CS 591, Guest Lecture

Jared Saia
University of New Mexico

Class Outline

- The *Abundance Problem*
- Link Analysis
- Clever Algorithm
- Clever Algorithm Analysis
- Google Algorithm

The Two Types of Search Queries

Two types of search engine queries

- *Specific Queries*: E.g. “Does Netscape support the JDK 1.1 codesigning API?”
- *General Queries*: E.g. “Find Information about Java”

The Abundance Problem

For General Queries, we have

- **The Abundance Problem: The number of relevant pages is far too large for a human to digest.**

More specifically:

- There are now hundreds of millions of pages on the web
- For most search queries, the number of pages that contain the query words is on the order of thousands
- A person typically wants to look at less than five pages

What we want

One way to handle the abundance problem:

- Filter out from the huge set of relevant pages, those few pages which are most “authoritative” or definitive”
- To do this, we need a notion of what makes a page “authoritative”

An Example Problem

- If we search for all pages containing the word "Microsoft" we get back over a million pages.
- There is nothing about *content* of the page www.microsoft.com that makes it stand out.
  - This page doesn't use the term "microsoft" most frequently.
  - This page doesn't use the term "microsoft" most prominently.
  - There is nothing completely unique in the *textual content* of the page that identifies it as being authoritative.
- What is it that makes the page www.microsoft.com most authoritative?

Link Analysis

- Link Structure is one external measure of authority of a page
- If page x links to page y, x is conferring authority on page y
- There are some potential pitfalls here
  - Links don't always confer authority (e.g. they could be for navigation or to ads)
  - There is a balance between popularity and relevance

The Web as a Graph

- Let each page in the web, be a node of a graph
- Let there be an edge from node x to node y iff there is a link from page x to page y
- This formulation allows us to bring lots of great tools of graph theory to bear on the problem of link analysis.

Popularity and Relevance

- Consider the simple heuristic for returning authoritative pages: of all pages containing the search term, return those which have the greatest number of in-links.
- *This heuristic fails*
  - Many authoritative pages for a search term do not contain that term.
  - Examples: “seach engines”, “automobile manufactures”
  - Many pages which are *not* authoritative for a given term would be incorrectly identified.
  - Examples: www.yahoo.com and www.microsoft.com would be considered authoritative for any search term these pages contained.

The *Clever* Approach

- Key observation: Certain pages point mainly to sites that are authoritative.
- These pages are called good *Hubs*
- Examples:
  - www.yahoo.com
  - "Jared's Big Page Of Links To Cool Sites"
- Question: But then how do we define what makes a page a good Hub?

The *Clever* Approach

- We use a mutually-recursive definition of the notion of Authoritative and Hub-like
- Good *Hub* pages point to many good *Authoritative* pages
- Good *Authoritative* pages are pointed to by many good *Hub* pages

The Clever Algorithm

High Level Idea:

- Clever collects all pages containing the search term, or pages that are neighbors of pages containing the search term.
- It then iteratively propogates "authority" scores and "hub" scores
  - All pages start with equal authority and hub scores. In each step:
  - The pages with hub weight confer authority weight to the pages they point to.
  - The pages with authority weight confer hub weight to the pages *that point to them*

The Clever Algorithm

Algorithm CLEVER:
Input: Graph G (G has n nodes and adjacency matrix A)
Constants: $\epsilon > 0$, $t (= 10)$

1. $\vec{w}_h^0 \leftarrow$ length n all 1's vector, $\vec{w}_a^0 \leftarrow \vec{0}$, $i \leftarrow 0$
2. Repeat
   (a) $i \leftarrow i + 1$
   (b) $\vec{w}_a^i \leftarrow A^T \vec{w}_h^{i-1}$, $\vec{w}_h^i \leftarrow A \vec{w}_a^{i-1}$
   (c) $\vec{w}_h^i \leftarrow \vec{w}_h^i / |\vec{w}_h^i|$, $\vec{w}_a^i \leftarrow \vec{w}_a^i / |\vec{w}_a^i|$
3. Until $|\vec{w}_h^i - \vec{w}_h^{i-1}| \leq \epsilon$ and $|\vec{w}_a^i - \vec{w}_a^{i-1}| \leq \epsilon$
4. (Auth. nodes, Hub nodes) $\leftarrow$ top t weight nodes in $\vec{w}_h^i$,$\vec{w}_a^i$

What does this algorithm compute?

- What exactly does this algorithm compute?
- If E is the set of edges in the link graph, it returns the vectors w_a and w_h such that $|w_a| = |w_h| = 1$ and $\sum_{(s,t)\in E} w_h[s] * w_a[t]$ is maximized
- This is the same as returning the vectors w_a and w_h such that $|w_a| = |w_h| = 1$ and $w_h^T A w_a$ is maximized.

Singular Value Decomposition(SVD)

$$A = \overbrace{\begin{bmatrix} | & & | \\ u_1 & \cdots & u_m \\ | & & | \end{bmatrix}}^{U:m\times m} \overbrace{\begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \end{bmatrix}}^{D:m\times n} \overbrace{\begin{bmatrix} - & v_1 & - \\ & \vdots & \\ - & v_n & - \end{bmatrix}}^{V^T:n\times n}$$

If A is $m \times n$, $n \leq m$ with rank r then $A = UDV^T$ where U, V orthonormal, D diagonal with $\sigma_1, \ldots \sigma_r$ positive and nonincreasing and $\sigma_j = 0\ \forall j > r$.

$\{u_i\}$ called left singular vectors, $\{v_i\}$ called right singular vectors, $\{\sigma_i\}$ called singular values.

SVD is unique up to multiplicities in singular values.

Two Crucial Facts

Let u_i be the i-th column of U and U_k be the $k \times n$ matrix consisting of the first k columns of U. Then:

Fact 1 *For any matrix* A, $max_{|x|=1,|y|=1} x^T Ay = \sigma_1(A)$ *and* $maxarg_{|x|=1,|y|=1} x^T Ay = (u_1, v_1)$.

Fact 2 *The matrix* $A_k = U_k U_k^T A$ *minimizes* $|A - B|$ *over all matrices* B *that have rank* $\leq k$. $|A - A_k| = \sigma_{k+1}(A)$.

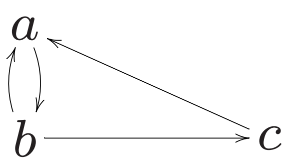
Link Analysis (Kleinberg, 1997)

Input: Link graph with adjacency matrix $A(= UDV^T)$ for pages relevant to a query.
Output: Vectors w_a, w_h giving "authority" and "hub" scores for all pages in the query set. If E is the set of edges in the link graph, gives $|w_a| = |w_h| = 1$ maximizing $\sum_{(s,t)\in E} w_h[s] * w_a[t]$ $(= w_h^T A w_a)$.

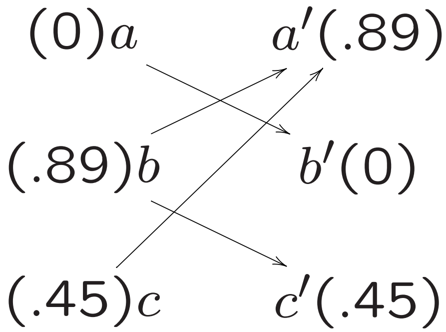
Algorithm: $w_h \leftarrow u_1$, $w_a \leftarrow v_1$
(this is same as limiting vectors of a weight updating algorithm).

Example Weight Assignment

Link Graph



hub *authority*



Google

- Google does something similar to Clever but somewhat simpler.
- Google assigns pages "authorities" based on the first singular vector of a different adjacency matrix.
- This singular vector is related to the stationary probabilities of a random walk on the web.
- Google computes the authority scores offline so it is much faster (but possibly less accurate)