Motori di ricerca per il Web

Caratteristiche dei motori di ricerca – linguaggio di interrogazione

- **Default:**
  - **Default AND**: HotBot, Google, MSN Search, Teoma, WiseNut
  - **Default OR**: Gigablast
  - **Default Phrase Search**: AltaVista Basic, for common phrases only
- **Boolean Capabilities and Constraints:**
  - **and, or, nesting**: AltaVista, Gigablast, HotBot, MSN Search
  - **not**: HotBot, MSN Search
  - **and not**: AltaVista, Gigablast
  - **OR only**: Google, Teoma [both must be OR in uppercase]
  - **implied Boolean (+, -)**: AltaVista Simple, Gigablast
  - **dash - for NOT**: AltaVista Simple, HotBot, Gigablast, Google, MSN Search, Lycos, Teoma, WiseNut

Caratteristiche dei motori di ricerca (2)

- **Proximity**
  - **Phrase Search**: Gigablast, HotBot, Lycos, MSN Search, Teoma, WiseNut
  - **NEAR**: AltaVista
  - **within**: AltaVista Advanced
  - **before**: AltaVista Advanced
  - **before near**: AltaVista Advanced
  - **after**: AltaVista Advanced
  - **after near**: AltaVista Advanced

Caratteristiche dei motori di ricerca (3)

- **Field searching**:
  - **title**: AltaVista, Gigablast, HotBot, Lycos, MSN Search
  - **intitle**: Google, Teoma
  - **url**: AltaVista, Gigablast, Lycos Advanced
  - **inurl**: Google, Teoma
  - **link**: AltaVista, Google, Gigablast, Lycos Advanced, MSN Search
  - **host**: AltaVista (same as site: at others)
  - **ip**: Gigablast
  - **domain**: HotBot, MSN Search
  - **site**: Gigablast, Google, Teoma, (use host: at AltaVista)
  - **anchor**: AltaVista
  - **image**: AltaVista
  - **related**: Google
  - **others**: AltaVista, Gigablast, Google, HotBot, MSN Search

Caratteristiche dei motori di ricerca (4)

- **Limits**:
  - Date (User Specified): AltaVista Advanced, HotBot, Teoma
  - Date (Specific Choices): Google, HotBot, MSN Search, Teoma
  - Language: AltaVista, Google, HotBot, Lycos, MSN Search, Teoma
  - Domain: Gigablast, Google, HotBot, MSN Search, Lycos, Teoma
  - Containing a media type: HotBot, MSN Search
  - Document Directory Depth: MSN Search
  - Page Depth: Gone. Formerly at HotBot
  - File Type:
    - PDFs: AltaVista, Gigablast, Google, MSN Search
    - MS Word (.doc): Gigablast, Google, MSN Search
    - PowerPoint (.ppt): Gigablast, Google, MSN Search
    - Excel (.xls): Gigablast, Google, MSN Search
    - PostScript (.ps): Gigablast, Google
    - WordPerfect (.wpd): Google

Caratteristiche dei motori di ricerca (5)

- **Case Sensitivity**:
  - **Yes**: AltaVista Advanced, AltaVista Simple (se il termine è fra apici)
  - **Unusual mixed case only**: HotBot, MSN Search
  - **No**: Gigablast, Google, Lycos, Teoma, WiseNut
- **Stop Words:**
  - **No Stop Words (all words searched): AltaVista Advanced, Google (se si usa + o se è in una frase), Gigablast, Lycos, Teoma (con +)**
  - **Stop words only if searched alone: HotBot, MSN Search**
  - **Stop words searchable with +: Google, Teoma, WiseNut**
- **Organizzazione dei risultati:**
  - **Sort by relevance: All**
  - **Sort by size: Openfind**
  - **Cluster by site: Google, Gigablast, HotBot, AltaVista, Teoma, WiseNut**

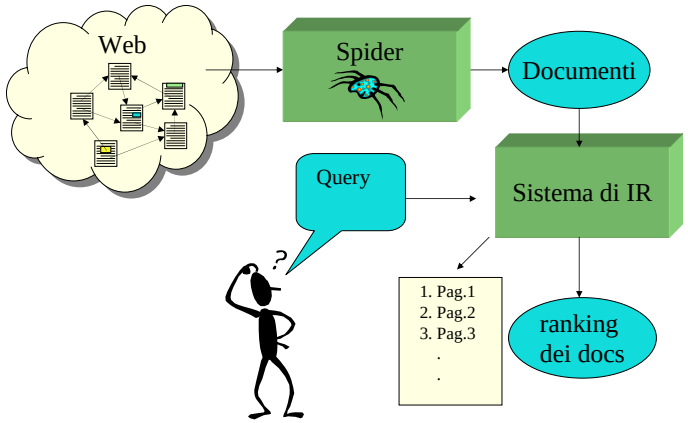
Caratteristiche dei motori di ricerca (6)

Search Engines	Boolean	Default	Proximity	Truncation	Case	Fields	Limits	Stop	Sorting
Google Review	-, OR	and	Phrase, GAPS	No, but stemming, word in phrase	No	intitle, inurl, link, site, more	Language, filetype, date, domain	Varies, + searches	Relevance, site
Yahoo! Review	AND, OR, NOT, ( ), -	and	Phrase	No	No	intitle, url, site, inurl, link, more	Language, file type, date, domain	Very few	Relevance, site
Ask Jeeves/Teoma Review	-, OR	and	Phrase	No	No	intitle, inurl	Language, site, date	Yes, + searches	Relevance, metasites
MSN Search Review	AND, OR, NOT, ( ), -	and	Phrase	No	No	link, site, loc, url	Language, site	Varies, + searches	Relevance,site, sliders
WiseNut Review	- only	and	Phrase	No	No	No	Language	Yes, + searches	Relevance, site
Gigablast Review	AND, OR, AND NOT, ( ), +, -	and	Phrase	No	No	title, site, ip, more	Domain, type	Varies, + searches	Relevance
Exalead Review	AND, OR, NOT, ( ),-	and	Phrase, NEAR	Yes	No	title	Language, file type, date, domain	Varies, + searches	Relevance, date

Web IRS (Web Search Engines)

- Fasi di funzionamento
  - Spidering/Crawling: esplorazione di una porzione del web
  - Indicizzazione: generazione di indici che associano i documenti a dei puntatori, su tre basi:
    - struttura del documento
    - contenuto
    - posizione del documento nel grafo del web
  - Search/Ranking: sulla base della query e degli indici, presentare gli indirizzi delle pagine ordinate per rilevanza

Architettura di un Web IRS



Spiders (Robots/Bots/Crawlers)

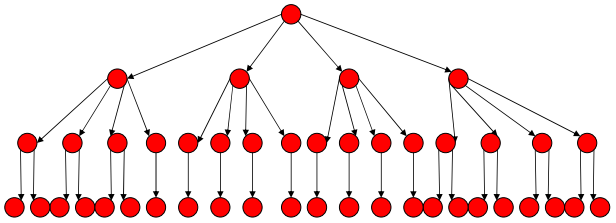
- Programmi che attraversano la rete spedendo pagine nuove o aggiornate ad un server principale (che si occupa di indicizzarle)
- Uno Spider spedisce richieste a server web remoti
  - parte da un insieme pre-identificato di URL "radice"
  - segue tutti i link a partire dalle radici, alla ricerca di pagine addizionali
- Alcuni sistemi consentono all'utente di sottomettere pagine per l'indicizzazione (o stabilire i nodi radice).

Spider Control

- Decide quali link i moduli spider/crawler debbano esplorare e quali ignorare
- Può usare dei feedback provenienti da "usage patterns" (analisi di comportamenti di utente) per guidare il processo di esplorazione

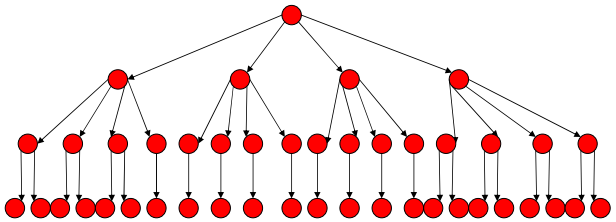
Strategie di spidering

Breadth-first Search



Strategie di spidering (2)

Depth-first Search



Algoritmo di Spidering

Inizializza la lista **Q** con un set iniziale di URL noti

Until Q vuota **o** si esauriscono i limiti di tempo o pagine memorizzabili:

Estrai un URL, L, da Q

Se L non è una pagina HTML (.gif, .jpeg, .ps, .pdf, .ppt…) continua il ciclo

Se L già visitata, continua il ciclo

Scarica la pagina P identificata da L

Se non è possibile scaricare P (e.g. 404 error, robot exclusion) continua ciclo

PROCESSA P (cioè aggiungi P all'indice, o memorizzane una copia cache)

Analizza P per ottenere una nuova lista di links L_P

Appendi L_P in coda a Q

Strategie di Spidering: due tipi di restrizioni

- Spidering specializzato
  - Location-driven: considera solo siti/directory specifici
    - Rimuove i links non contenuti nei siti/directory indicati
  - Metriche di importanza:
    - Interest, Popularity
- Alcuni siti Web o pagine possono specificare che gli spiders/robot non accedano né indicizzino certe aree
  - Robots Exclusion Protocol: E' un protocollo che vieta l'accesso da directory specificate all'intero sito
  - Robots META Tag: Etichetta documenti specifici per evitarne l'indicizzazione o l'esplorazione.

2. Indicizzazione

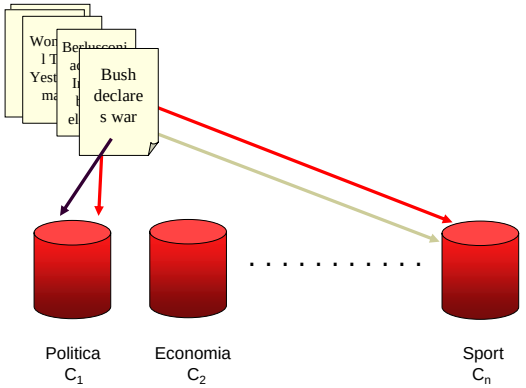
Modalità per indicizzare le pagine sul Web

- Per ogni pagina identificata dallo Spider va creato un riferimento
- Tre metodi (non esclusivi):
  - Si usano directory, che classificano le pagine Web per argomento
  - Si indicizza ogni documento esplorato dallo Spider come full text (*flat* o strutturato)
  - Si tiene conto della struttura ipertestuale del Web, cioè degli hyperlinks

Classificazione automatica

- Perché:
  - La classificazione manuale dei documenti sul web ad opera dei gestori di portali o motori di ricerca richiede molto tempo e molta forza-lavoro, è soggettiva, e soggetta a errori.
  - I metodi automatici di classificazione dei testi possono essere usati per alleviare questo lavoro.
- Questi metodi si basano su tecniche di machine learning.
- Classificazione tramite tassonomie
  - Lo sviluppo di tassonomie è un processo complesso, richiede tempo, è soggettivo, produce errori.
  - Esistono metodi automatici (Hierarchical Agglomerative Clustering (HAC)) per ottenere automaticamente una strutturazione gerarchica.

Metodi di classificazione automatica



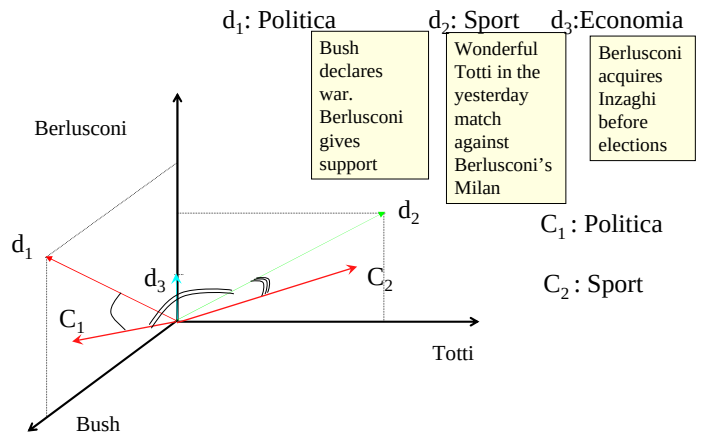
Classificazione Automatica

- Dato:
  - Un insieme di categorie: $C = \{ C^1,..,C^n \}$
  - Un insieme *T* di documenti *d*,
- … scopri una funzione

$$f : T \rightarrow 2^C$$

- VSM (Vector Space Model)
  - I documenti sono rappresentati come vettori (modello vettoriale).
  - Sia i documenti che le categorie sono vettori.
  - *d* è assegnato a C^i se $\vec{d} \times \vec{C}^i > th$

Esempio



Creazione automatica di indici

- Molti sistemi usano varianti dell'inverted file
  - Ad ogni parola o keyword viene associato l'insieme dei puntatori alle pagine che contengono la parola
  - Eliminazione di stopwords, punteggiatura..
  - Usualmente si associa anche una breve descrizione del contenuto della pagina (per mostrarlo all'utente)
- Grandi volumi
  - Poiché occorrono circa 500 bytes per ogni pagina, occorrono 50 Gb per 100 milioni di pagine !!
  - Tecniche di compressione per ridurre un inverted file di circa il 30% (dunque bastano 15 Gb per 100 milioni di pagine)

Creazione automatica di Indici

- Alcuni inverted files puntano all'occorrenza di una parola nel testo,
  - è possibile fare "proximity search" cioè cercare frasi
  - ma la tecnica è troppo costosa in termini di memoria.
- Qualche motore di ricerca consente proximity search, ma le tecniche non sono note interamente.
- Una soluzione intermedia consiste nel puntare a blocchi anziché a singole occorrenze nel documento.
  - Questo riduce la taglia dei puntatori e velocizza la ricerca.

Metodi avanzati: Indicizzazione usando il testo nelle ancore

- Lo Spider analizza il testo nelle ancore (between <a> and) di ogni link seguito.
- Il testo nelle ancore di solito è molto informativo sul contenuto del documento cui punta.
- Si aggiunge il testo contenuto nell'ancora alla vista della pagina destinazione (keywords)
- Questo metodo è usato da Google:
  - Evil Empire
  - IBM

Indicizzazione usando il testo nelle ancore (cont)

- E' particolarmente utile quando il contenuto descrittivo nelle pagine destinazione è "nascosto" nei logo delle immagini piuttosto che in testi accessibili.
- Spesso non è affatto utile:
  - "click here"
- Migliora la descrizione del contenuto di pagine "poplari" con molti link entranti, ed aumenta la recall di queste pagine.
- Si può anche assegnare un peso maggiore al testo contenuto nelle ancore.

Searching e Ranking

- I metodi di ranking usati dai motori di ricerca sono top-secret
- I metodi più usati sono il modello vettoriale e booleano
- Alcuni nuovi motori utilizzano per il ranking anche gli hyperlinks.
  - Le pagine "risposta" *p* vengono valutate (ed il loro numero viene esteso) sulla base del numero di connessioni da e verso *p*.

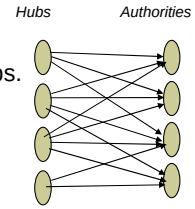
Link Analysis

Primi metodi di link analysis

- Gli Hyperlinks contengono informazioni che sussumono un giudizio sulla pagina
- Più links entrano in un sito, più importante è il giudizio
- La visibilità di un sito si misura mediante il numero di siti che puntano ad esso
- La luminosità di un sito è il numero di altri siti che esso punta

HITS

- Hubs e authorities
  - Un sito è autorevole se riceve molte citazioni. Le citazioni da parte di siti importanti pesano di più di quelle da parte di siti meno importanti.
  - Un buon hub è un sito che è collegato con molti siti autorevoli
- Grafo bipartito:
  - Gli Hub puntano a molte authorities.
  - Le Authorities sono puntate da molti hubs.

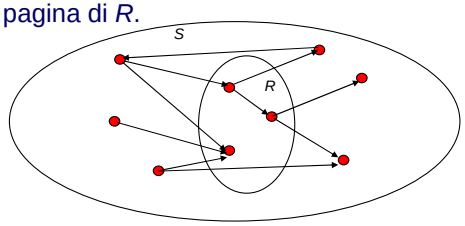


Algoritmo HITS

- Calcola gli hubs e le authorities per un particolare argomento (dominio informativo)
- 2 Fasi:
  1. Prima determina un sottografo (*base*), corrispondente ad un insieme di pagine rilevanti
  2. Analizza la struttura dei link del sottografo corrispondente a *S* per identificare quali pagine di *S* sono authorities e quali sono hubs.

Costruzione del sottografo base S

- Dato un particolare argomento, sia R un insieme di pagine rilevanti (detto *insieme radice*).
- Inizializza i nodi di *S* con tutte le pagine *R*.
- Aggiungi ai nodi di *S* tutte le pagine puntate da qualche pagina di *R*.
- Aggiungi a *S* tutte le pagine che puntano a qualche pagina di *R*.



Authorities and In-Degree

- Even within the base set *S* for a given query, the nodes with highest in-degree are not necessarily authorities (may just be generally popular pages like Yahoo or Amazon).
- True authority pages are pointed to by a number of hubs (i.e. pages that point to lots of authorities).

Iterative Algorithm

- Use an iterative algorithm to slowly converge on a mutually reinforcing set of hubs and authorities.
- Maintain for each page $p \in S$:
  - Authority score: a_p (vector $\boldsymbol{a}$)
  - Hub score: h_p (vector $\boldsymbol{h}$)
- Initialize all $a_p = h_p = 1$
- Maintain normalized scores:

$$\sum_{p \in S} (a_p)^2 = 1 \qquad \sum_{p \in S} (h_p)^2 = 1$$

HITS Update Rules

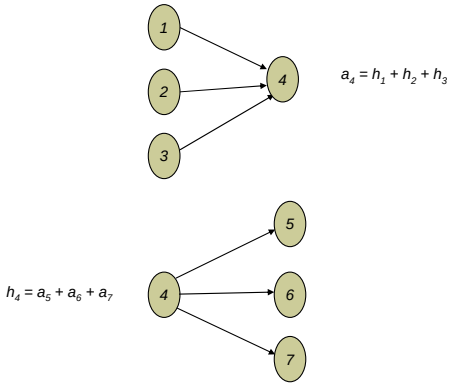
- Authorities are pointed to by lots of good hubs:

$$a_p = \sum_{q:q \to p} h_q$$

- Hubs point to lots of good authorities:

$$h_p = \sum_{q:p \to q} a_q$$

Illustrated Update Rules



HITS Iterative Algorithm

Initialize for all $p \in S$: $a_p = h_p = 1$

For i = 1 to k:

For all $p \in S$: $a_p = \sum_{q:q \to p} h_q$ (*update auth. scores*)

For all $p \in S$: $h_p = \sum_{q:p \to q} a_q$ (*update hub scores*)

For all $p \in S$: $a_p = a_p / c$ c: $\sum_{p \in S} (a_p / c)^2 = 1$ (*normalize* $\boldsymbol{a}$)

For all $p \in S$: $h_p = h_p / c$ c: $\sum_{p \in S} (h_p / c)^2 = 1$ (*normalize* $\boldsymbol{h}$)

Algorithm converges to a *fix-point* if iterated indefinitely.
In practice, 20 iterations produces fairly stable results.

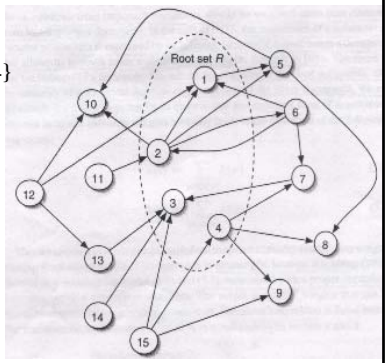
Esempio di HITS

Trova un sottografo di partenza:

• Inizia con il set di radici R {1, 2, 3, 4}

• {1, 2, 3, 4} - nodi rilevanti per l'argomento (query)

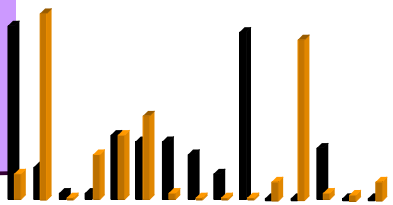
• Espandi il set di radici R includendo tutti i figli e un numero fisso di nodi genitori



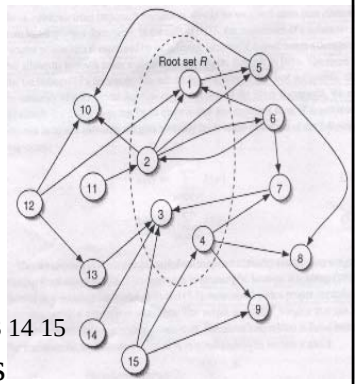
→ Un nuovo set S (sottografo base) →

Risultati di HITS

Authority
Hubness



Pesi di authority e hubness



Results

- Authorities for query: "Java"
  - java.sun.com
  - comp.lang.java FAQ
- Authorities for query "search engine"
  - Yahoo.com
  - Excite.com
  - Lycos.com
  - Altavista.com
- Authorities for query "Gates"
  - Microsoft.com
  - roadahead.com

Finding Similar Pages Using Link Structure

- Given a page, *P*, let *R* (the root set) be *t* (e.g. 200) pages that point to *P*.
- Grow a base set *S* from *R*.
- Run HITS on *S*.
- Return the best authorities in *S* as the best similar-pages for *P*.
- Finds authorities in the "link neighbor-hood" of *P*.

Similar Page Results

- Given “honda.com”
  - toyota.com
  - ford.com
  - bmwusa.com
  - saturncars.com
  - nissanmotors.com
  - audi.com
  - volvocars.com

PageRank

- Alternative link-analysis method used by Google (Brin & Page, 1998).
- Does not attempt to capture the distinction between hubs and authorities.
- Ranks pages just by authority.
- Applied to the entire web rather than a local neighborhood of pages

Initial PageRank Idea

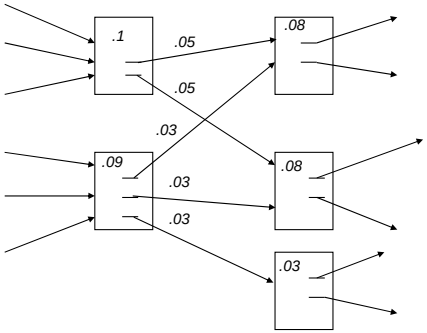
- Just measuring in-degree (citation count) doesn’t account for the authority of the source of a link.
- Initial page rank equation for page *p*:

$$R(p) = c \sum_{q:q \to p} \frac{R(q)}{N_q}$$

  - N_q is the total number of out-links from page *q*.
  - A page, *q*, “gives” an equal fraction of its authority to all the pages it points to (e.g. *p*).
  - *c* is a normalizing constant set so that the rank of all pages always sums to 1.

Initial PageRank Idea (cont.)

- Can view it as a process of PageRank “flowing” from pages to the pages they cite.



Initial Algorithm

- Iterate rank-flowing process until convergence:

  Let *S* be the total set of pages.

  Initialize $\forall p \in S$: $R(p) = 1/|S|$

  Until ranks do not change (much) *(convergence)*

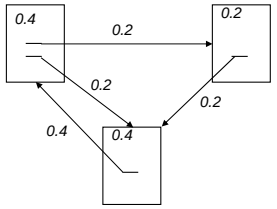
  For each $p \in S$:

$$R'(p) = \sum_{q:q \to p} \frac{R(q)}{N_q}$$

$$c = 1 / \sum_{p \in S} R'(p)$$

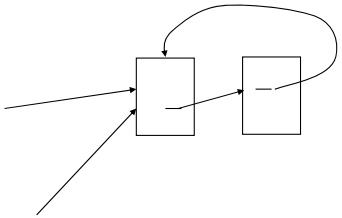
  For each $p \in S$: $R(p) = cR'(p)$ *(normalize)*

Sample Stable Fixpoint



Problem with Initial Idea

- A group of pages that only point to themselves but are pointed to by other pages act as a "rank sink" and absorb all the rank in the system.



Rank flows into cycle and can't get out

Rank Source

- Introduce a "rank source" *E* that continually replenishes the rank of each page, *p*, by a fixed amount *E*(*p*).

$$R(p) = c\left(\sum_{q:q \to p} \frac{R(q)}{N_q} + E(p)\right)$$

PageRank Algorithm

Let S be the total set of pages.

Let $\forall p \in S$: $E(p) = \alpha/|S|$ *(for some* $0<\alpha<1$*, e.g. 0.15)*

Initialize $\forall p \in S$: $R(p) = 1/|S|$

Until ranks do not change (much) (convergence)

For each $p \in S$:

$$R'(p) = \sum_{q:q \to p} \frac{R(q)}{N_q} + E(p)$$

$$c = 1 / \sum_{p \in S} R'(p)$$

For each $p \in S$: $R(p) = cR'(p)$ *(normalize)*

Random Surfer Model

- PageRank can be seen as modeling a "random surfer" that starts on a random page and then at each point:
  - With probability $E(p)$ randomly jumps to page p.
  - Otherwise, randomly follows a link on the current page.
- $R(p)$ models the probability that this random surfer will be on page p at any given time.
- "E jumps" are needed to prevent the random surfer from getting "trapped" in web sinks with no outgoing links.

Speed of Convergence

- Early experiments on Google used 322 million links.
- PageRank algorithm converged (within small tolerance) in about 52 iterations.
- Number of iterations required for convergence is empirically O(log n) (where n is the number of links).
- Therefore calculation is quite efficient.

Simple Title Search with PageRank

- Use simple Boolean search to search web-page titles and rank the retrieved pages by their PageRank.
- Sample search for "university":
  - Altavista returned a random set of pages with "university" in the title (seemed to prefer short URLs).
  - Primitive Google returned the home pages of top universities.

Google Ranking

- Complete Google ranking includes (based on university publications prior to commercialization).
  - Vector-space similarity component.
  - Keyword proximity component.
  - HTML-tag weight component (e.g. title preference).
  - PageRank component.
- Details of current commercial ranking functions are trade secrets.