Chapter 13

Searching the Web

Introduction

- The Web can be treated as a very large, unstructured, global information system, but it is not a DBMS
  - Web documents: text, audio, video, and images
- There is a demand on efficient tools to *manage*, *retrieve* and *filter* information from the Web
- Approaches
  - Search Web documents using user-specified words/ patterns in a text
    - search engines: *index* a portion of the Web documents
    - Web directories: classify Web documents by *subjects*
    - hyperlinks: create pointers from one Web page to another

Challenges

- Distributed network systems: data spans over computers, interconnected with *no* predefined topology, *unreliable* network (bandwidth varies)
- High percentage of volatile data: *dynamic* data, *dangling* links, and data *relocation*
- Hugh volume: *exponential growth* rate; difficult to cope with
- Unstructured and redundant data: Web text docs are *semi-structured*/*unstructured* in nature and *replicated* (~30%)
- Quality of data: no editorial process on Web data which can be *invalid*, *poorly written* and *erroneous*
- Heterogeneous data: multiple media types (formats) with different languages

Problems Posted by the Web

- Most of the problems, such as diversity of data types and *poor data quality*, are unsolvable
- Problems on the interaction with the retrieval system:
  - How to specify a query?
  - How to interpret the answers provided by the system?
    - How do we handle a *large* (a 1,000 documents) answer?
    - How do we rank the documents?
    - How do we select documents that really are of interest to the user?
    - How do we browse efficiently in large retrieved documents?
- Design goal: (i) formulate "good" *queries*, and (ii) obtain a manageable and relevant *answer*

Ranking Algorithms

- Yuwono and Lee (1997) propose 4 ranking algorithms:
  1) Boolean Spreading Activation: *Classical Boolean* + "simplified link analysis"
  2) Most-cited: based only on the terms included in pages having a *link* to the pages in the answer set
  3) TFxIDF: based on *word distribution statistics*
  4) Vector Spread Activation: *Vector space* model + spread activation model; relatively superior, 76% (precision)
- (1) and (2): rely on WWW meta-information (i.e., hyperlink structure) without considering term frequencies
- (3): uses only *word occurrence* statistics
- (4): uses both *word occurrence* statistics & *hyperlink* structures

Ranking Algorithms

Terminology and Notations

M: the number of *query words*.

N: the number of Web *documents* in the collection.

q_j: the j^{th} query word, $1 \le j \le M$.

d_i: the i^{th} Web document, $1 \le i \le N$.

$R_{i,q}$: the *relevance score* of d_i with respect to query q.

$C_{i,j}$: *occurrence* of q_j in d_i, where $C_{i,j} = \begin{cases} 1 & \text{if } d_i \text{ contains } q_j \\ 0 & \text{otherwise.} \end{cases}$

$L_{i_{i,k}}$: the o*ccurrence* of an *incoming hyperlink* from d_k to d_i, where

$$L_{i_{i,k}} = \begin{cases} 1 & \text{if such a hyperlink exists} \\ 0 & \text{otherwise.} \end{cases}$$

$L_{o_{i,k}}$: the *occurrence* of an *outgoing hyperlink* from d_i to d_k, where

$$L_{o_{i,k}} = \begin{cases} 1 & \text{if such a hyperlink exists} \\ 0 & \text{otherwise.} \end{cases}$$

Boolean Spreading Activation Ranking Algorithm

- BSA, which is based on an extended Boolean model: docs are ranked according to the occurrence/absence of query words w/o considering '$\vee$' and '$\neg$', which can be *simulated*
- *Propagates* the occurrence of a query word in a document to its (incoming/outgoing) linked documents
- An assumption is made that two linked documents are *related semantically*, and

If doc d_i is linked to/from doc d_k, which contains q_j, then d_i contains q_j

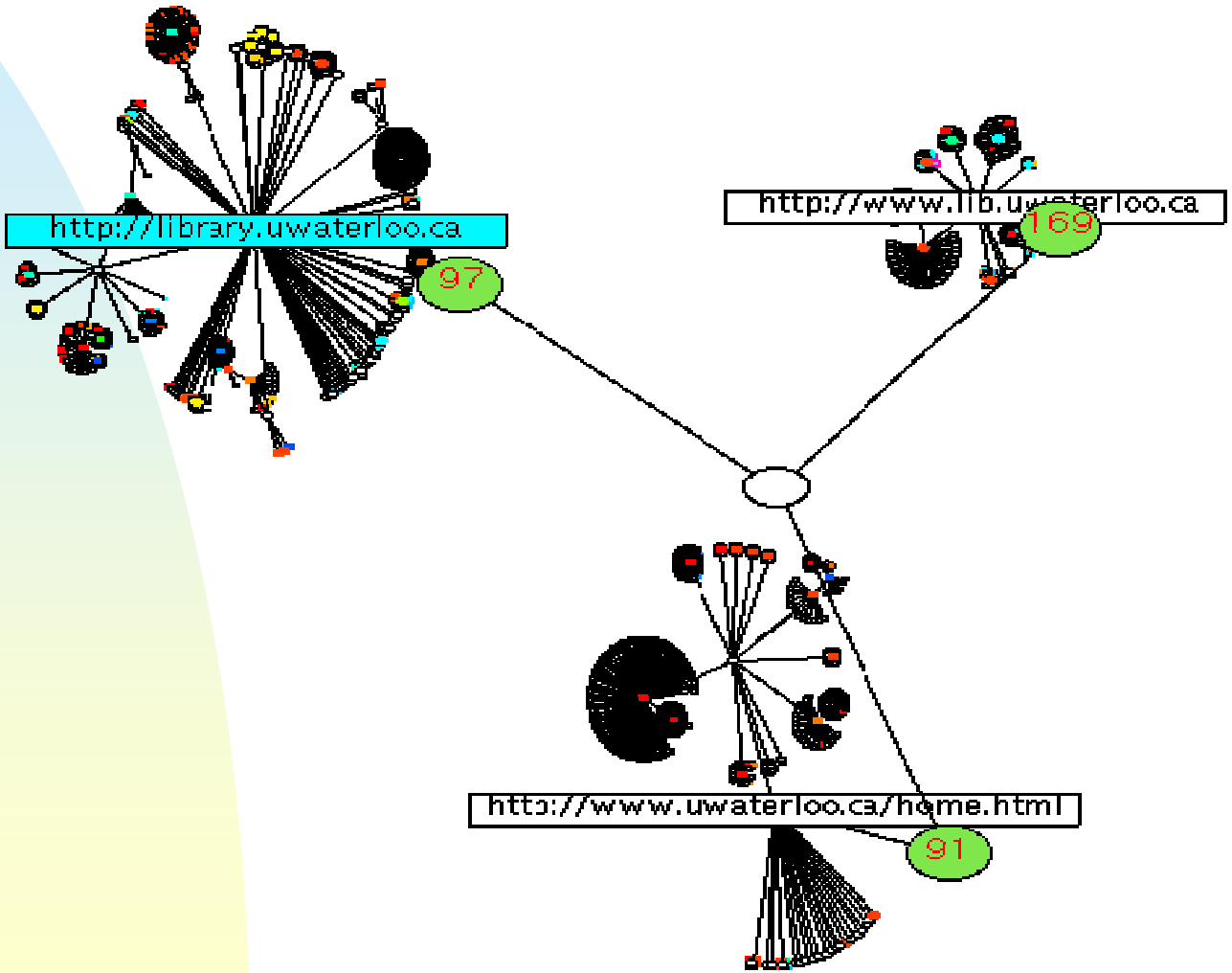
- Doument *i* is assigned a relevance score, $R_{i,q}$, with respect to query q, where

$$R_{i,q} = \sum_{j=1}^{M} I_{i,j} \text{ , and } C_1 > C_2 > 0 \text{ (e.g., } C_1 = 10 \text{ and } C_2 = 1) \text{ such that}$$

$$I_{i,j} = \begin{cases} C_1 & \text{if } C_{i,j} = 1 \\ C_2 & \text{if } \exists k \text{ such that } C_{k,j} = 1 \text{ and } L_{i_{i,k}} + L_{o_{i,k}} > 0 \\ 0 & \text{otherwise} \end{cases}$$

Ranking

- WebQuery



Most-Cited Ranking Algorithm

- Takes advantage of information about hyperlinks between Web docs, as with the BSA algorithm
- Doument *i* is assigned a relevance score, $R_{i,q}$, with respect to query q, which is the sum of the number of query words in q contained in the (incoming) linked docs

$$R_{i,q} = \sum_{k=1,\, k \neq i}^{N} (L_{i_{i,k}} \times \sum_{j=1}^{M} C_{k,j})$$

- Objective: assign larger scores to, among other potentially relevant documents, the referenced documents

TF×*IDF* Ranking Algorithm

- Based on the well-known VSM
- Vector length *normalization* is applied in computing the relevance score, $R_{i,q}$, of document i with respect to query q,

$$R_{i,q} = \frac{\sum_{term_j \in q} (0.5 + [0.5 \times freq(i, j) / max(freq(i, l))]) \times log(N / \sum_{i=1}^{N} C_{i,j})}{\sqrt{\sum_{term_j \in i} (0.5 + [0.5 \times freq(i, j) / max(freq(i, l))])^2 \times log(N / \sum_{i=1}^{N} C_{i,j})^2}}$$

- The *normalized VSM* is very expensive to implement as the *normalized factor* is very expensive to compute
- The relevant score, $R_{i,q}$, of *TF*×*IDF* algorithm is

$$\sum_{term_j \in q} (0.5 + [0.5 \times freq(i, j) / max(freq(i, l))]) \times log(N / \sum_{i=1}^{N} C_{i,j})$$

Vector Spreading Activation Ranking Algorithm

- Combines the VSM (using word frequency) and Most-Cited (using hyperlink structure) in ranking
- Approach:
  1) doument *i* is first assigned a relevant score using *TF*×*IDF* algorithm
  2) the score of *i* is propagated to the documents that cite document *i*
- $R_{i,q}$, the relevant score of document i with respect to query q:

$$R_{i,q} = S_{i,q} + \sum_{j=1, j \neq i}^{N} \alpha \times L_{i_{i,j}} \times S_{j,q}$$

where

$S_{i,q}$ is the *TF*×*IDF* score of document *i*, and

α ($0 < \alpha < 1$) is a *constant link weight* (optimal $\alpha = 0.2$)

Recall & Precision for Ranking Algorithms

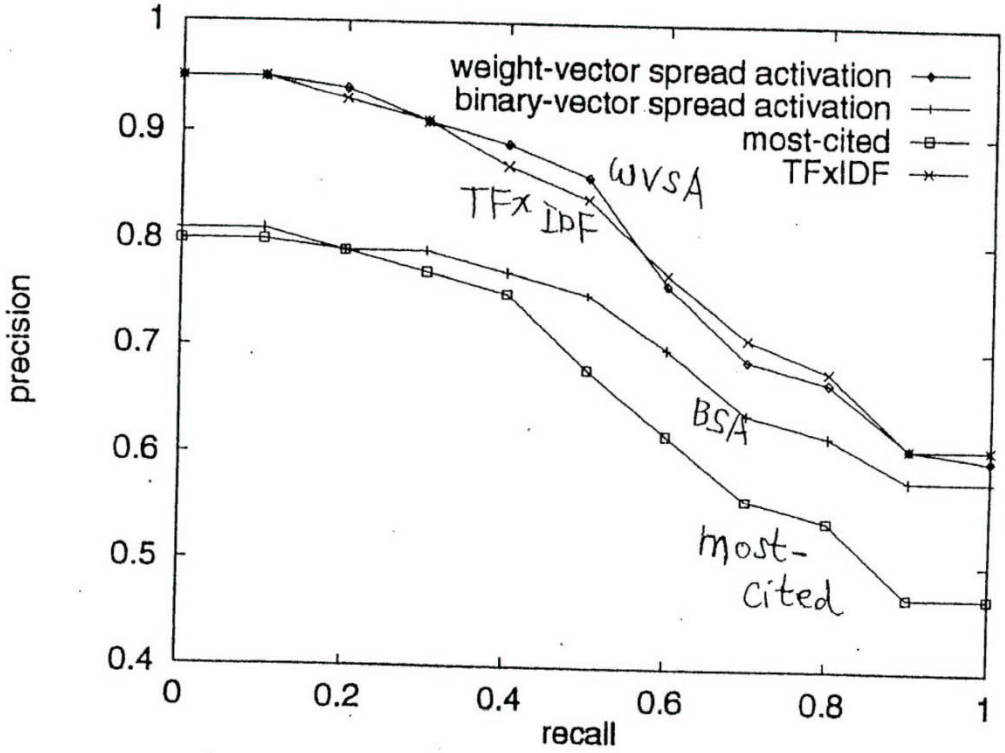


Fig. 3. The recall/precision curves of the four document ranking algorithms obtained by averaging the curves over the 56 test queries. over 2,393 URLs

Recall & Precision for Ranking Algorithms

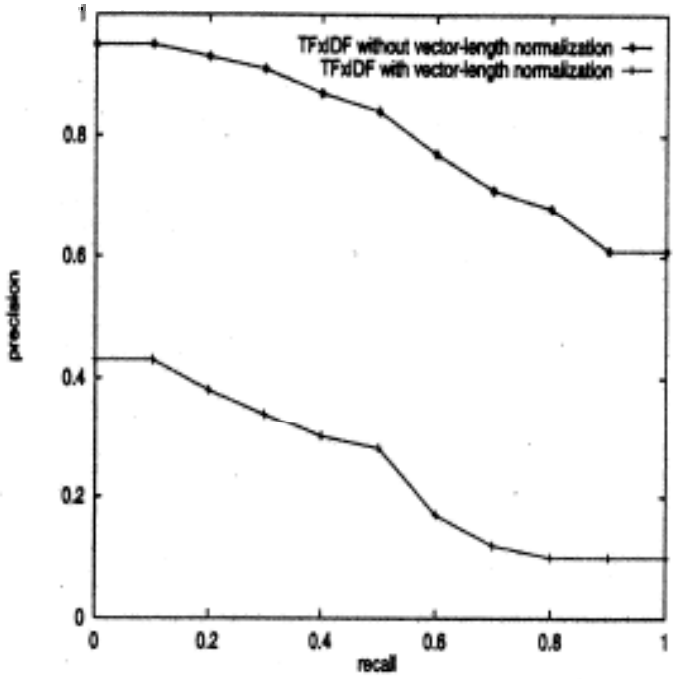


Figure 4: Recall-precision curve for TFxIDF with and without vector-length normalization obtained by averaging the curves over the 56 test queries.

- TFxIDF performs significantly better without *vector-length normalization*
- *Vector length normalization* does not work well for *short* documents
- In a Web page, a topic is often only represented by a hypertext archor

Recall & Precision for Ranking Algorithms

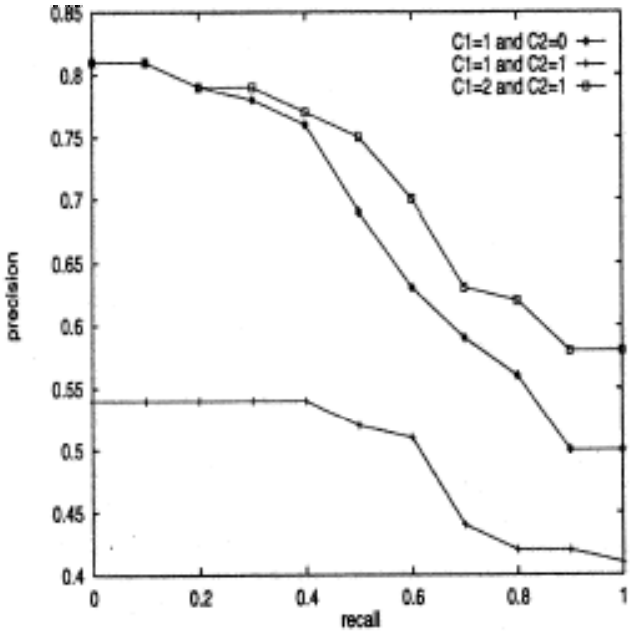


Figure 5: Recall-precision curves of Boolean Spreading Activation algorithm on the 56 test queries with (c_1, c_2) value pairs of (1,0), (2,1) and (1,1).

- The recall ratio of BSA is improved by setting $C_1 > C_2 > 0$, i.e., $C_1 = 2$ & $C_2 = 1$, or (5, 1), (10, 1), or (10, 5).

- *Poor* retrieval performance when $C_1 = C_2$, i.e., when many pages with various degrees of actual relevance is given the same relevance socre

Recall & Precision for Ranking Algorithms

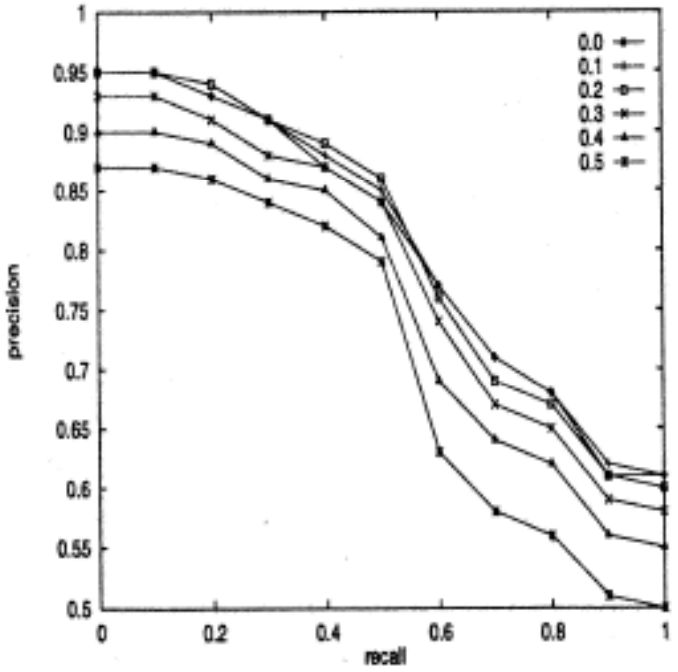


Figure 6: Recall-precision curves of Vector Spreading Activation algorithm on the 56 test queries with α value of 0.0, 0.1, 0.2, 0.3, 0.4 and 0.5.

- The best retrieval performance for WVSM occurs when $\alpha = 0.2$
- When $\alpha = 0$, WVSM becomes TFxIDF

Search Engines

- Estimating the Usefulness of Search Engines (Meng'99)
  - Propose a statistical method to estimate the usefulness of a search engine for any given query
    - *usefulness*: defined to be a combination of the number of similar documents to the query and their averages
    - *database*: defines the set of documents that can be searched by the search engine
    - *index*: created and stored in the search engine for all documents in the DB to speed up query processing
  - Recognize the shortcomings of a single search engine:
    - its processing power and storage capability may not scale to the virtually unlimited amount of data
    - gathering all data on the Web and keeping them *up-to-date* are extremely difficult, if not impossible

Search Engines

- Solution: a 2-level approach to provide search services

  1) bottom level: local search engines
  2) top level: a meta search engine – an interface that does not maintain its own *index* on documents (but contain *characteristic information*). Upon receiving a query
     a) first passes the *query* to selected local search engines
     b) then *collects* the *results* from the search engines used

  Advantages:

  - user queries can be *evaluated* against smaller DBs in *parallel*
  - *updates* to indexes can be *localized*
  - *local information* can be gathered more *easily* and *quickly*
  - *storage space* and *processing power* @ each local search engine is more *manageable*

Search Engines

- A meta search engine should identify local search engines that can provide *useful* results to a given query *q*; o.w., resources are wasted. Usefulness is measured by
  - ***NoDoc***: number of docs in db *D* of the search engine whose similarities w/ *q* are higher than a specified *threshold*, i.e.,

$$NoDoc(T, q, D) = |\{ d \mid d \in D \wedge sim(q, d) > T \}|, \text{ and}$$

  - ***AvgSim***: the average similarity of "high"-similarity documents

$$AvgSim(T, q, D) = \frac{\sum_{d \in D \wedge sim(q, d) > T} sim(q, d)}{NoDoc(T, q, D)}$$

where *q* and *d* are sets of (key)words transformed into vectors with *weights*, and

sim(*q*, *d*) is the s*imilarity* measured by the dot product of *q* and *d*

Search Engines

- The subrange-based Method
  - Let m be the number of distinct terms in a db D
  - Let $d \in D$ be represented as a vector $d = <d_1, \ldots, d_m>$, where d_i $(1 \le i \le m)$ is the weight of term $t_i \in d$
  - Let q be a query such that $q = <u_1, \ldots, u_m>$, where u_i $(1 \le i \le m)$ is the weight of term $t_i \in q$
  - The similarity between q and d is defined as

$$sim(q, d) = u_1 \times d_1 + \ldots + u_m \times d_m$$

  - Let D be represented as m pairs $\{(p_1, w_1), ..., (p_m, w_m)\}$, where
    - p_i: probability that term $t_i \in d \in D$
    - w_i: average weight of all terms $t_i \in S \subseteq D$
    - $q' = <u_1, \ldots, u_r>$, where q' is q .э. $0 < r \le m$
  - Then, the similarity between q' and d is defined as

$$\text{sim}(q', d) = u_1 \times d_1 + \ldots + u_r \times d_r$$

Search Engines

- Let m be the number of distinct terms in a db D
- Let p_i be the probability that term $t_i \in d \in D$
- Let w_i be the average weight of all terms $t_i \in S \subseteq D$
- Let q be $<u_1, \ldots, u_r>$ and u_i $(0 < i \leq r \leq m)$ is the weight of t_i
- Let X be a *dummy variable*
- Then, the generating function with the *probabilities* that docs in D have certain similarities w/ q is defined as

$$(p_1 \times X^{w_1 \times u_1} + (1 - p_1)) \times (p_2 \times X^{w_2 \times u_2} + (1 - p_2)) \times \ldots \times (p_r \times X^{w_r \times u_r} + (1 - p_r))$$

Proposition. If t_i s are *independent*, then the coefficient of X^s is the probability that d has similarity s with q

Search Engines

- Example. Given the generating function below with the *probabilities* that docs in *D* have certain similarities with *q*

$$(p_1 X^{w_1 \times u_1} + (1 - p_1)) \times (p_2 X^{w_2 \times u_2} + (1 - p_2)) \times \ldots \times (p_r X^{w_r \times u_r} + (1 - p_r))$$

  - Suppose the *term weights* of $q = \langle u_1, u_2, u_3 \rangle = \langle 1, 1, 1 \rangle$
  - Let *D* contain 5 documents with their *vector representations* as

    $\langle 3, 0, 0 \rangle, \langle 1, 1, 0 \rangle, \langle 0, 0, 2 \rangle, \langle 2, 0, 2 \rangle, \langle 0, 0, 0 \rangle$

    Hence,

    $(p_1, w_1) = (0.6, 2), (p_2, w_2) = (0.2, 1), (p_3, w_3) = (0.4, 2)$

    Therefore,

    $$(0.6 \times X^2 + 0.4) \times (0.2 \times X + 0.8) \times (0.4 \times X^2 + 0.6)$$

  - The *coefficient* of X^2 is

$$\left.\begin{array}{l} p_1 \times (1 - p_2) \times (1 - p_3) = 0.6 \times 0.8 \times 0.6 \ (= 0.288) \\ + (1 - p_1) \times (1 - p_2) \times p_3 = 0.4 \times 0.8 \times 0.4 \ (= 0.128) \end{array}\right\} = \underline{0.416}$$

  i.e., the estimated probability of any $d \in D$ *has similarity* 2 with q

Search Engines

- After expanding and merging terms w/ the same X^s the generating function becomes

$$a_1 \times X^{b_1} + a_2 \times X^{b_2} + \ldots + a_c \times X^{b_c}, \text{ where } b_1 > b_2 > \ldots > b_c$$

- By the proposition, a_i is the probability that a documents in D has similarity b_i with query q
- If D contains n documents, then $n \times a_i$ is the expected no. of documents in D that have similarity b_i with q
- Let T be a given similarity threshold, and let C be the largest number to satisfy $b_C > T$. Then, the expected number of documents whose *similarity* (w/ q) > T is

$$\sum_{i=1}^{C} (n \times a_i)$$

Search Engines

- The *NoDoc* measure of *D* for query *q* based on the threshold value *T* can be estimated as

$$est\_NoDoc(T, q, D) = \sum_{i=1}^{C} (n \times a_i) = n \times \sum_{i=1}^{C} a_i$$

Since $n \times a_i \times b_i$ is the expected *sum* of all similarities of documents whose similarities w/ *q* are b_i, thus

$$\sum_{i=1}^{C} (n \times a_i \times b_i)$$

is the expected sum of all similarities of documents whose similarities (w/ *q*) > *T*.

- The *AvgSim* measure of *D* for *q* based on threshold value *T* can be estimated as

$$est\_AvgSim(T, q, D) = \frac{n \times \sum_{i=1}^{C} (a_i \times b_i)}{n \times \sum_{i=1}^{C} a_i} = \frac{\sum_{i=1}^{C} (a_i \times b_i)}{\sum_{i=1}^{C} a_i}$$

Search Engines

- Example.
  - Assume that the *term weights* of $q = \langle u_1, u_2, u_3 \rangle = \langle 1, 1, 1 \rangle$
  - Let D have 5 documents w/ their *vector representations* as

$$\langle 3, 0, 0 \rangle, \langle 1, 1, 0 \rangle, \langle 0, 0, 2 \rangle, \langle 2, 0, 2 \rangle, \langle 0, 0, 0 \rangle$$

and the expanded generating function is

$$a_1 \times X^{b1} + a_2 \times X^{b2} + \ldots + a_c \times X^{bc}$$

which yields $(0.6 \times X^2 + 0.4) \times (0.2 \times X + 0.8) \times (0.4 \times X^2 + 0.6)$, and

$$0.048\,X^5 + 0.192\,X^4 + 0.104\,X^3 + 0.416\,X^2 + 0.048\,X + 0.192$$

hence,

$est\_NoDoc(3, q, D) = 5 \times (0.048 + 0.192) = 1.2$, and

$est\_AvgSim(3, q, D) = (0.048 \times 5 + 0.192 \times 4)/(0.048 + 0.192)$

$= 4.2$

whereas,

$NoDoc(3, q, D) = 1$, since $sim(q, d_4) = 2 \times 1 + 2 \times 1 = 4$, and

$AvgSim(3, q, D) = 4 / 1 = 4$

Ranking

- Most commonly-used models: *Boolean* or *Vector Space* (VSM) and their variations
- Ranking has to be performed without accessing the text, just the index
- *Ranking algorithms*: almost impossible to measure recall as the number of relevant pages can be quite large for simple queries
- *No public information* is avaliable about the specified ranking algorithms used by current search engines: all information are “top secret”

Ranking

- Comparison of these techniques considering 56 queries over a collection of 2400 Web pages shows that VSM yields a better *recall-precision curve*, with an *average precision* of 75%
- Some of the new ranking algorithms also use hyperlink information, an important difference between the Web and normal IR DBs.
  - Number of hyperlinks that point to a page provides a measure of its *popularity* and *quality*
- Links in common between pages often indicates a *relationship* between those pages

Ranking

- Three examples of ranking techniques based in link analysis
  - WebQuery allows *visual browsing* of Web pages
  - WebQuery takes a set of Web pages (e.g., the answer to a query) and *ranks* them based on how connected each Web page is

Ranking

2. ***Kleinberg ranking scheme*** depends on the query & considers

   (i) the set of pages *S* that point to, or

   (ii) are pointed by pages in the answer

   - Pages that have many links pointing to them in *S* are called authorities
   - Pages that have many outgoing links are called hubs

- Better authority pages come from incoming edges from good hubs and better hub pages come from outgoing edges to good authorities

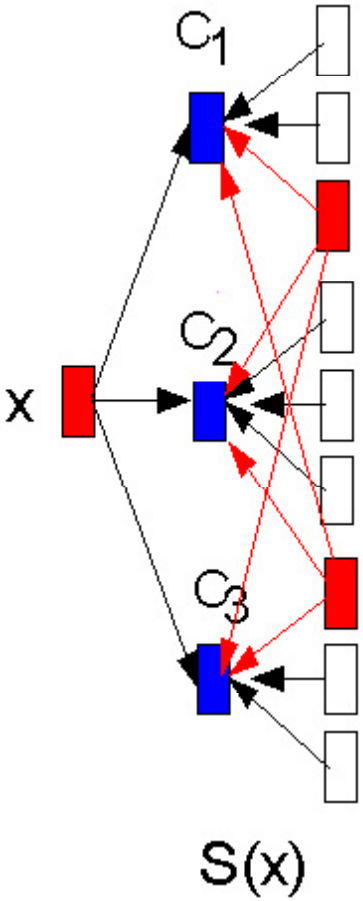
Ranking

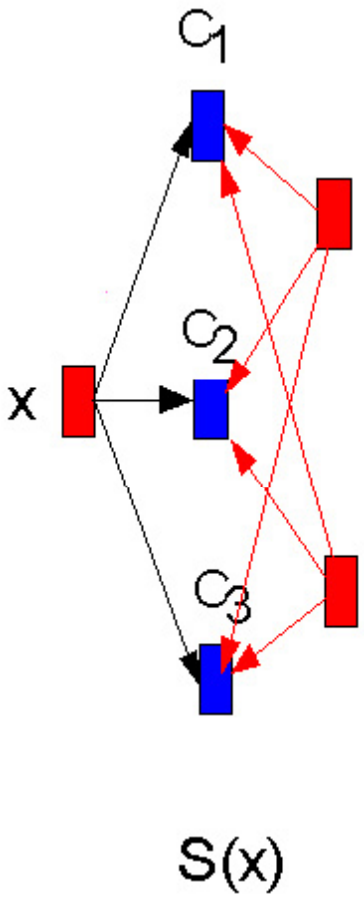
***H*(*p*): the *hub value* of page *p*, where *S* is a set of pages**

$$H(p) = \sum_{u \in S | p \to u} A(u)$$

A*(*p*): the *authority value* of page *p

$$A(p) = \sum_{v \in S | v \to p} H(v)$$





Ranking

3. ***PageRank*** (*PR*) simulates a user navigating randomly in the Web who jumps to a *random page* with probability *q* or follows a *random hyperlink* (on the current page) with probability (1 – *q*)

   - This process can be modeled with a *Markov chain*, from where the *stationary probability* of being in each page can be computed

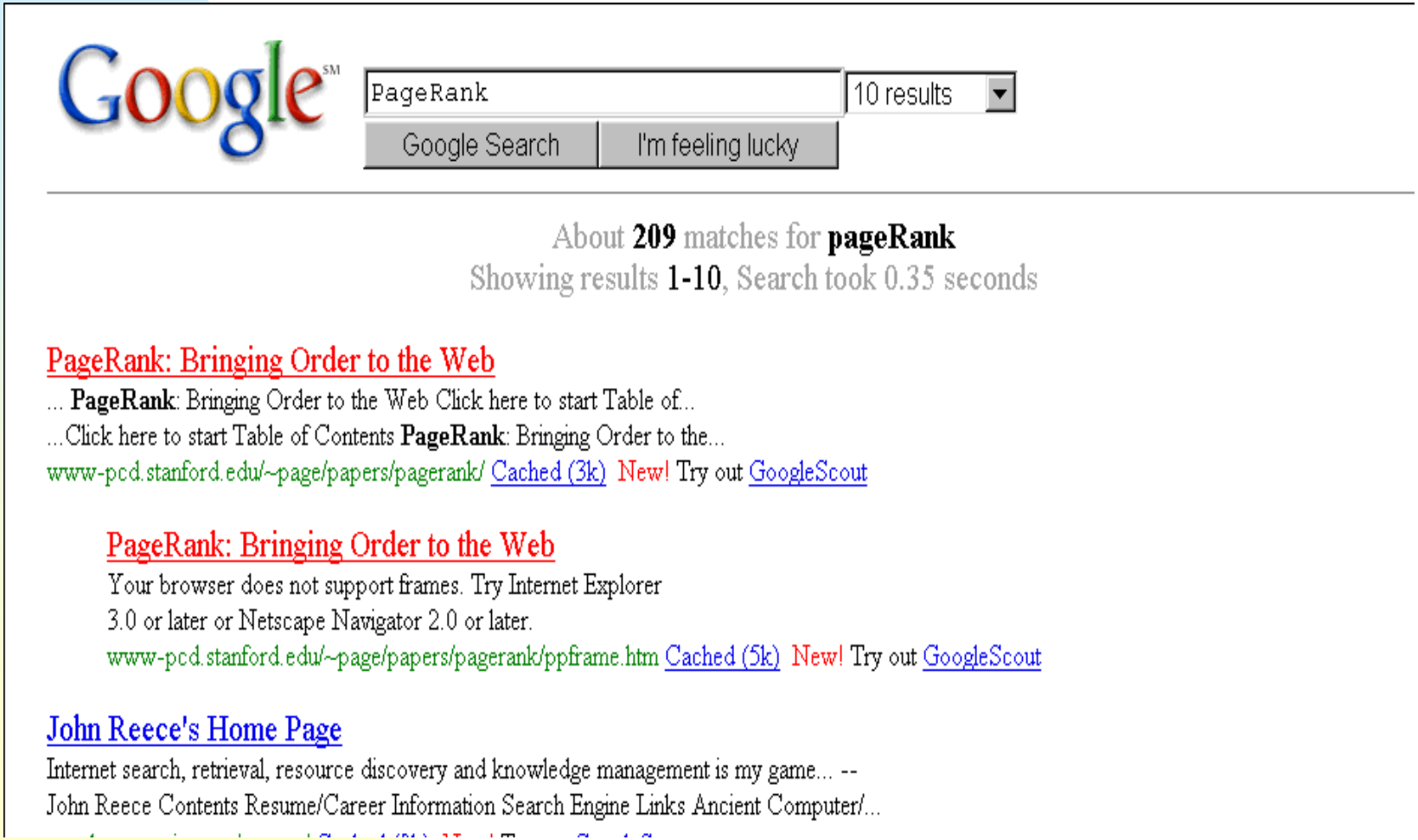
   - Let *C*(*a*) = no. of outgoing links of page *a* and suppose that page *a* is pointed to by pages p_1 to p_n. Then

$$PR(a) = q + (1-q)\sum_{i=1}^{n} \frac{PR(p_i)}{C(p_i)}$$

Ranking

$$PR(a) = q + (1-q)\sum_{i=1}^{n} \frac{PR(p_i)}{C(p_i)}$$

, where probability q is set by the system (~ 0.15)

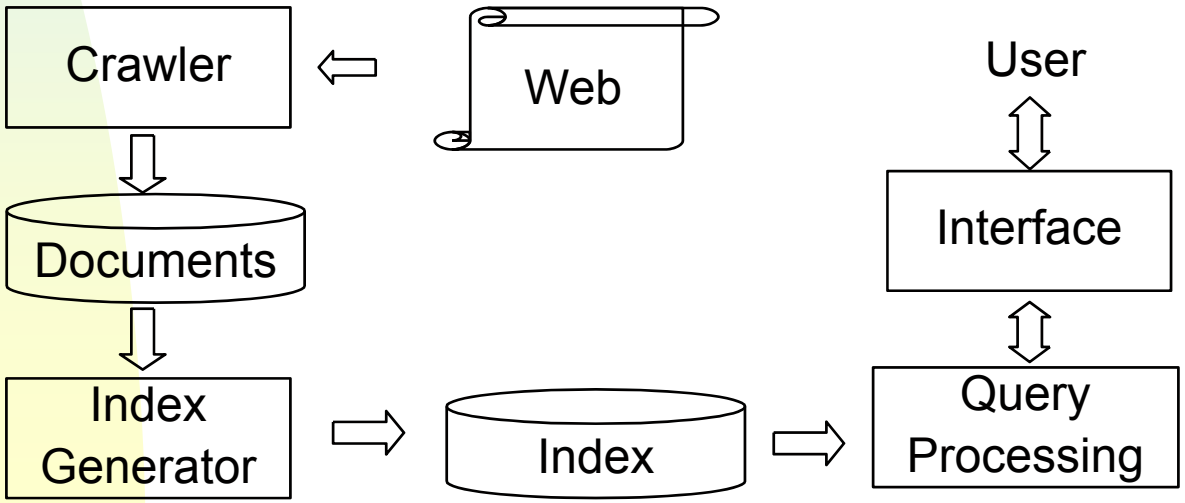


Search Engines

- Model the Web as a full-text DB
- A search engine is a system which collects, organizes and presents a way to select Web documents based on certain words, phrases, or patterns within documents
- Most search engines use the *whole* Web, while some use only a *limited* portion of the Web
- Two types of search engine architectures:
  - Centralized architecture
  - Distributed architecture

Search Engines

- Centralized (crawler-indexer):
  - *Crawler/Spider* (software agent): sends requests to remote Web servers & retrieves new/updated pages for *indexing*
  - *Index system*: maintains frequency-sorted inverted files
  - *Query processing*: processes documents in *ps*, *pdf*, *doc* & *HTML* formats
  - *User interface*: handling different languages



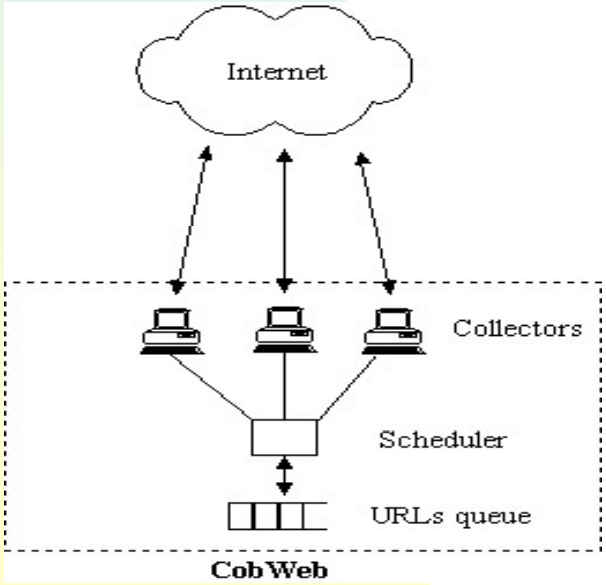
Search Engines

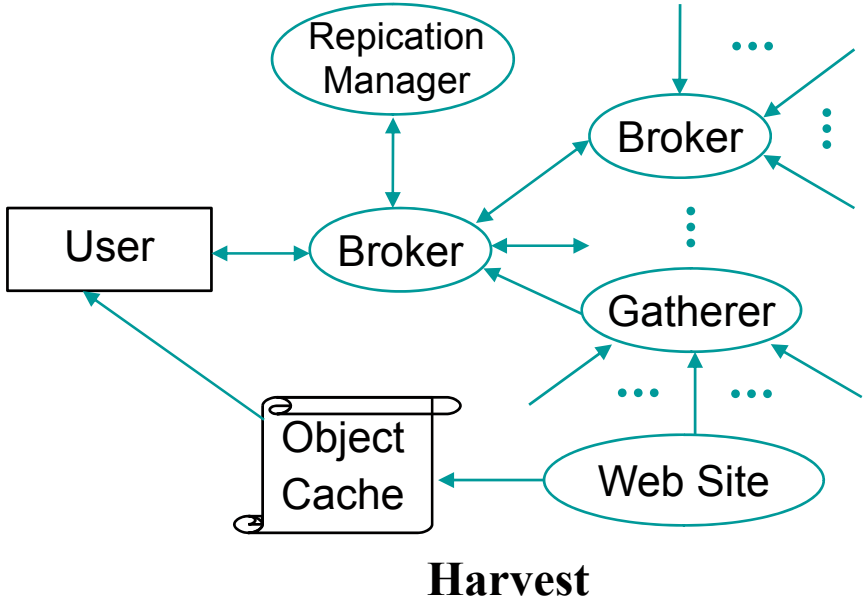
- Centralized (crawler-indexer)
  - Problems:
    1. Stability: Data are highly *dynamic* nature
    2. Connection: *Saturated* communication links
    3. Availability: *High load* at Web servers
    4. Quantity: *Hugh volume* of pages (indexed in 1998)
       - Alta Vista (140 millions), HotBot (110 millions), NorthernLight (67 millions), Excite (55 millions)
  - Impacts:
    - Increase *system load*: requests from different crawlers
    - No *coordination* among different crawlers
    - *Bandwidth* are wasted: discard “useless” pages

Search Engines

- **Distributed**
  - Gather and distribute (broker) data
  - *Gatherer*: collects & extracts indexing information from different Web servers
  - *Broker*: (i) provides the indexing mechanism & the query interface to the data

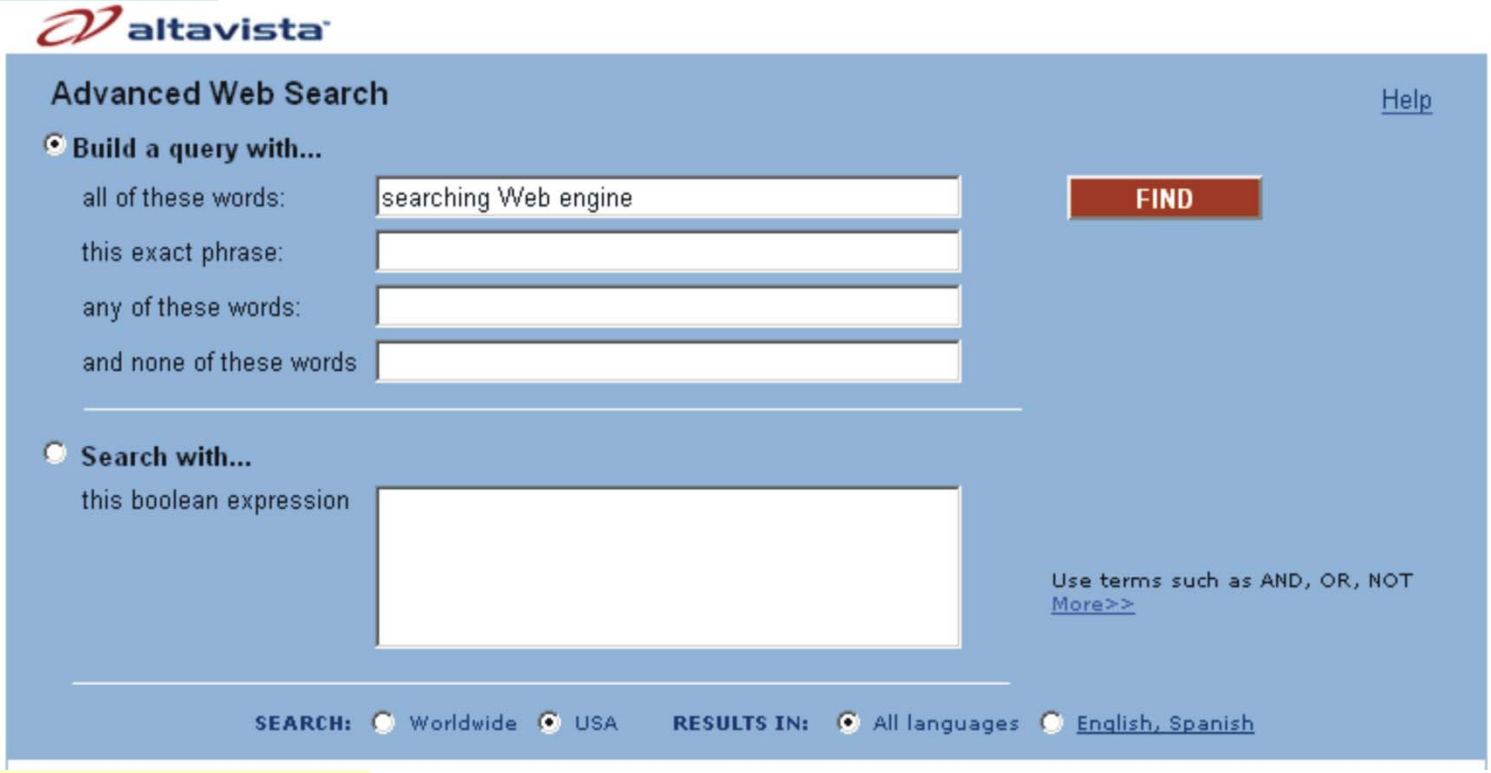
    (ii) retrieves info. from ≥ 1 brokers or gathers





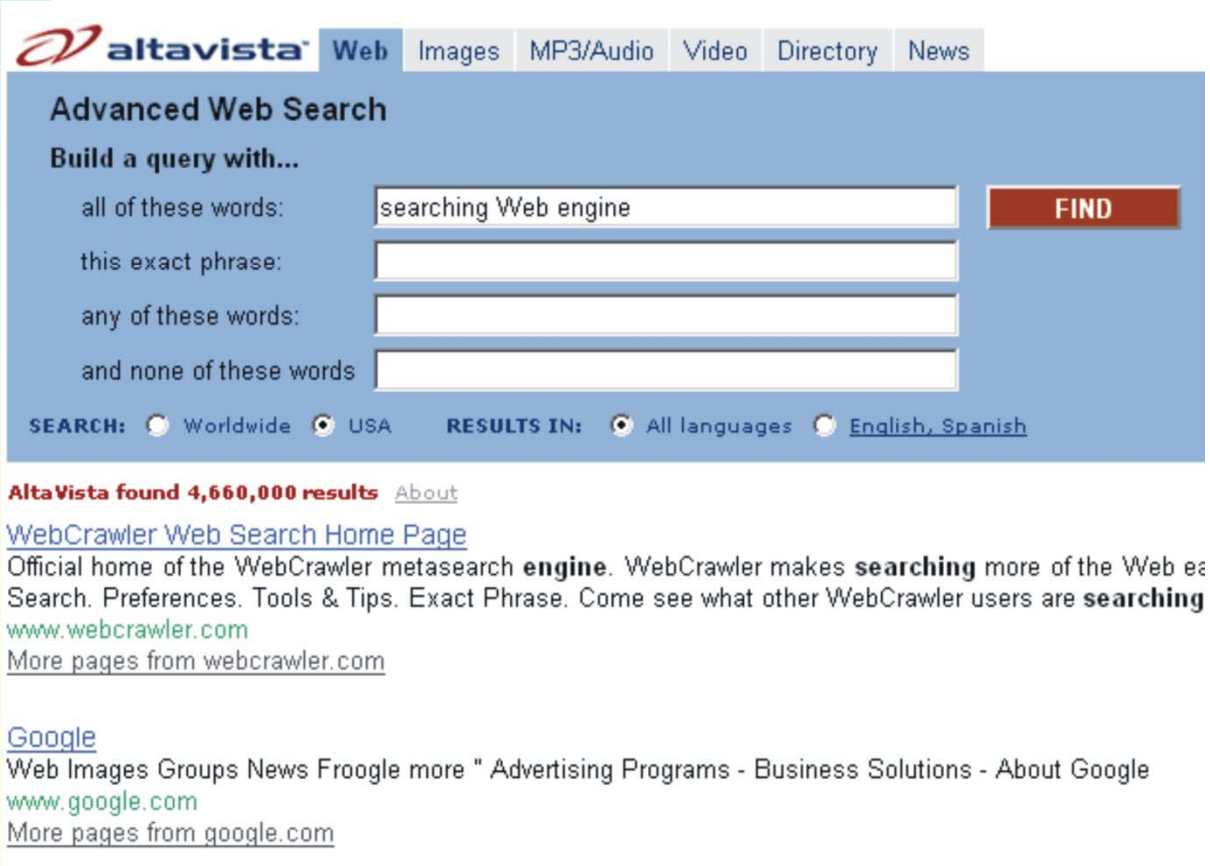
User Interfaces

- **Query Interface**:
  - Search for documents by considering *all* or *any* of the words in the input box
  - *Phrase* search, *proximity* search, *wide cards*: using Boolean operators w/ present/absent of any words



User Interfaces

- **Answer Interface**:
  - Consists of a list of *top ranked*, retrieved Web docs
  - Order of the list is determined by *relevance*
  - User can *refine* the query based on the retrieved docs



Crawling the web

- Start with a set of URLs and from there extract other URLs which are followed *recursively* in a breadth-first or depth-first fashion
- Strategies:
  - Search engines allow users to submit *top* Web sites that will be added to the URL set, or
  - Start with a set of popular URLs, because we can expect that they have information frequently asked
- Both cases work well for one crawler, but it is difficult to coordinate several crawlers to avoid visiting the same page > 1

Crawling the Web

- The *order* in which the URLs are traversed is important
  - Breadth first: First look at all the pages *linked* by the current page, and so on. This matches "well" Web sites that are structured by related topics, but the coverage will be *wide* but *shallow*
  - Depth first: Follow the 1st link of a page & do the same on the page until can't go deeper, returning recursively – a *narrow* but *deep* traversal
  - Good ordering schemes can make a difference if crawling *better pages first* (*PageRank*)

Crawling the Web

- Robots can overwhelm a server with rapid requests and can use significant *bandwidth*; a set of *guidelines* for robot behavior has been developed
- Crawlers can also have problems with HTML pages that use *frames* (dividing a page) or *image* maps (hyperlinks to image)
- Dynamically generated pages cannot be indexed as well as password protected pages

Crawling the Web

- Another approach: partition the Web using *country codes* or *Internet names*, and assign ≥ 1 robots to each partition, and explore each partition exhaustively
- Web traversal: the *index* of a search engine can be thought of as analogous to *stars* in the sky. What we see has never existed, as the light has traveled different distances to reach our eye

Crawling the web

- Similarly, Web pages referenced in an index were also explored at *different* dates and they may not *exist* any more
- Q: How fresh are the Web pages referenced in an index?

  A: The pages will be from one day to two months old. Most search engines show in the answer the date when the page was indexed
- The percentage of *invalid links* stored in search engines vary from 2% to 9%

Crawling the Web

- Some engines traverse the *whole* Web site, while others select just a *sample* of pages or pages up to a certain depth
- There are some engines that learn the *change frequency* of a page and visit it accordingly
- The current *fastest* crawlers are able to traverse up to 10 million Web pages per day

Indices

- Most indices use *variants* of the inverted file
  - An inverted file is a list of sorted words (*vocabulary*), each one having a set of *pointers* to the pages where it occurs
  - Some search engines use *elimination of stopwords* to reduce the size of the index. Normalization operations may include *removal of punctuation* and *multiple spaces*, etc.
  - To give the user some idea about each document retrieved, the index is complemented with a short *description* of each Web page

Indices

- Assume that 500 bytes are required to store the URL & the description of each Web page, we need 50 GB to store the description for 100 million pages
- A user initially receives only a *subset* of the complete answer to each query
- The search engine usually keeps the whole answer set in memory, to avoid having to *recompute* it if the user asks for more documents

Indices

- Indexing techniques can reduce the size of an inverted file to about 30% of the size of the text (less if *stopwords* are used). For 100 million pages, this implies about 15 GB of disk space
- A query is answered by doing a *binary search* on the sorted list of words of the inverted file
- Searching *multiple words*, the results have to be combined to generate the final answer
- Problem: words occur *frequently* in documents

Indices

- Inverted files can point to the actual occurrences of a word within a doc in space for the Web
  - too costly (space): each pointer has to specify (i) a page & (ii) a position inside the page
  - alternative: *word numbers* can be used instead of actual *bytes*; however, *positions* of the words in a page allow *phrase searches* or *proximity queries* (for finding words that are near each other)
- Currently, some search engines are providing phrase searches, but the actual implementation is not known

Indices

- Finding words starting with a given prefix requires 2 *binary* searches in the sorted list of words
- More complex searches (e.g., wild card searches) can be performed by doing a *sequential scan* over the vocabulary
  - The best *sequential algorithms* can search around 20 Mb of text stored in RAM in 1 second (5 MB is about the vocabulary size for 1 GB of text)
  - Even though Sequential search is too slow for the Web, it's not impossible
    - Using ***Heaps' law***, which estimates the size of the vocabulary of a text, and let $\beta = 0.7$ for the Web, the vocabulary size for 1 Tb is 630 Mb $\Rightarrow$ a searching time of 30 seconds

Indices

- Pointing to pages or to word positions is an indication of the *granularity* of the index
- The index can be less *dense* if we point to logical blocks instead of pages
  - Reduce the variance of the different document sizes, by making all (logical) blocks roughly the *same* size
    - Reduces the *size* of the pointers (because there are fewer blocks than documents)
    - Reduces the *number* of pointers (because words have *locality* of reference)

Indices

- ***Glimpse*** (core of Harvest)
  - Queries are resolved as for inverted files, obtaining a list of blocks that are searched sequentially (sequential search can be done over 30 MB/sec in RAM)
  - originally used only 256 blocks, efficient up to 200 MB for searching words that were not too *frequent*, obtaining an index of only 2% of the text
  - Not used in the Web because sequential search can not be afforded, as it implies a network access. However, in a *distributed architecture* where the index is also distributed, *logical blocks* make sense

Indices

Announcements and News

Overview

WebGlimpse adds search capabilities to your WWW site automatically and easily. It attaches a small search box to the bottom of every HTML page, and allows the search to cover the *neighborhood* of that page or the whole site. With WebGlimpse there is no need to construct separate search pages, and no need to interrupt the users from their browsing pages remain unchanged except for the extra search capabilities. It is even possible for the search to efficiently cover r pages linked from your pages. (WebGlimpse will collect such remote pages to your disk and index them.) Installation, customization (e.g., deciding which pages to collect and which ones to index), and maintenance are easy.

Conclusions

- Nowadays, search engines uses basically *Boolean* or *Vector* (*Space*) models and their variations
- *Link Analysis* Techniques seems to be the “next generation” of the search engines
- Indices: *compression* and *distributed architecture* are keys

References

- CHAKRABARTI, S.; DOM, B.; RAGHAVAN, P.; RAJAGOPALAN, S.; GIBSON, D.; KLEINBERG, J. Automatic Resource Compilation by Analyzing Hyperlink Structure and Associated Text, 1998
- GIBSON, D.; KLEINBERG, J.; RAGHAVAN, P. Structural Analysis of the World Wide Web. Position paper at the WWW Consortium Web Characterization, 1998
- GOOGLE, www.google.com
- Members of the Clever Project. Hypersearching the Web. Scientific American, March, 1999
- TODOBR, www.todobr.com.br
- WEBGLIMPSE, glimpse.cs.arizona.edu/webglimpse/