Ranking Queries on the WWW

In the previous lecture we discussed the task of building an index for a large set of documents. This process is often called *Inversion* in the literature, because of the fact that we take a set of documents (this set can be thought of as a function from Documents to sets of Index terms), and construct an Index (which is a function from Index terms to sets of Documents). This is more-or-less inverting the original "function" corresponding to the set of documents.

The setting for the previous lecture was the large-scale environment, where there are very many documents to process. Therefore this setting includes the problem of Indexing the WWW (where documents are webpages), where the "scale" is billions of documents. The two algorithms that we presented for Indexing are not directly applicable to indexing the web, because they are sequential stand-alone algorithms that are executed on a single computer (even if we do use the hard disk for the second algorithm). Given the scale of the WWW, search engines such as Google$^{\text{TM}}$ must maintain large clusters of connected machines which individually perform indexing on their own set of webpages. The Indexing task must be synchronized across all the machines in the cluster, so that we have a (distributed) index for the entire web. This issue of how the indexing task is synchronized across the Google$^{\text{TM}}$ servers involves many concerns, some of which are hardware-related, some algorithmic, some IR-related (IR means Information Retrieval) etc. These cannot be discussed in any significant way in the time available so we leave them. If you are interested in learning something about how a search engine system fits together, have a look at the Brin and Page paper, "An Anatomy . . . " (referenced in Further Reading).

In this lecture we turn our attention to a topic related to Indexing: the topic of *ranking*. Suppose we have constructed an Index for a set of webpages. Now suppose we do a search for "Pease porridge". The index will contain very many webpages that are relevant for this term. How do we rank them in order of importance? Certainly we don't want all the pages. For example, a search using Google$^{\text{TM}}$ with the term "Pease porridge", keeping the quotes to indicate that the words must be adjacent in webpages, the estimate on the first page is 42900 pages (search carried out on 30 December 2008).

One approach to ranking webpages for a particular query might be to count the number of occurrences of the query terms in the webpages (our algorithms for Indexing from Lecture Note 11 actually compute this information and store it in the Index). In practice, this is a bad approach as very informative webpages may contain very few or even no occurrences of the query term(s). For example, if we search for "University of Edinburgh", there are many webpages that have far more more occurrences of this phrase than the university webpage at `http://www.ed.ac.uk`. All the same, the university webpage is first in the ranking (as it should be).

Instead, most ranking systems take a very different approach to the ranking of webpages, by using the *link structure* of the WWW to determine a ranking. The idea is that by linking to another page, the source page confers authority on

the destination page. This is the idea behind PageRank$^{\text{TM}}$ and other models of ranking such as Kleinberg's Hub-Authority model (see Further Reading).

12.1 Basic PageRank$^{\text{TM}}$

Regardless of the model of ranking we use, the ranking of webpages will vary greatly depending with the query. Therefore for this lecture we mostly assume that we are working with a set of webpages relating to one particular query. We could think of this as being the list of pages associated with the query term in the Inverted Index for the query. In practice there will be extra pages in our set of documents, since some relevant webpages may not contain any occurrences of the query term. We sometimes see evidence of this in Google$^{\text{TM}}$ when we see the following message in a cached copy of a page:

```
These terms only appear in links pointing to this page: ...
```

Keeping in mind the principle that a link to a webpage confers authority on that webpage, we have the following model of a ranking system. We consider the web (or the part of it pertaining to this particular query) as a *directed graph*, where the set of *vertices* $V = [N]$ (where $[N]$ is shorthand for $\{1, 2, \ldots, N\}$) is the set of webpages (pertaining to that query) and the *directed edges* $E \subseteq [N] \times [N]$ of the graph are the links of the webgraph. Let $G = (V, E)$ denote our webgraph. Let $M = |E|$, the number of edges. Recall the following definitions (which apply to any graph):

Definition 12.1. Let $u \in V$ be a vertex in the webgraph.

- The *in-degree* $\text{in}(u)$ of u is the number of incoming edges to u (number of links from other pages that point to u). The set of in-edges to u is written as $\text{In}(u)$.
- The *out-degree* $\text{out}(u)$ of u is the number of outgoing edges from u (links that u has to other pages). The set of out-edges from u is written as $\text{Out}(u)$.

Definition 12.2. The *adjacency matrix* of G is the $N \times N$ matrix $A = (a_{ij})_{0 \le i,j \le N-1}$ with

$$a_{ij} = \begin{cases} 1, & \text{if there is an edge from vertex number } i \\ & \text{to vertex number } j; \\ 0, & \text{otherwise.} \end{cases}$$

We are now ready to explain the PageRank$^{\text{TM}}$ model. The idea is, as mentioned above, that a link into a vertex u confers some authority on u. Therefore one possible ranking could be to assign a rank that is directly proportional to the *number of webpages* pointing into u (i.e., proportional to $\text{in}(u)$). That is a reasonable idea except for the fact that we do not consider all links into u to confer the same amount of authority. Instead, the webpages that are themselves ranked highly should be able to confer more authority than lower-ranked webpages. Also, a link from a page with few links on it should be regarded as more significant than a link from a page that has a huge number of links.

In this basic treatment we will make certain assumptions for technical reasons. We assume that all web pages in our system have some outgoing links. This is not an entirely natural assumption (especially as we really have a sub-webgraph obtained by looking at the links between the webpages relevant to our query) but it is crucial to our argument so we make it (for now). We assume that our webgraph is strongly connected—in other words, that for any two pages u and v, there is a sequence of links leading from u to v. With these assumptions, here is PageRank™: Let R_v denote the *rank of* v for any webpage $v \in [N]$. For every webpage u in our collection, the following equality must hold:

$$R_u \;=\; \sum_{v \in \mathrm{In}(u)} R_v / |\mathrm{Out}(v)|$$

So the rank of u is the total amount of Rank given from the incoming links to u. Considering the entire system of ranks of the webgraph, we can actually write this condition using a weighted form of the adjacency matrix of the webgraph:

$$(R_1, R_2, \ldots, R_N) \;=\; (R_1, R_2, \ldots, R_N) \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1N} \\ p_{21} & p_{22} & \cdots & p_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ p_{N1} & p_{N2} & \cdots & p_{NN} \end{pmatrix} \tag{12.1}$$

where

$$p_{uv} \;=\; \begin{cases} 1/\mathrm{out}(u) & \text{if } v \in \mathrm{Out}(u) \\ 0 & \text{otherwise.} \end{cases}$$

Note the following relationship between P and the adjacency matrix A:

$$p_{uv} \;=\; a_{uv}/\mathrm{out}(u),$$

for all $u, v \in V$.

We could also write (12.1) in shorthand as

$$R^T = R^T P, \tag{12.2}$$

where $P = [p_{uv}]_{u,v \in [N]}$ and R is the column vector of ranks for $[N]$ (as usual R^T denotes the transpose of a vector or matrix). Examining (12.2), notice that it is exactly the same as asking for a fixed point of the following type

$$R = P^T R, \tag{12.3}$$

Furthermore, note that $R = P^T R$ is almost the condition for R to be an eigenvector of P^T. The only difference is that the related eigenvalue λ does not seem to appear in the equation (we would expect to have $P^T x = \lambda x$). This is because the ranking R is the eigenvector of P^T *corresponding to the eigenvalue* 1.

This raises quite a few questions, the main questions being:

- How do we know that 1 is an eigenvalue of the matrix P^T?
- If 1 *is* an eigenvalue of P^T, then how do we know that it is a *simple* eigenvalue (i.e., that any two ranking vectors R that satisfy $R^T = R^T P$ are *linearly dependent*—one is a non-zero multiptle of the other)?

These questions are important if the PageRank™ model is to mean anything in the context of our webgraphs.

We start by considering the first question: how do we know that the matrix has an eigenvector of value 1? Have a look at the system of equations (12.1). Consider the particular row of matrix P corresponding to the webpage $u \in [N]$. Then the sum of the values in that column is the sum of fractional rankings that u passes to each of its links:

$$\sum_{v=1}^{N} p_{uv} \;=\; \Big(\sum_{v \in \mathrm{Out}(u)} 1/\mathrm{out}(u)\Big) + \big(\textstyle\sum_{v \notin \mathrm{Out}(u)} 0\big) \;=\; |\mathrm{Out}(u)|/\mathrm{out}(u) + 0 \;=1.$$

So for every row of the matrix P, the entries of that row sum to 1. There is a special name for this sort of matrix in the literature—it is called a *stochastic matrix*, and a stochastic matrix is guaranteed to have the eigenvalue 1 (we omit the proof). So first question answered.

How can we be sure that the matrix cannot have two linearly independent eigenvectors for 1? This depends on our assumptions about the structure of the web graph. We assumed that for every pair of webpages u and v, there was a sequence of links leading from u to v through our webgraph—a graph with this property is known as a *connected graph*. With this assumption (and another tiny assumption, that the series of links from u to v are *aperiodic*[1]), then we can be assured of the uniqueness of the vector R. Again we omit the proof.

12.2 Finding Eigenvectors for eigenvalue 1

We now make an observation about how we can find a ranking R in the PageRank™ model. It is not as difficult as it might seem, particularly because we know that R corresponds to the eigenvector 1. We just have to observe that our conditions for R given in (12.1) simply correspond to a linear system of equations. You will have learnt various methods for solving these in your mathematics classes, and moreover, efficient algorithms for applying these methods do exist (though that's not the approach taken with respect to web ranking—see later). As a very simple example, consider the webgraph of Figure 12.3, perhaps dating back to the early 1990s when there were hardly any webpages on the web.

This mini-graph satisfies all of the conditions that were mentioned in this section, even the condition of aperiodicity (this is obvious because the graph has a cycle of length 2 and a cycle of length 3; so no need to look at any more). Therefore there is a unique (up to constant multiples) eigenvector R satisfying $R^T = R^T P$. To find this particular eigenvector, we write down the matrix P, assuming the webpages are in order u, v, w, z:

$$(R_u, R_v, R_w, R_z) \;=\; (R_u, R_v, R_w, R_z) \begin{pmatrix} 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 \end{pmatrix}. \tag{12.4}$$

[1]This condition requires that if we look at the lengths of all the cycles of the graph then the largest integer that divides *all* of them is 1. Note that if we find a cycle of length 5 (say) and one of length 8 (say) in the graph we know it is aperiodic [why?], an important saving in time.

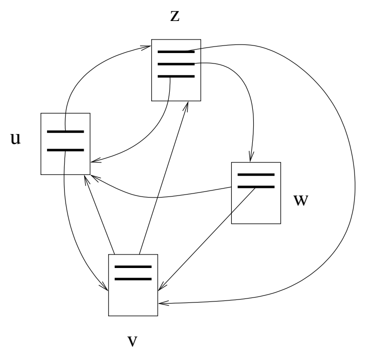

Figure 12.3. An example webgraph returned by a rare query in ancient times.

We can straightaway read off the equality $R_w = R_z/3$. Then we can eliminate R_w by first expressing R_w in terms of R_z on the right-hand side of the equations.

$$(R_u, R_v, R_w, R_z) \quad = \quad (R_u, R_v, R_w, R_z)\begin{pmatrix} 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 \\ \frac{1}{3}+\frac{1}{6} & \frac{1}{3}+\frac{1}{6} & \frac{1}{3} & 0 \end{pmatrix}. \qquad (12.5)$$

Now we can just write $R_w = R_z/3$ to one side and continue with a smaller matrix:

$$(R_u, R_v, R_z) \quad = \quad (R_u, R_v, R_z)\begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & 0 \end{pmatrix}. \qquad (12.6)$$

This is equivalent to:

$$(R_u, R_v - R_z, R_z) \quad = \quad (R_u, R_v, R_z)\begin{pmatrix} 0 & 0 & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & 0 \end{pmatrix}. \qquad (12.7)$$

Observe that the middle equation reads $R_v - R_z = (R_z - R_v)/2$. This means that $R_v = R_z$. Observe that the final equation states that $R_z = (R_u + R_v)/2$. Then with our new knowledge that $R_z = R_v$, this tells us that $R_u = R_z$ also.

An alternative, but equivalent approach, is to multiply out the right hand side of (12.4) to obtain a set of linear equations:

$$\begin{aligned} R_u &= \frac{1}{2}R_v + \frac{1}{2}R_w + \frac{1}{3}R_z \\ R_v &= \frac{1}{2}R_u + \frac{1}{2}R_w + \frac{1}{3}R_z \\ R_w &= \frac{1}{3}R_z \\ R_z &= \frac{1}{2}R_u + \frac{1}{2}R_v. \end{aligned}$$

It is clear that subtracting the second equation from the first would give us useful information, we obtain

$$R_u - R_v = \frac{1}{2}R_v - \frac{1}{2}R_u$$

from which it follows that $R_v = R_u$. Substituting into the fourth equation we obtain $R_z = R_u$. This method is probably preferable for such small examples.

Hence we have a solution $R_u = R_v = R_z$, $R_w = R_z/3$. So if we take any value r and set $R = (r, r, r/3, r)$, we get a solution to (12.4). Note all these solutions are essentially one eigenvector, they are all constant multiples of $(1, 1, 1/3, 1)$. Of course all that a constant multiple does is to change the scale of ranking but it does not alter relative positions. It is a good idea to check that this solution works in (12.4) as an exercise. Note that this solution is not the same as the answer we would get by allocating rank directly according to in-degree.

12.3 PageRank™ in general

In general of course, the web graph will not satisfy the extremely nice conditions that we laid out in the previous subsection (where we want a sequence of links from any page u to any other page v). There is a modified definition of PageRank™ which works in the general case. The intuition behind this ranking system for general web graphs is not quite as elegant as before, however, it allows us always to come up with a ranking regardless of the structure of the webgraph.

We observe first that when there are webpages with no outgoing links, they in some sense "leak" some rank value from the entire web graph. That's because they "take in" rank, but never send any out (corresponding to an all-0's column vector for that page). Hence the eigenvalue that we should consider may be slightly smaller than 1. PageRank™ takes care of this by using the symbol $1/c$ for this eigenvalue, and specifying that it should be as large as possible. The other condition that was required by our basic version of PageRank™ was that there should be a sequence of links from u to v for any pair of pages u and v. We accomplish this by assuming that for every page u, there is a small amount of rank assigned to every page v in the entire system (basically this models the chance that the user might hop to another page at random, without using the links). This means that there is an artificial sequence of links from any page u to any other page v (by co-incidence, this also ensures the overall matrix is aperiodic).

In this more general setting, the PageRank™ model requires that for the minimum value of $c > 1$, every webpage u in our collection, the following equality should be satisfied for every page u:

$$R'(u) \;=\; c^{-1}(1-p)\sum_{v\in \mathrm{In}(u)} R'(v)/|\mathbf{Out}(v)| + c^{-1}(p/N)\mathbf{1}$$

Here p is the (small) fraction of ranking that is leaked uniformly to all pages in the webgraph, $1/c$ is the maximum eigenvalue (for this new system), and **1** is the vector consisting of 1 in every position, i.e., $(1, 1, \ldots, 1)$, the length being deduced from the context.

In this setting we know that, apart from linearly dependent solutions, we have a ranking (for the maximum eigenvalue) for any web graph we consider.

Techniques for solving a linear system apply to this case, though they are a bit more complicated than for the basic (unrealistic) case. In practice, a web crawl (a bit like a *random walk* on the webgraph) is used in constructing the ranking.

12.4 Further Reading

Neither [GT] nor [CLRS] present any material on Algorithms for the WWW. There is also a lot of information online, for example:

- An Anatomy of a Large-Scale Hypertextual Web Search Engine, by Sergey Brin and Lawrence Page, 1998. Online at:

  `http://www-db.stanford.edu/ backrub/google.html`

- The PageRank Citation Ranking: Bringing Order to the Web, by Page, Brin, Motwani and Winograd, 1998. Available online from:

  `http://dbpubs.stanford.edu:8090/pub/1999-66`

- Authoritative Sources in a Hyperlinked Environment, by Jon Kleinberg. Available Online from Jon Kleinberg's webpage:

  `http://www.cs.cornell.edu/home/kleinber/`