CMU SCS

15-826: Multimedia Databases and Data Mining

Lecture #15: Text - part II

C. Faloutsos

CMU SCS

Must-read Material

- Textbook, Chapter 6

CMU SCS

Optional (but terrific to read)

★ • McIlroy, M. D. (Jan. 1982). "Development of a Spelling List." IEEE Trans. on Communications COM-30(1): 91-99.

★ • Severance, D. G. and G. M. Lohman (Sept. 1976). "Differential Files: Their Application to the Maintenance of Large Databases." ACM TODS 1(3): 256-267.

CMU SCS

Outline

Goal: 'Find similar / interesting things'

- Intro to DB
- Indexing - similarity search
- Data Mining

CMU SCS

Indexing - Detailed outline

- primary key indexing
- secondary key / multi-key indexing
- spatial access methods
- fractals
- text
- multimedia
- ...

CMU SCS

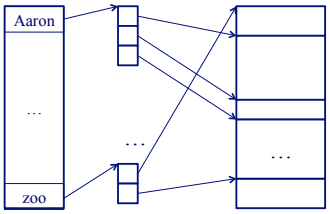
Text - Detailed outline

- text
  - problem
  - full text scanning
  - inversion
  - signature files
  - clustering
  - information filtering and LSI

CMU SCS

Text - Inversion

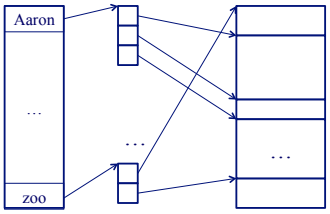
Dictionary Postings lists Text file



CMU SCS

Text - Inversion

Dictionary Postings lists Text file

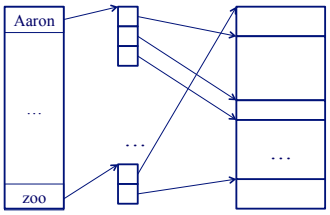


Q: space overhead?

CMU SCS

Text - Inversion

Dictionary Postings lists Text file



A: mainly, the postings lists

CMU SCS

Text - Inversion

- how to organize dictionary?
- stemming – Y/N?
- insertions?

CMU SCS

Text - Inversion

- how to organize dictionary?
  - B-tree, hashing, TRIEs, PATRICIA trees, ...
- stemming – Y/N?
- insertions?

CMU SCS

Text - Inversion

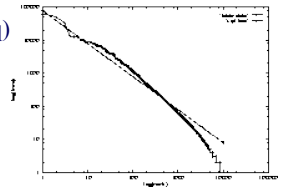
- variations:
- Parallelism [Tomasic+,93]
- Insertions [Tomasic+94], [Brown+]
  - 'zipf' distributions
- Approximate searching ('glimpse' [Wu+])

CMU SCS

Text - Inversion

- postings list – more Zipf distr.: eg., rank-frequency plot of 'Bible'

log(freq)



$$freq \approx \frac{1}{rank \ \ln(1.78V)}$$

log(rank)

CMU SCS

Text - Inversion

- postings lists
  - Cutting+Pedersen
    - (keep first 4 in B-tree leaves)
  - how to allocate space: [Faloutsos+92]
    - geometric progression
  - compression (Elias codes) [Zobel+] – down to 2% overhead!
  - Compression and doc reordering [Blandford+2002]

CMU SCS

Integer coding: small integers -> few bits

number	binary	Self-delimiting
2	10	00 1 10
3	11	00 1 11
15	1111	0000 1 1111

- *O (log (i))* bits for integer *i*
- can drop middle *'1'*
- can apply recursively, to length

CMU SCS

Document Reordering

	Doc1	Doc2	Doc3	…		
Aaron	1	0	1	0	1	0 …
zoo	0	0	0	1	0	1 …

CMU SCS

Document Reordering

	Doc1	Doc3	…			Doc2
Aaron	1	1	1	0	0	0 …
zoo	0	0	0	1	1	0 …

Shorter runs; easier to compress

CMU SCS

Conclusions

- Conclusions: needs space overhead (2%-300%), but it is the fastest

CMU SCS

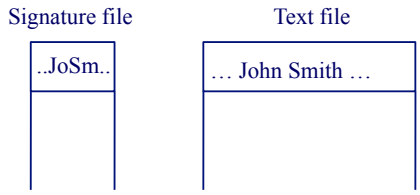
Text - Detailed outline

- text
  - problem
  - full text scanning
  - inversion
  - signature files
  - clustering
  - information filtering and LSI

CMU SCS

Signature files

- idea: 'quick & dirty' filter



CMU SCS

Signature files

- idea: 'quick & dirty' filter
- then, do seq. scan on sign. file and discard 'false alarms'
- Adv.: easy insertions; faster than seq. scan
- Disadv.: O(N) search (with small constant)
- Q: how to extract signatures?

CMU SCS

Signature files

- A: superimposed coding!! [Mooers49], ...

Word	Signature
data	001 000 110 010
base	000 010 101 001
doc.signature	001 010 111 011

m (=4 bits/word)
F (=12 bits sign. size)

CMU SCS

Signature files

- A: superimposed coding!! [Mooers49], ...

Word	Signature
data	001 000 110 010
base	000 010 101 001
doc.signature	001 010 111 011

data ↑ ↑↑ ↑

actual match

CMU SCS

Signature files

- A: superimposed coding!! [Mooers49], ...

Word	Signature
data	001 000 110 010
base	000 010 101 001
doc.signature	001 010 111 011

retrieval ↑ ↑ ↑ ↑

actual dismissal

CMU SCS

Signature files

- A: superimposed coding!! [Mooers49], ...

Word	Signature
data	001 000 110 010
base	000 010 101 001
doc.signature	001 010 111 011

nucleotic ↑ ↑ ↑ ↑

false alarm ('false drop')

CMU SCS

Signature files

- A: superimposed coding!! [Mooers49], ...

Word	Signature
data	001 000 110 010
base	000 010 101 001
doc.signature	001 010 111 011

'YES' is 'MAYBE'
'NO' is 'NO'

CMU SCS

Signature files

- Q1: How to choose F and m ?
- Q2: Why is it called 'false drop'?
- Q3: other apps of signature files?

CMU SCS

Signature files

- Q1: How to choose *F* and *m* ?

Word	Signature
data	001 000 110 010
base	000 010 101 001
doc.signature	001 010 111 011

m (=4 bits/word)
F (=12 bits sign. size)

CMU SCS

Signature files

- Q1: How to choose *F* and *m* ?
- A: so that doc. signature is 50% full

Word	Signature
data	001 000 110 010
base	000 010 101 001
doc.signature	001 010 111 011

m (=4 bits/word)
F (=12 bits sign. size)

CMU SCS

Signature files

- Q1: How to choose *F* and *m* ?
- Q2: Why is it called 'false drop'?
- Q3: other apps of signature files?

CMU SCS

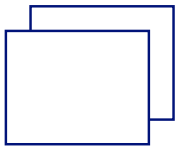
Signature files

- Q2: Why is it called 'false drop'?
- Old, but fascinating story [1949]
  - how to find qualifying books (by title word, and/or author, and/or keyword)
  - in O(1) time?
  - **without computers** (1949…)

CMU SCS

Signature files

- 'State of the art': cards – but > O(1)

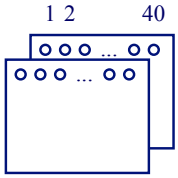


- one copy alpha by author
- one by title
- …

CMU SCS

Signature files

- Solution: edge-notched cards

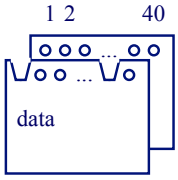


•each title word is mapped to *m* numbers (how?)
•and the corresponding holes are cut out:

CMU SCS

Signature files

- Solution: edge-notched cards

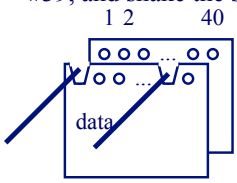


'data' -> #1, #39

CMU SCS

Signature files

- Search, e.g., for 'data': activate needle #1, #39, and shake the stack of cards!



'data' -> #1, #39

CMU SCS

Signature files

- Also known as 'zatocoding', from 'Zator' company.

CMU SCS

Signature files

- Q1: How to choose *F* and *m* ?
- Q2: Why is it called 'false drop'?
- Q3: other apps of signature files?

CMU SCS

Signature files

- Q3: other apps of signature files?
- A: anything that has to do with '**membership testing**': does '*data*' belong to the set of words of the document?

Word	Signature
data	001 000 110 010
base	000 010 101 001
doc.signature	001 010 111 011

CMU SCS

Signature files

- UNIX's early 'spell' system [McIlroy]
- Bloom-joins in System R* [Mackert+] and 'active disks' [Riedel99]
- differential files [Severance + Lohman]

Doug McIlroy

Eric Riedel

Guy Lohman

CMU SCS

App#1: Unix's spell

Dictionary

~30,000 words

aaron
apple
...
zoo

? ← *electroencephalogram*

What to do if the dictionary does not fit in memory (~1980)?

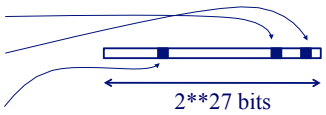
CMU SCS

App#1: Unix's spell

A: allow for a few typos! And use

- huge bit string (2**27)
- Hash each dictionary word to a bit
- Compress the string

aaron
apple
...
zoo



CMU SCS

App#1: Unix's spell

Sub-questions:

- Q1: How often do we allow typos?
- Q2: Will the (compressed) bit string fit in memory?
- Q3: How to compress the bit string?

CMU SCS

App#2: Bloom-joins

R @Chicago

A	B	C
1		
1		
3		
…		
1		
12		

S @NY

A	E	F
1		
18		
1		
…		
23		
2		

R join S (@PIT)

CMU SCS

App#2: Bloom-joins

R @Chicago

A	B	C
1		
1		
3		
…		
1		
12		

S @NY

A	E	F
1		
18		
1		
…		
23		
2		

Idea: reduce transmission cost: ‘R semijoin S’

CMU SCS

App#2: Bloom-joins

Idea: reduce transmission cost: ‘R semijoin S’

That is,
- ‘S’ ships its unique values of ‘A’
- ‘R’ deletes non-matching tuples
- (and they both send their tuples to PIT)

Q: what if we want to send at most, say 100 bytes NY -> Chicago?

CMU SCS

App#2: Bloom-joins

Q: what if we want to send at most, say 100 bytes NY -> Chicago?

A: Bloom-join! Send a bloom filter of the S.A values

CMU SCS

App#3: Differential files

Problem definition:

- A large file (eg., with EMPLOYEE records), nicely packed and organized (eg., B-tree)
- A few insertions/deletions, that we would like to keep separate, and merge, at night
- How to search, eg., for employee #123?

CMU SCS

App#3: Differential files

ssn	name	…
1		
5		
12		
…		
503		
509		

flag	ssn	name, ..
i	123	
i	55	
d	17	
d	33	

Differential file

CMU SCS

App#3: Differential files

- Q: How to search, eg., for employee #123?
- A: bloom-filter, for keys of diff. file

- Q: What are the advantages of differential files?
- A: <see paper, for 10(!) of them>

CMU SCS

Signature files - conclusions

- easy insertions; slower than inversion
- brilliant idea of 'quick and dirty' filter: quickly discard the vast majority of non_qualifying elements, and focus on the rest.

CMU SCS

References

- Blandford, D. and Blelloch, G. 2002. *Index Compression through Document Reordering*. Data Compression Conference (DCC '02) (April 02 - 04, 2002).
- Brown, E. W., J. P. Callan, et al. (March 1994). *Supporting Full-Text Information Retrieval with a Persistent Object Store*. EDBT conference, Cambridge, U.K., Springer Verlag.

CMU SCS

References - cont'd

- Faloutsos, C. and H. V. Jagadish (Aug. 23-27, 1992). On B-tree Indices for Skewed Distributions. 18th VLDB Conference, Vancouver, British Columbia.
- Karp, R. M. and M. O. Rabin (March 1987). "Efficient Randomized Pattern-Matching Algorithms." IBM Journal of Research and Development 31(2): 249-260.
- Knuth, D. E., J. H. Morris, et al. (June 1977). "Fast Pattern Matching in Strings." SIAM J. Comput 6(2): 323-350.

CMU SCS

References - cont'd

- Mackert, L. M. and G. M. Lohman (August 1986). R* Optimizer Validation and Performance Evaluation for Distributed Queries. Proc. of 12th Int. Conf. on Very Large Data Bases (VLDB), Kyoto, Japan.
- Manber, U. and S. Wu (1994). GLIMPSE: A Tool to Search Through Entire File Systems. Proc. of USENIX Techn. Conf.
- ★ McIlroy, M. D. (Jan. 1982). "Development of a Spelling List." IEEE Trans. on Communications COM-30(1): 91-99.

CMU SCS

References - cont'd

- Mooers, C. (1949). Application of Random Codes to the Gathering of Statistical Information Bulletin 31. Cambridge, Mass, Zator Co.
- Pedersen, D. C. a. J. (1990). Optimizations for dynamic inverted index maintenance. ACM SIGIR.
- Riedel, E. (1999). Active Disks: Remote Execution for Network Attached Storage. ECE, CMU. Pittsburgh, PA.

CMU SCS

References - cont'd

- Severance, D. G. and G. M. Lohman (Sept. 1976). "Differential Files: Their Application to the Maintenance of Large Databases." ACM TODS 1(3): 256-267.
- Tomasic, A. and H. Garcia-Molina (1993). Performance of Inverted Indices in Distributed Text Document Retrieval Systems. PDIS.
- Tomasic, A., H. Garcia-Molina, et al. (May 24-27, 1994). Incremental Updates of Inverted Lists for Text Document Retrieval. ACM SIGMOD, Minneapolis, MN.

CMU SCS

References - cont'd

- Wu, S. and U. Manber (1992). "AGREP- A Fast Approximate Pattern-Matching Tool." .
- Zobel, J., A. Moffat, et al. (Aug. 23-27, 1992). An Efficient Indexing Technique for Full-Text Database Systems. VLDB, Vancouver, B.C., Canada.