

CS 589 Fall 2020

Information retrieval infrastructure

Instructor: Susan Liu

TA: Huihui Liu

Stevens Institute of Technology

Metrics for a good search engine

- Return what the users are looking for
- Return results fast
- Users likes to come back
- Relevance, CTR = click thru rate
- Latency
- Retention rate

Information Retrieval Techniques

How does Google know cs 589 refers to a course?
How does Google

cs 589 **stevens**

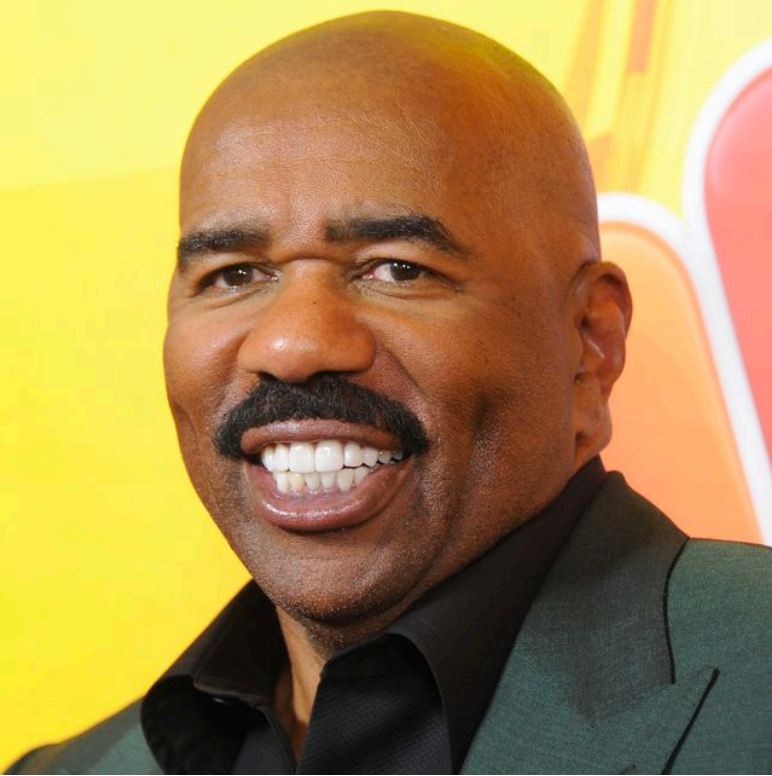
[All](#) [Images](#) [Maps](#) [News](#) [Videos](#) [More](#)

About 3,220,000 results (0.58 seconds)

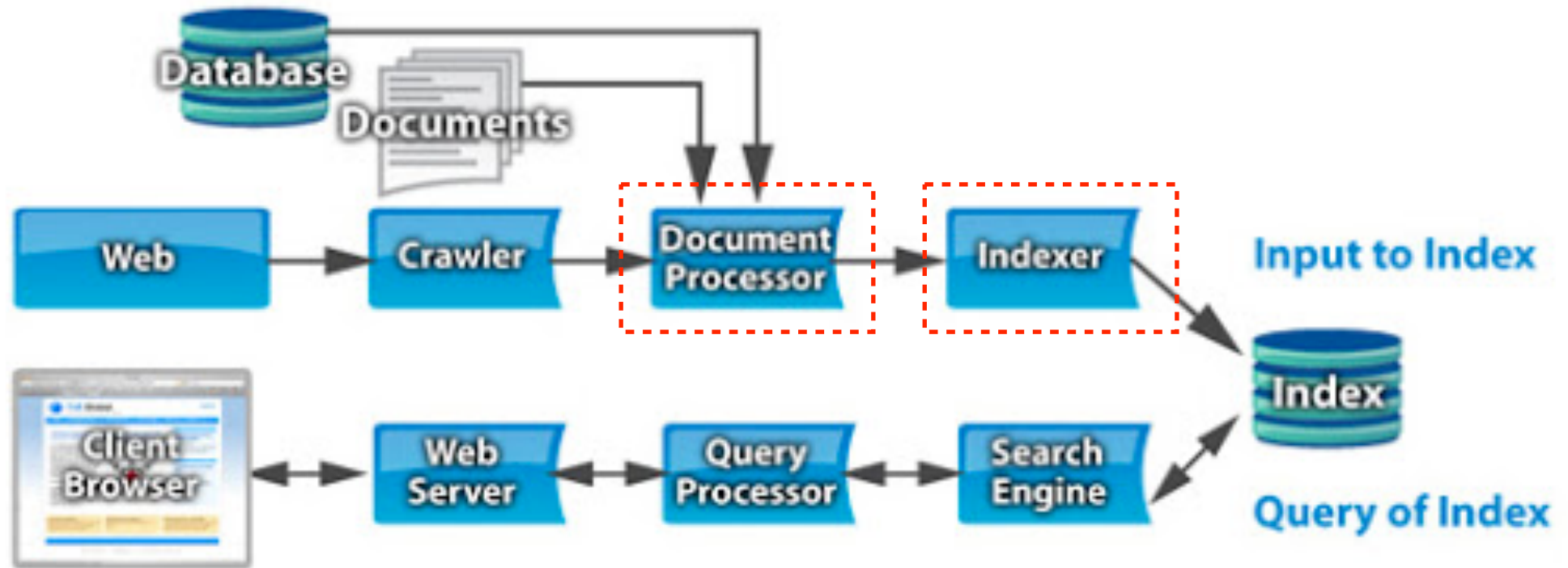
www.cs.stevens.edu > [~xliu127](#) > [teaching](#) > [cs589_20f](#)

cs589

CS 589: Text Mining and Information Retrieval. Home | Canvas | Resources
Stevens' guidelines on the coronavirus emergency (COVID-19), ...



Information Retrieval Infrastructure



Inverted index

- In Lecture 2, we learned retrieval models
 - Compute $\text{score}(q, d)$
 - Select the d that maximizes $\text{score}(q, d)$
- In an industry scale search engine, there could be trillions of q 's and billions of d 's
 - For each query, search time complexity = $O(|D|)$
 - Solution for faster retrieval: inverted index

Inverted index

1: Winter is coming.
2: Ours is the fury.
3: The choice is yours.



<u>term</u>	<u>freq</u>	<u>documents</u>
choice	1	3
coming	1	1
fury	1	2
is	3	1, 2, 3
ours	1	2
the	2	2, 3
winter	1	1
yours	1	3

Dictionary

Postings

time complexity: $O(\#unique\ words\ in\ q \times avg_len(postings\ lists))$

$$\ll |D|$$

Problems with inverted indexing

- Data processing
 - Choosing the unit for indexing
 - Determining the vocabulary
- Constructing/speeding up inverted index
 - Skipping index
 - Prefix indexing
 - Indexing with blocks
 - MapReduce
- Index compression
- Other issues
 - Indexing position
 - Spelling correction

Choosing the correct document unit for indexing

- Documents often consists of sub documents
 - e.g., email contains multiple attached documents
- Trade-off on the unit size
 - Smaller units: missing important passages
 - Larger unit: gets spurious matches, e.g.,**text** messages....gold **mining**...

Determining the vocabulary

- Tokenization

Input: Friends, Romans, Countrymen, lend me your ears;

Output: Friends Romans Countrymen lend me your ears

- o'neil, aren't, C#

- Dropping stop words

- Stop words are common terms
- **Web search engines generally do not use stop words!**

A	It	These
About	Its	They
Again	Itself	This
All	Just	Those
Almost	km	Thus
Also	Made	To
Although	Mainly	Upon
Always	Make	Use
An	May	Used
And	mg	Using
Another	Might	Various
Any	ml	Very
Are	mm	Was
As	Most	We
At	Mostly	Were

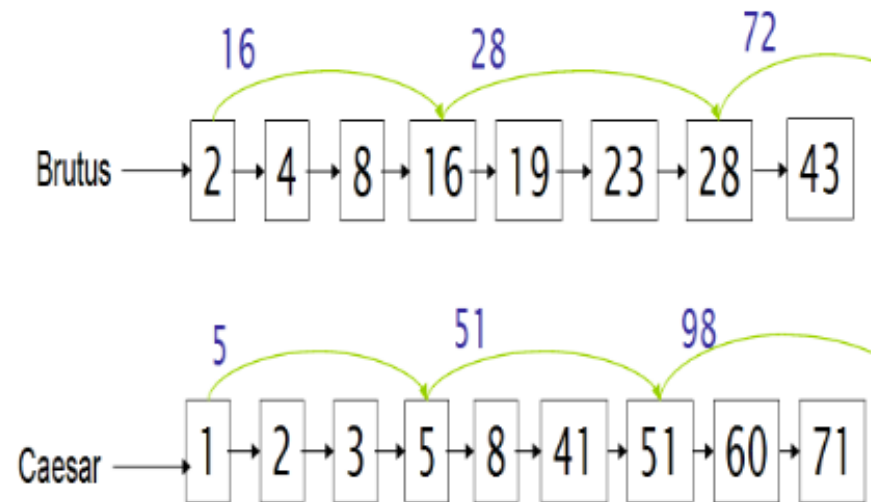
Determining the vocabulary

- Normalization
 - Abbrev: USA vs. United states of America
 - Case:
 - Cat -> cat
 - **SAT** -> **sat**
- Stemming/lemmatization
 - singing -> sing, cars -> car, **sat** -> **sit**
 - porter stemmer, snowball stemmer

Speeding up: skipping lists

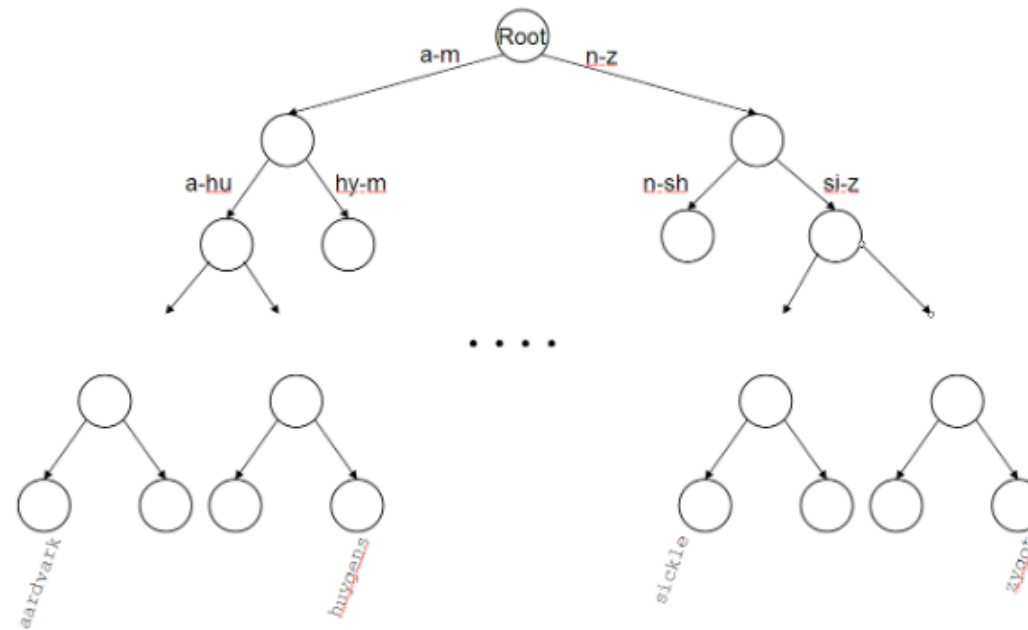
- Finding the intersection of two post listings
 - Without skip: $O(m + n)$

```
i, j = 0, 0
while i < m and j < n:
    if arr1[i] < arr2[j]:
        i += 1
    elif arr2[j] < arr1[i]:
        j += 1
    else:
        print(arr2[j])
        j += 1
        i += 1
```



Speeding up: prefix indexing

- Speeding up the indexing using prefix tree
 - time complexity: $O(\text{\#unique words in } q \times \text{avg_len}(\text{postings lists}))$

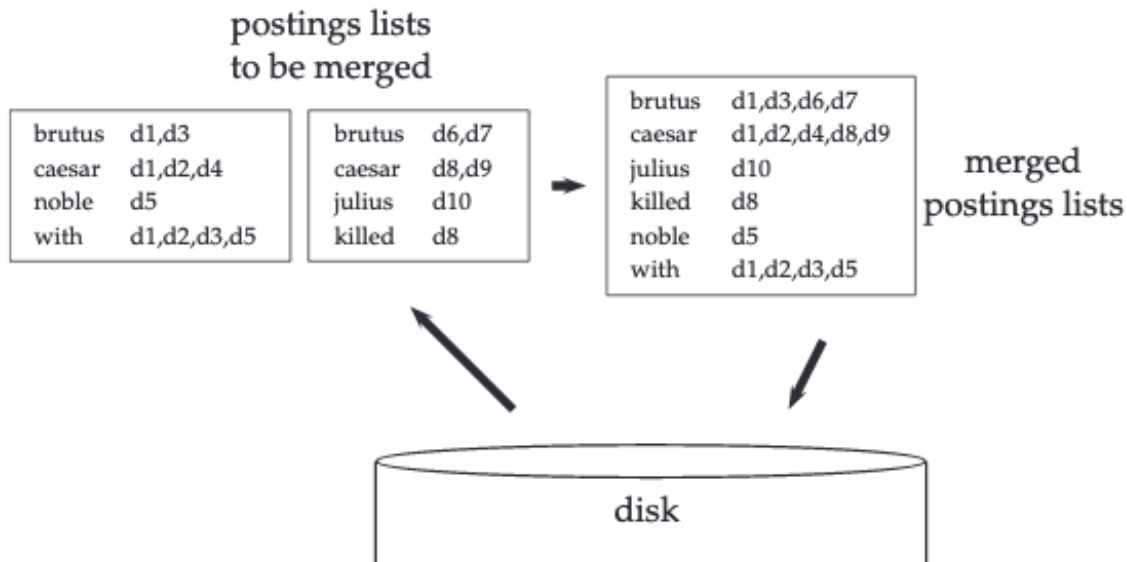


Constructing inverted index: hardware basics

- Decisions on an IR system largely depends on the hardware which the system runs on
- **Chunks:**
 - Splitting data into more chunks takes more *seek time*
- **Blocks**
 - *Accessing data in memory >> accessing data on disk*
 - *Constructing* inverted index using blocks
 - Typical IR system: GBs of memory, disk space orders of magnitude larger

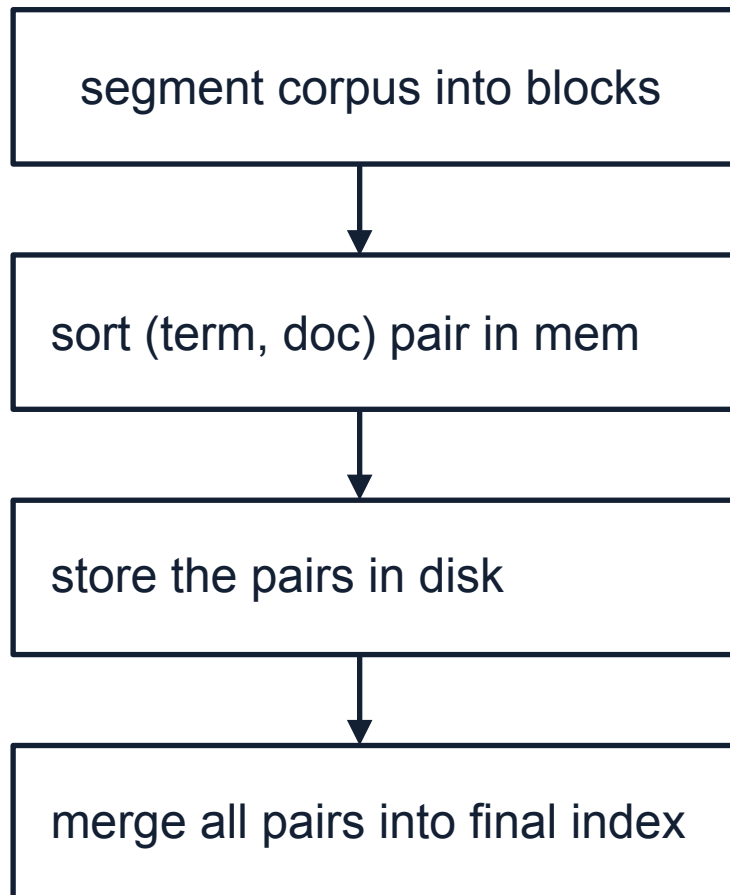
Block sort-based indexing (BSBI)

- Indexing large corpus
 - Reuters-RCV1: 2.5GB = 2.5×10^9 Bytes, 1 billion
 - Today's text corpus contains petabytes of data: 10^{15} Bytes
 - Memory \ll size of corpus



- Index each block using memory
- Write each blocks' index into disk
- Merge all inverted indices

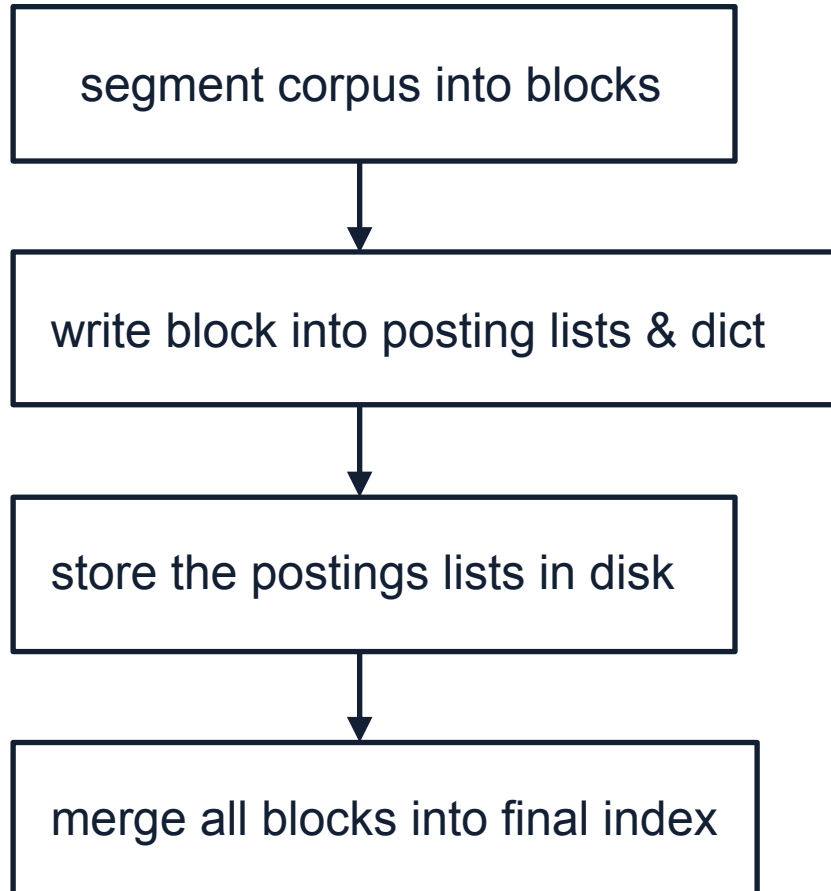
Block sort-based indexing



BSBINDEXCONSTRUCTION()

```
1  $n \leftarrow 0$   
2 while (all documents have not been processed)  
3 do  $n \leftarrow n + 1$   
4    $block \leftarrow \text{PARSENEXTBLOCK}()$   
5    $\text{BSBI-INVERT}(block)$   
6    $\text{WRITEBLOCKTODISK}(block, f_n)$   
7  $\text{MERGEBLOCKS}(f_1, \dots, f_n; f_{\text{merged}})$ 
```

Single-pass in-memory indexing

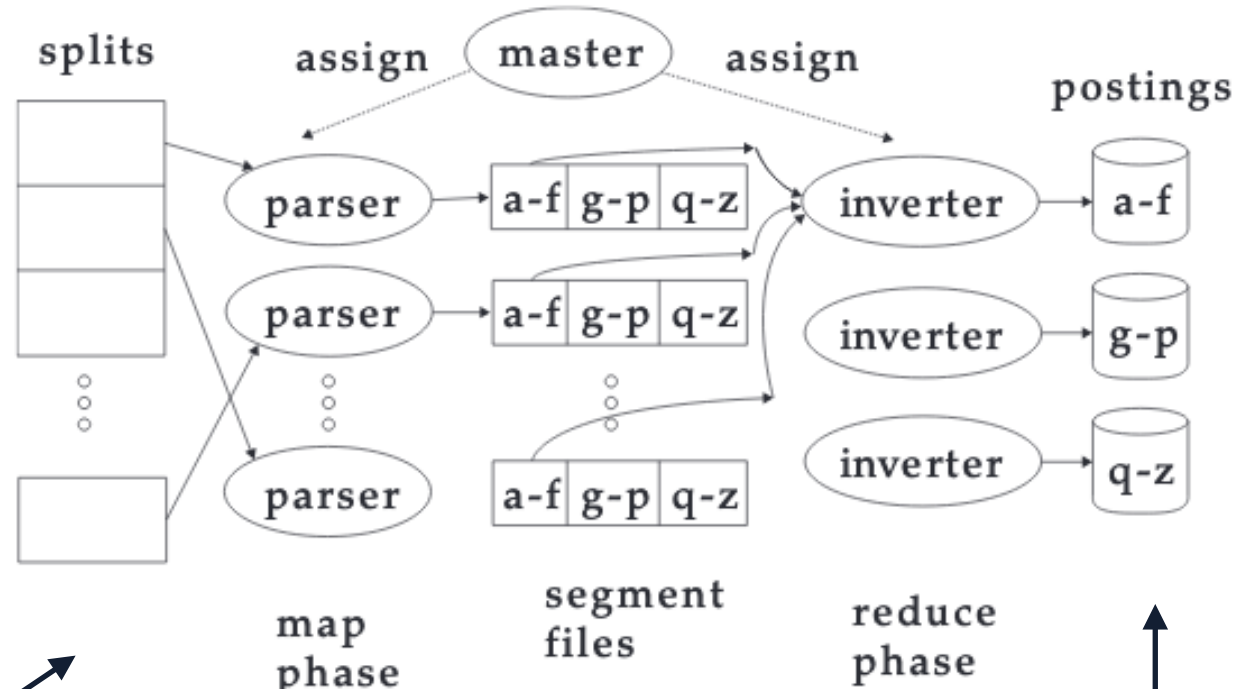


- Handling posting lists directly
- Eliminating the expensive sorting in BSBI
- Leveraging compression

Handling web scale indexing

- Web-scale indexing must use clusters of servers
 - Google had 1 million servers in 2011
- Fault tolerance of a massive data center
 - If a non-fault tolerance system has 1000 nodes, each has 99.9% uptime, then 63% of the time one or more servers is down
- Solution
 - Maintain a “master” server
 - Break indexing into parallel tasks
 - Assign each task to an idle machine

Map-reduce



master assigns split to idle machine

map phase
parser emits (term,doc) pair

reduce phase
merge partitions in inverter

complete the index

Examples of map-reduce

map: d_2 : C died. d_1 : C came, C c'ed.



$\langle\langle C, d_2 \rangle, \langle \text{died}, d_2 \rangle, \langle C, d_1 \rangle, \langle \text{came}, d_1 \rangle, \langle C, d_1 \rangle, \langle \text{c'ed}, d_1 \rangle\rangle$

reduce: $\langle\langle C, (d_2, d_1, d_1) \rangle, \langle \text{died}, (d_2) \rangle, \langle \text{came}, (d_1) \rangle, \langle \text{c'ed}, (d_1) \rangle\rangle$



$\langle\langle C, (d_1:2, d_2:1) \rangle, \langle \text{died}, (d_2:1) \rangle, \langle \text{came}, (d_1:1) \rangle, \langle \text{c'ed}, (d_1:1) \rangle\rangle$

MapReduce: Industry practice

- Term partition vs. document partition
 - Term-partitioned: one machine handles a subrange of terms
 - Document-partitioned: one machine handles a subrange of documents
- Most industry search engine use document-partitioned index
 - Better load balancing (**why?**)

MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean and Sanjay Ghemawat

jeff@google.com, sanjay@google.com

Google, Inc.

Logarithmic dynamic indexing

- Logarithmic merge:
 - Maintain a series of indexes, each twice the size of the previous one
 - Keep smaller ones in memory (Z0)
 - Larger ones on disks (I0, I1, ...)
 - If Z0 gets too big, write to disk as I0
 - Or merge with I0 as Z1
 - Either merge Z1 to disk as I1
 - Or merge with existing I1 to form Z2

Real time search of Twitter

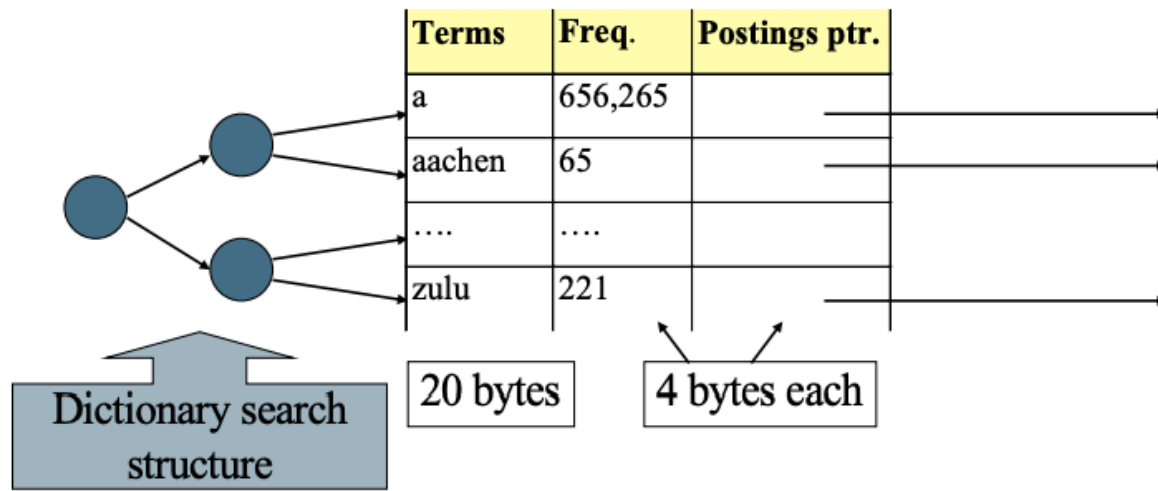
- Requires high real time search
 - Low latency, high throughput query evaluation
 - High ingestion rate and immediate data availability
 - Concurrent reads and writes of the index
- Solution: using segments
 - Each segment consists of 2^{32} tweets (in memory)
 - New posts are appended to the posting lists
 - Only one segment can be written to at each time

Index compression

- Why compression?
 - Using less disk space
 - Compressing dictionary
 - Allowing the dictionary to be stored in memory
 - Compressing posting files
 - Reducing disk space
- Zipf's law
 - The i th most frequent term has frequency proportional to $1/i$

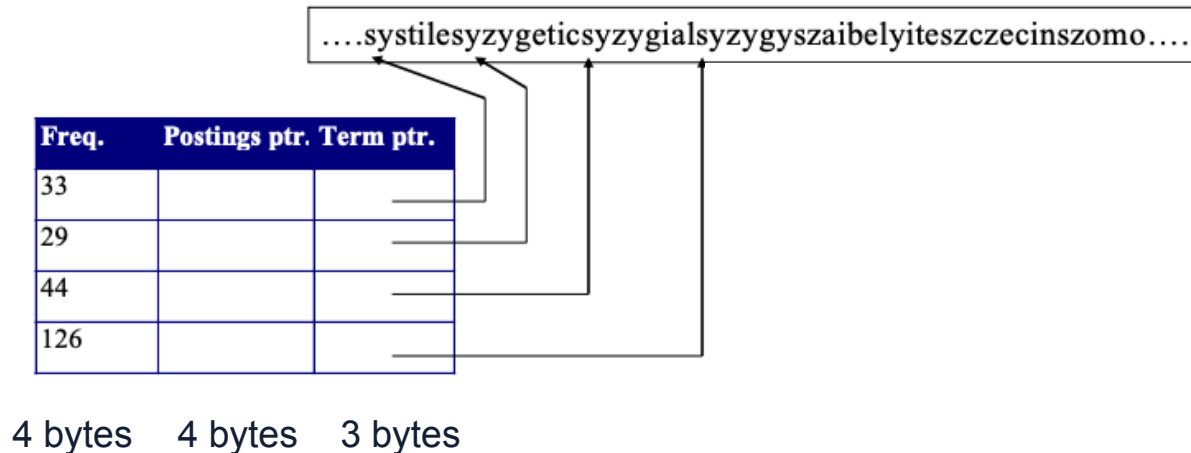
Dictionary compression

- Most of the space in the table is wasted
 - Most words are 20 bytes
 - Table storage = $28N$



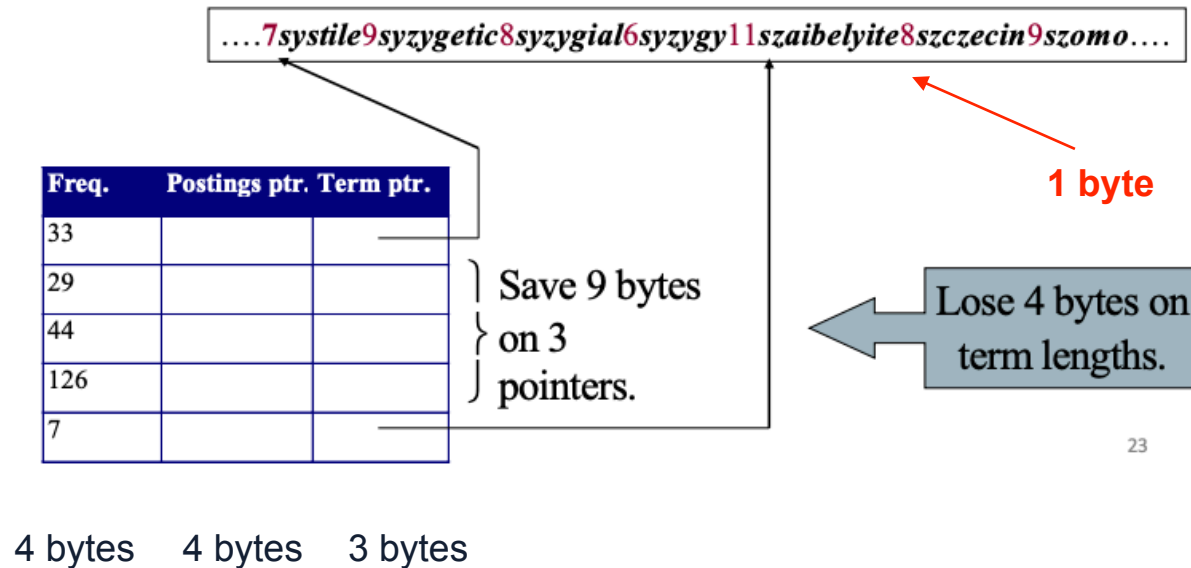
Dictionary-as-a-string

- Table storage = $11N$
- How to further improve the storage space?
 - Instead of storing absolute term pointers, store the gaps



Dictionary-as-a-string

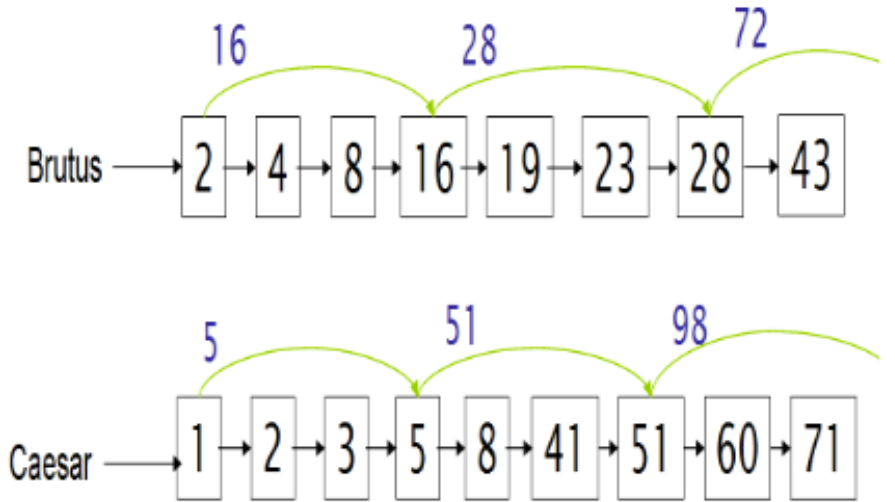
- Table storage = $8N + 3N * (7/12) = 9.75N < 11N$
- Trade-off between skipping more vs. skipping less



23

Postings compression

- Observations of posting files
 - Instead of storing docID, store gaps
 - Brutus: 2,4,8,3,4,5,15
 - Binary seq: 10,100,1000,11,100,101,1



- Prefix encoding
 - Binary encoding such that the sequence can be uniquely decoded
 - e.g., Huffman encoding
 - Unary encoding: {2:110,4:11110, ...}
 - A uniquely decodable seq: 110111101111111101110...

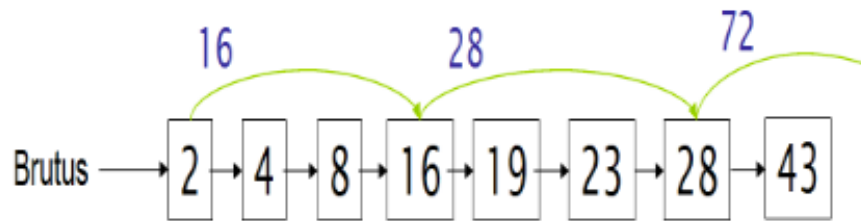
Postings compression

- Problem with unary encoding
 - Too long!
- **Gamma code** of 13: 1110,101
 - Unary code of length - 1: 1110
 - Offset (last length - 1 bits): 13 \rightarrow 1101 \rightarrow 101
- What is the gamma code of 5? 101 \rightarrow 110,01
- We can prove gamma code is uniquely decodable
- Gamma code compression rate: 11.7%

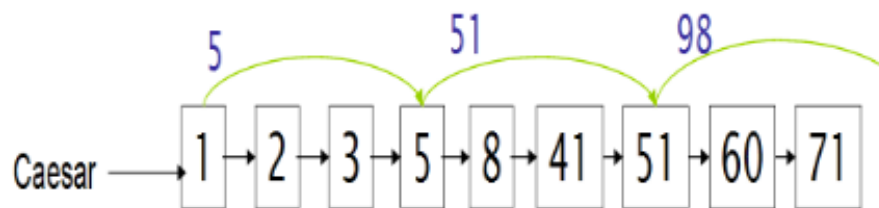
number	length	offset	γ -code
0			none
1	0		0
2	10	0	10,0
3	10	1	10,1
4	110	00	110,00
9	1110	001	1110,001
13	1110	101	1110,101
24	11110	1000	11110,1000
511	111111110	11111111	111111110,11111111
1025	11111111110	0000000001	11111111110,0000000001

Indexing Position

- Indexing the position of word within the document
- Intersection algorithm finds where the two terms appear between within k words



Brutus: 2:<0>, 4: <429,433>, 8: <150>, ...



Caesar: 1:<10>, 2: <5>, 8: <17, 250>, ...

Spelling correction



stevens institute of **tecnology**

Showing results for **stevens institute of *technology***

- Edit distance
- k-gram index for spelling correction
- context sensitive spelling correction

Edit distance

- Dynamic programming: $O(|s_1| \times |s_2|)$

```

EDITDISTANCE( $s_1, s_2$ )
1  int  $m[i, j] = 0$ 
2  for  $i \leftarrow 1$  to  $|s_1|$ 
3  do  $m[i, 0] = i$ 
4  for  $j \leftarrow 1$  to  $|s_2|$ 
5  do  $m[0, j] = j$ 
6  for  $i \leftarrow 1$  to  $|s_1|$ 
7  do for  $j \leftarrow 1$  to  $|s_2|$ 
8     do  $m[i, j] = \min\{m[i - 1, j - 1] + \text{if } (s_1[i] = s_2[j]) \text{ then } 0 \text{ else}$ 
9          $m[i - 1, j] + 1,$ 
10         $m[i, j - 1] + 1\}$ 
11  return  $m[|s_1|, |s_2|]$ 

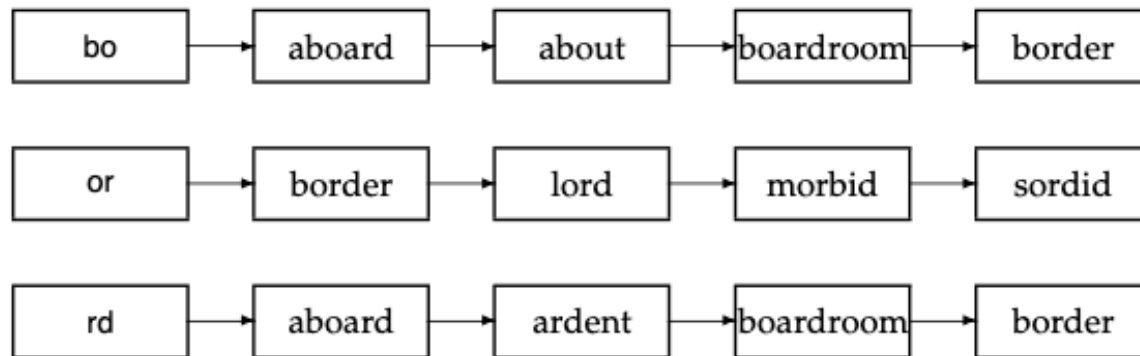
```

		f	a	s	t
	0	1 1	2 2	3 3	4 4
c	1 1	1 2 2 1	2 3 2 2	3 4 3 3	4 5 4 4
a	2 2	2 2 3 2	1 3 3 1	3 4 2 2	4 5 3 3
t	3 3	3 3 4 3	3 2 4 2	2 3 3 2	2 4 3 2
s	4 4	4 4 5 4	4 3 5 3	2 3 4 2	3 3 3 3

Levenshtein distance

k-gram indexes for spelling correction

- Running DP on all pairs of words is time consuming
- Leveraging k-gram index to speed up spelling correction
 - boardroom vs. bord



boarder:3

boardroom: 2

aboard: 2

ardent:1

...

Context sensitive spelling correction

- How to correct “flew form healthrow”?
 - All three words are spelled correctly
 - Enumerating each character: the space is large
 - Solution: using logs of queries, e.g., **flew from** vs. fled fore

Li et al. A generalized hidden Markov model with discriminative training for query spelling correction. SIGIR 2012

PageRank

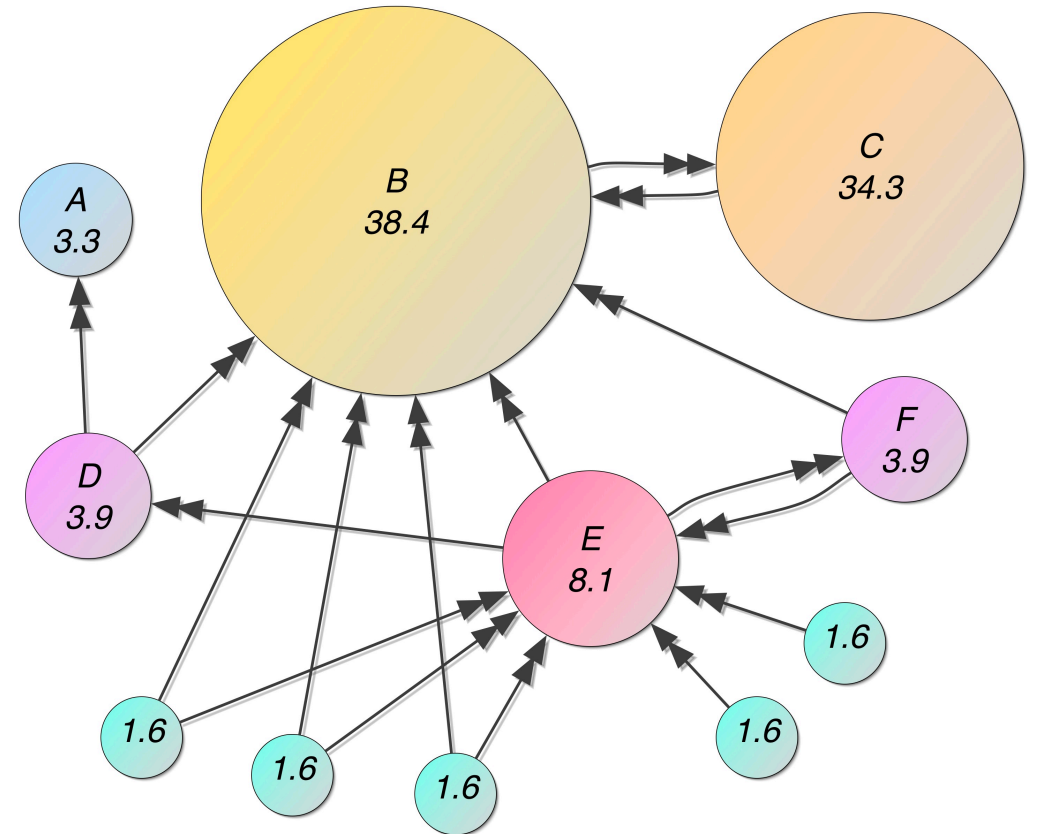
- How to rank webpages?
 - Using retrieval models: only captures relevance
- Capturing quality of web pages:
 - Based on how often the page is cited
 - Intuition: a popular website (e.g., Google) would be cited by a lot of other webpages

PageRank

- ***“The Anatomy of a Large-Scale Hypertextual Web Search Engine”*** - Sergey Brin and Lawrence Page, *Computer networks and ISDN systems*, 1998

$$PR(p_i) = \frac{1 - d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$

- Favors pages that are highly cited, and pages cited by highly cited pages

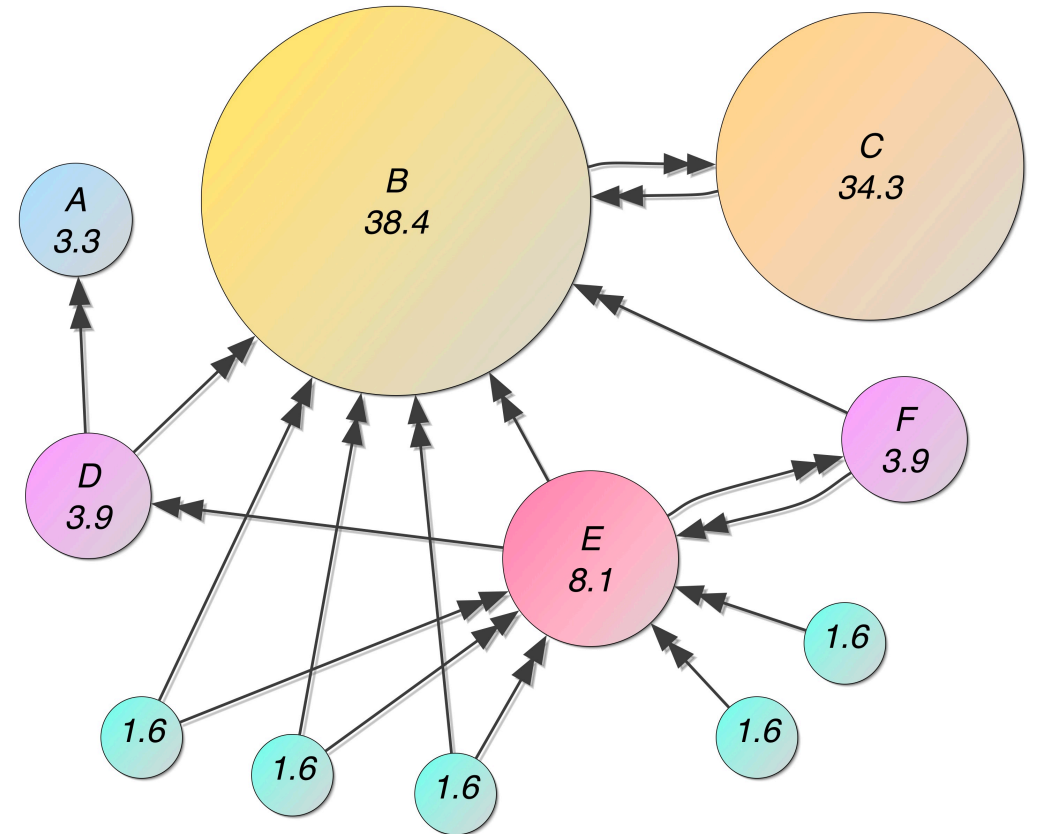


1/2 probability of randomly walking into B

PageRank

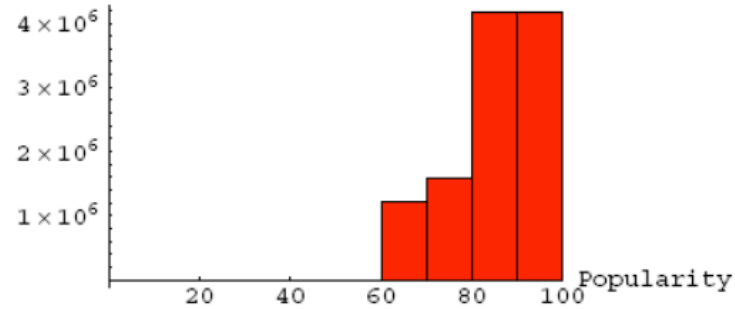
- Assign each node an initial page rank
- Repeat until convergence
 - Calculate the page rank of each node using the equation

$$PR(p_i) = \frac{1 - d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$

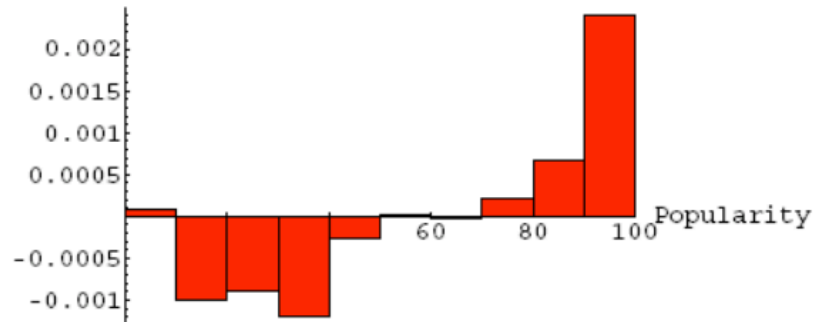


Problems of page rank

Absolute increase in the no. of In-Links



Absolute increase in the PageRank values



rich gets richer



Web Images Groups News Froogle Local more »

miserable failure

Search

Advanced Search
Preferences

Web

Results 1 - 10 of about 969,000 for [miserable failure](#). (0.06 seconds)

[Biography of President George W. Bush](#)

Biography of the president from the official White House web site.

www.whitehouse.gov/president/gwbbio.html - 29k - [Cached](#) - [Similar pages](#)

[Past Presidents](#) - [Kids Only](#) - [Current News](#) - [President](#)

[More results from www.whitehouse.gov »](#)

[Welcome to MichaelMoore.com!](#)

Official site of the gadfly of corporations, creator of the film Roger and Me and the television show The Awful Truth. Includes mailing list, message board, ...

www.michaelmoore.com/ - 35k - Sep 1, 2005 - [Cached](#) - [Similar pages](#)

[BBC NEWS | Americas | 'Miserable failure' links to Bush](#)

Web users manipulate a popular search engine so an unflattering description leads to the president's page.

news.bbc.co.uk/2/hi/americas/3298443.stm - 31k - [Cached](#) - [Similar pages](#)

[Google's \(and Inktomi's\) Miserable Failure](#)

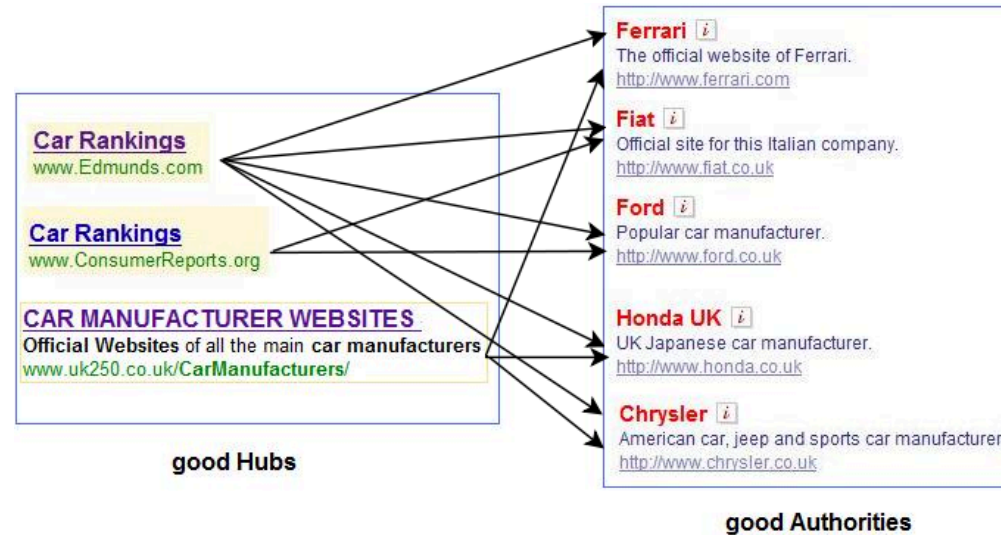
A search for **miserable failure** on Google brings up the official George W. Bush biography from the US White House web site. Dismissed by Google as not a ...

searchenginewatch.com/sereport/article.php/3296101 - 45k - Sep 1, 2005 - [Cached](#) - [Similar pages](#)

Google bombing

HITS

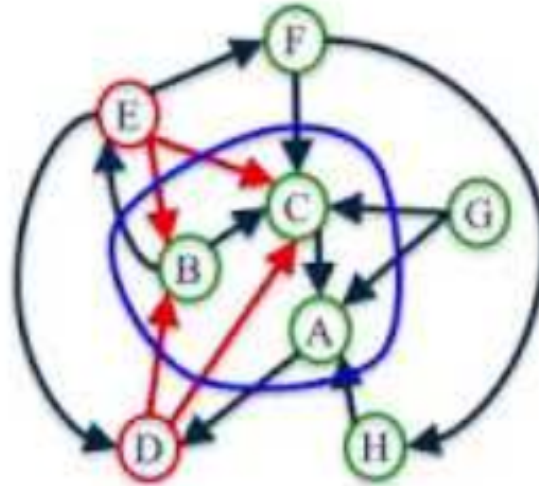
- Hubs: compilations of a broad catalog of information that led users direct to other authoritative pages
- Authorities: a page that is linked by many different hubs



Query: [Top automobile makers](#)

HITS











- Repeat k times
 - Update hub score: $v = A^T u$
 - Update authority score: $u = A^T v$



Search engine tools

- Apache Lucene
 - Free and open search engine library
 - First developed in 1999
- ElasticSearch
 - A search engine
 - based on Lucene

Elasticsearch vs Lucene
Which companies use these tools?

 Elasticsearch	 Uber	 9GAG
	 Asana	 Codecademy
 Lucene	 Twitter	 Evernote
	 Slack	 Kifi

ElasticSearch

- Using a REST api

Dev Tools

Console

```
1 POST bibliography/novels/_bulk
2 {"create": {"_id": "1"}}
3 {"author": "Johann Wolfgang von Goethe", "title": "Die
4 Leiden des jungen Werther", "year": "1774"}
5 {"create": {"_id": "2"}}
6 {"author": "Umberto Eco", "title": "Il nome della rosa",
7 "year": "1980"}
8 {"create": {"_id": "3"}}
9 {"author": "Margaret Atwood", "title": "The Handmaid's Tale",
10 "year": "1985"}
```

Dev Tools

Console

```
1 GET /integrity/body/870595443049000/_termvectors
2 ?pretty=true
3 {
4   "fields": ["_all"]
5 }
```

Homework 2: Using Elasticsearch to build a search engine

- Build an inverted index
- Evaluate three search algorithm's performance
 - TF-IDF
 - BM25
 - Dirichlet-LM

The screenshot shows a search engine interface. At the top, there's a 'Search Results' header with a link for 'Advanced Search Tips' and a blue 'Ask Question' button. Below the header, it says 'Results for how to sort dictionary by value'. There's a search input field containing 'how to sort dictionary by value' and a blue 'Search' button. Below the search bar, it shows '500 results' and three sorting options: 'Relevance' (selected), 'Newest', and 'More' with a dropdown arrow. The main content area displays a search result for the question 'Q: How do I sort a dictionary by value?'. It has 3420 votes and 34 answers. The question text is: 'how can I **sort** based on the values? Note: I have read Stack Overflow question here **How** do I **sort** a list of dictionaries **by** a **value** of the **dictionary**? and probably could change my code **to** have a list of ... I have a **dictionary** of values read from two fields in a database: a string field and a numeric field. The string field is unique, so that is the key of the **dictionary**. I can **sort** on the keys, but ...'. There are three tags: 'python', 'sorting', and 'dictionary'. The question was asked on Mar 5 '09 by Gern Blanston.