Department of Computer Science
The University of Melbourne
COMP90042 WEB SEARCH AND TEXT ANALYSIS (Semester 1, 2017)

Workshop exercises: Week 11

Discussion

1. What are the two components in the **PageRank model** of link analysis? What are the resulting weights used for?
   - Basically, we have a model of a "random web surfer" who, when visiting a page, can either: follow a link (randomly based on the out-links of a given page); or "teleport" to another page (randomly based on all pages).
   - We can then build a Markov process and solve for the equilibrium weights for each page. These weights can then be used as an indication of how "popular" a page is, where we expect that — all else equal — more popular pages should be higher up the ranking.

   (a) Why do we typically use "eigenvalue methods" to calculate PageRank weights?
   - Basically, because we have a big matrix equation, and this is much faster than the iterative method (of estimating the weights, and updating those weights according to the estimated weights of in-links). It is also (typically) subject to less numerical error.

   (b) Given a collection of two documents, where one document (D_1) contains a link to the other document (D_2), find the equilibrium PageRank weights when $\alpha = 0.5$.
   - First, we have a matrix P comprised of **adjancencies** a_{ij}: 1 if document i has an out-link to j, and 0 otherwise. This matrix is normalised so that each row sums to 1 — but what should we do about document 2, which has no outgoing links, so its row sums to 0?
   - There are numerous equivalent formulations to the following transformation: we will assume that documents with no out-going links instead have an out-going link to *every* page, which leads to our initilised P as follows:

P	1	2
1	0	1
2	$\frac{1}{2}$	$\frac{1}{2}$

   - We now scale by $(1-\alpha) = 0.5$, and then add $\frac{\alpha}{N} = \frac{1}{4}$ to every cell:

P	1	2
1	$\frac{1}{4}$	$\frac{3}{4}$
2	$\frac{1}{2}$	$\frac{1}{2}$

   - You might like to confirm that the rows sum to 1, and that the cells for document 2 are logistically consistent.
   - At this point, solving the eigenvalue problem is quite easy (as eigenvalue problems go), but I will demonstrate an iterative formulation, where:
     - We initialise each document with a weight (typically uniform, although other values tend to produce the same results)
     - We estimate a new weight for a given page based on (post–)multiplying the row vector of document weights by the transition matrix.

- Assuming a uniform initialisation of document weights (which is typical, although other values are possible and tend to converge to the same equilibrium) $[0.5, 0.5]$:

$$\begin{aligned}
[0.5, 0.5] \begin{vmatrix} \frac{1}{4} & \frac{3}{4} \\ \frac{1}{2} & \frac{1}{2} \end{vmatrix} &= [0.375, 0.625] \\
[0.375, 0.625] \begin{vmatrix} \frac{1}{4} & \frac{3}{4} \\ \frac{1}{2} & \frac{1}{2} \end{vmatrix} &\approx [0.406, 0.594] \\
[0.406, 0.594] \begin{vmatrix} \frac{1}{4} & \frac{3}{4} \\ \frac{1}{2} & \frac{1}{2} \end{vmatrix} &\approx [0.398, 0.602]
\end{aligned}$$

- As we can see even after just a few iterations, the values are converging to 0.4 for document 1 (with the out-link) and 0.6 for document 2 (with the in-link).

(c) How is the **HITS model** — as described in the lectures — similar to PageRank and how is it different?

- With HITS, instead of modelling a single weight for each document, we introduce the idea of **hub scores** and **authority scores**, according to the following (circular) logic:
  - Good hubs link to good authorities
  - Good authorities are linked from good hubs
- The formal HITS algorithm (from its journal article) sets the initialisation according to the terms in the query, but we have abtracted away from that to simply solving (in this case, two) eigenvalue problems.

2. How can we compress a **postings list** in an inverted index?

- When thinking about the document identifiers, we can get effective compression by keeping the identifiers in order, and then compressing the differences between successive values (usually called d–gaps, or rarely, *deltas*).
- The idea behind the d–gaps is that the postings lists for common terms will consist of many small gaps, and for rare terms, few large gaps. We will then choose a compression scheme that represents small values with few bits (as they are common).
- (A postings list also has term frequencies (or document weights) which we need to compress a little differently.)

(a) What is **Variable Byte Compression** and how does it compress an integer?

- Variable Byte Compression is a variable multi–byte coding format for unsigned integers, where we reserve 1 bit of each byte as a "continuation bit" — which determines whether this byte is the final byte of the integer.
- For example:
  - Values of 0 to 127 require 7 bits (as they are less than 2^8) — in Variable Byte Compression, these appear as 1 byte (7 data bits, plus the continuation bit which indicates that this is the final byte).

  - Values of 128 to 16383 can be represented in Variable Byte Compression as 2 bytes. To do this, we store the 7 low bits in the first byte, with a continuation bit that indicates that more bytes are required; we store the 7 high bits in the second byte, with a continuation bit that indicates this is the final byte.
  - And so on.
- The advantage of this scheme, is that potentially any positive integer can be represented (which is itself an advantage over fixed–width 4- or 8-byte integer coding schemes), but the most common values will be represented in only 1 byte.