions. On the other hand, it is not known how to compute the diameter of a graph faster than this. Instead, we may be satisfied with merely a lower bound, which can be obtained by randomly sampling nodes and measuring the distance only between sampled nodes. This provides us with lower bounds on the diameter. Using these techniques, Broder et al. showed that the diameter of the SCC is at least 28, and the diameter of the whole WWW at least 503. The much larger bound on the entire WWW is likely due to long chains of pages (chapters of a book, for instance), which would likely be either in the OUT component, or inside tendrils. The *average* distance between sampled pairs of nodes in the SCC was found to be ca. 16 if the edge direction was considered and ca. 7 if the edge direction was disregarded. An interesting question with regards to the sampling is for which types of graphs it will actually give *good* lower bounds, which are close to the true diameter.

2 Converting graphs of the web to something useful: How to search?

Writing a search engine involves many challenges (apart from writing the crawler itself), which can be broken down into three main categories. The first type of challenge relates to handling natural language processing and information retrieval issues. The programmer has to deal with the fact that different people have different authoring styles and that the content is stored in different formats (html, pdf, jpg, etc.). A relatively recent but extremely relevant problem is that of adversarial IR, which is visible in the form of link-spam and the field of search engine optimization in general. The second kind of problems are queries that are too short, a common occurrence since all the users of a search engine might not be well versed with the nuances of searching (grad. student versus his/her parents). This leads to the *abundance* problem, in which the number of pages related to the word can be extremely large (think of the number of pages with the word “car”) and you have to order the pages according to their *relevance*, which itself is not clearly defined. Another related problem is the fact that often the search term does not appear in the target page (the words “car manufacturers” do not appear on the websites of Toyota and Honda, for example). The third problem area is that of inaccurate and outdated data. One way of solving this partially is by frequent crawling to maintain an up-to-date web graph.

Constructing a (nearly complete) web graph involves performing multiple crawls with different starting points and then taking a union of the graphs obtained. A basic problem in this context is to find the starting points for the crawls. Since we need to start at *known* pages and we cannot *know* pages unless we crawl, we end up with a chicken and egg problem. This can be solved by asking users to submit URLs or by auto-generating IP addresses. A nice starting strategy would be to start at *popular* pages. One of the natural strategies can be to start at a random page from the set we already have (e.g. from randomly generated IP addresses).

A crawler comes across various problems, some natural and some man-made, during the execution of its task. Some of the natural problems being link loops, overloading web servers and connection timeouts. The man-made problems, which may be tougher to solve, include text spamming, where an author places *interesting* words in his page to increase its relevance. Another kind of spamming is link spamming, whereby a selfish poster posts comments on open sites like wikis and blogs with hyperlinks to his website.

Degree distribution: [1] and previous works have suggested that the distribution of degrees follows a power law. The power law for in-degree states that *the number of nodes having in-degree i is proportional $1/i^x$ (for some $x > 1$)*. Pages with *low* out-degree follow a different distribution, possibly Poisson or a combination of Poisson and power law. The distribution of sizes of weakly connected components exhibits a power law

with exponent roughly 2.5. The power law degree distribution is exciting because this conclusively proves that the web *cannot* be modeled as a Erdős–Rényi random graph, the most commonly used graph model.

3 Authoritative Sources in Hyperlinked Environment

There are various types of queries

- Specific queries : E.g., “Does Netscape support the JDK 1.1 code-signing API ?” The difficulty in handling specific queries : Scarcity Problem : There are very few pages that contain the required information, and it is often difficult to determine the identity of these pages.
- Broad-topic queries : E.g., “Find information about the Java programming language.” Abundance problem : The number of pages that could reasonably be returned as relevant is far too large for a human user to digest. Simple text-based indexing fails here if we do not take into account the link structure etc.
- Similar-page queries : E.g., “Find pages ‘similar’ to java.sun.com.”
- Current events (needs updating the database regularly)

One strategy for handling broad-topic queries is to identify *relevant* pages from a large collection of candidates. Using the parameters mentioned in [2], the candidate set is approximately 200 top text-based search results, together with all out-linking pages from these and some (approx. 50) in-linking pages each. This results in an induced graph G . A page could be interesting for two reasons: it could be a good *authority*, holding genuine information about the topic, or a good *hub*, collecting links to many good authorities. In turn, an authority is likely pointed to by many good hubs. The tacit assumption here is that links constitute an endorsement, via which hubs can confer authority to the pages they point to, and authorities confer “hub weight” to pages that point to them.

This definition is circular, but we can translate it into an actual algorithm as follows. Each page v has authority weight a_v and hub weight h_v , initialized to 1. In any one iteration, these are updated, by having authorities inherit authority weights from the hubs that point to them, and hubs inherit hub weight from the authorities they point to.

$$\begin{aligned} a'_v &= \sum_{u \rightarrow v} h_u && \text{for all } v \\ h'_v &= \sum_{v \rightarrow u} a'_u && \text{for all } v \end{aligned}$$

Notice that after one iteration, the authority weight of v is the number of pages pointing to v , which is intuitively related to the authority of a page. Further iterations refine this notion by giving more weight to hubs that are deemed more relevant.

To ensure that this rule may converge to a solution, we need to normalize the weights after each update, for instance such that $\sum a_v^2 = 1$ and $\sum h_v^2 = 1$ (otherwise, the values would keep growing). If the rule were to converge, it would have to converge to fixed points $\vec{a} = (a_v)_V \in \mathbb{R}^n$ and $\vec{h} = (h_v)_V \in \mathbb{R}^n$ where $n = |V|$. Being a *fixed point* means that the values of the authority and hub weights do not change in further iterations (for any page), i.e., $\vec{a} = \vec{a}'$ after normalization.

We can express the above operations in terms of matrices as follows. Let B be the adjacency matrix of G , i.e., $B_{ij} = 1$ iff there is an edge $i \rightarrow j$. Then, we can rewrite the update rules as:

$$\begin{aligned} \vec{a}' &= B^T \cdot \vec{h} \\ \vec{h}' &= B \cdot \vec{a}' \end{aligned}$$

Hence, the update rule for authority weights can be written as $\vec{a}'(B^T B) \cdot \vec{a}$. At a fixed point, \vec{a} and \vec{a}' have to be the same up to the normalization constant λ we had to multiply by, so $\vec{a} = \lambda \cdot (B \cdot B^T) \cdot \vec{a}$. Thus, the authority weights are exactly an eigenvector of B^T with eigenvalue λ .

Since $B^T B$ is symmetric, \mathbb{R}^n has a basis of orthonormal eigenvectors $\vec{\omega}_1, \vec{\omega}_2, \dots, \vec{\omega}_n$ of $B^T B$. Let $\lambda_1, \lambda_2, \dots, \lambda_n$ be the corresponding eigenvalues with $|\lambda_1| \geq |\lambda_2| \geq \dots \geq |\lambda_n|$. We can always write the starting vector in terms of this basis as $\vec{a}_0 = \sum_{i=1}^n \alpha_i \vec{\omega}_i$ with $\alpha_1 \neq 0$. The next approximation to the authority weight vector then satisfies

$$\begin{aligned} \vec{a}_1 &= (B^T B) \cdot \vec{a}_0 &= \sum_{i=1}^n \alpha_i (B^T B) \vec{\omega}_i \\ &= \sum_{i=1}^n \alpha_i \lambda_i \vec{\omega}_i. \end{aligned}$$

By induction, we can now show that for any k , $\vec{a}_k = \sum_{i=1}^n \alpha_i \lambda_i^k \vec{\omega}_i$.

Hence, whenever $|\lambda_1| > |\lambda_2|$, as $k \rightarrow \infty$, in the normalized version, we converge to $\vec{a}_\infty = \vec{\omega}_1$. Therefore, authority weights are just the first eigenvector of $B^T B$. (hub weights are similarly the first eigenvector of BB^T).

The authority weight computation can thus simply be thought of as using the *power iteration* method to compute the first eigenvector of $B^T B$. Instead, one could also use different techniques to compute the eigenvector. Usually, power iteration is quite fast, though its convergence rate depends on the *spectral gap* $|\lambda_1| - |\lambda_2|$. In fact, if $|\lambda_1| = |\lambda_2|$, the method may not converge at all. If the spectral gap is bounded by a constant, independent of the number of nodes n , then the power iteration will converge exponentially fast. On the other hand, if $|\lambda_1| - |\lambda_2| = O(1/n^2)$, for instance, then the convergence time will grow as $\Omega(n^2)$.

Side note: $B^T B$ is known as the *co-citation matrix* and $B^T B_{ij}$ counts the number of pages pointing to both i and j

References

- [1] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. Graph structure in the web. In *9th International World Wide Web Conference*, May 2000.
- [2] J. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proc. 9th ACM-SIAM Symposium on Discrete Algorithms*, 1998.