Data Structures and Methods

Johan Bollen
Old Dominion University
Department of Computer Science

jbollen@cs.odu.edu
http://www.cs.odu.edu/~jbollen
Lecture Objectives

1. To this point: models
   (a) Information Retrieval Techniques
   (b) General models and approaches

2. Underneath general model
   (a) Text search procedures
   (b) Data structures for retrieval

3. This lecture:
   (a) String Matching
   (b) Data structures
   (c) Text retrieval
   (c) Storage procedures and optimizations
String matching: Similarity

1. Definition of distance measure between strings
2. Definition string:
   (a) Assume bounded size alphabet
   (b) Sequence of finite length from alphabet
3. Distance: any function for which we can say
   (a) Reflexive: \( d(s_1, s_1) = 0 \)
   (b) Symmetric: \( d(s_1, s_2) = d(s_2, s_1) \)
   (c) Triangular inequality:
      \[ d(s_1, s_3) \leq d(s_1, s_2) + d(s_2, s_3) \]
4. Often used:
   (a) Hamming distance: number of different characters
   (b) “edit” distance or Levenshtein
      i. Substitutions
      ii. Deletions
      iii. Insertions
   (c) Total “cost” to turn on string into another (demo)
Hamming and Levenshtein

\[ d("tree","frie") = 2 \]
\[ d("tree","free") = 1 \]

\[ d("maples","qpple") = 4 \]
Other String matching methods

1. Automatons
   (a) different way to represent Levenshtein distance calculation
   (b) same principle

2. Regular Expressions
   (a) Represented as automaton

3. In spite of string matching methods
   (a) We require mechanism of efficient storage and retrieval
   (b) Able to operate on large scale
Text Storage and Retrieval

1. Data Structures
   (a) Inverted Files or indexes
   (b) PAT trees and arrays (Suffix trees)
   (c) Signature Files

2. Used for specific retrieval purposes
   (a) Inverted files: term-document
   (b) Signature files: term-document
   (c) PAT trees and arrays: string matching
Inverted Files

1. Most commonly applied
2. Quite simple and efficient
3. Main principles
   (a) Index term extracted or defined
   (b) Store index term with list of occurrence locations
   (c) Retrieval of relevant text segments

4. Extensions:
   (a) block addressing
   (b) Posting files
   (c) Use of data structures:
       i. B-trees
       ii. Tries
Inverted Files

Index File

Postings

Documents File

term DF Link
information 3

DocID Link
ID

Document 1

Document 3
Example

Edgar Allen Poe’s The Raven (1845)
First paragraph:

1) Once upon a **midnight** dreary, while I pondered, weak and weary,
2) Over many a quaint and curious **volume** of forgotten **lore**, 
3) While I nodded, nearly napping, suddenly there came a **tapping**, 
4) As of some one gently rapping, rapping at my **chamber door**. 
5) “’Tis some **visiter,**” I muttered, ”tapping at my **chamber door** -
6) Only this, and **nothing** more.”

**Hand selected keyterms - document frequency:**
1) chamber-2, 2) door-2, 3) lore-1, 4) midnight-1,
   5) nothing-1, 6) tap-1, 7) visiter-1, 8) volume-1
Example, cont’d

Index File

<table>
<thead>
<tr>
<th>term</th>
<th>DF</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>chamber</td>
<td>2</td>
<td>4,5,2</td>
</tr>
<tr>
<td>door</td>
<td>2</td>
<td>4,5</td>
</tr>
<tr>
<td>lore</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>midnight</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>nothing</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>tap</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>visiter</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>volume</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Documents File

1. Once upon a midnight dreary
2. While I pondered, weak and weary,
3. Over many a quaint and curious
4. Volume of forgotten lore, while I
5. Nodded, nearly napping, suddenly
6. There came a tapping, as of one
7. Gently rapping, rapping at my
8. Chamber door, "tis some visiter",
9. I muttered, "tapping at my
10. Chamber door—only this and
11. Nothting more"
Issues

1. Overhead:
   (a) Construction of additional files: index, posting
   (b) Storage may be issue

2. Updates
   (a) Expensive procedure
   (b) Requires re-scan of text files
   (c) Merging existing index files?

3. Restricted vocabulary
   (a) Term matches in index file
   (b) Partial Matching procedures?
      i. hamming
      ii. edit distance
Extensions and data structures for inverted index files

1. Sorted Array
   (a) Array containing terms, document frequency, and document links
   (b) Search of array: binary search
   (c) Sorted for efficient searching
   (d) Expensive to update, but simple and efficient

2. B-trees
   (a) Fast searches
   (b) Essentially replaces sorted array
   (c) Tree leaves link into posting file

3. Tries
   (a) Digital decomposition
   (b) Patricia Tree replaces B-tree
   (c) Internal node indicate bit skips
Example prefix B-tree
PAT trees, suffix trees and suffix arrays

1. Need to move beyond constraints of fixed vocabulary

2. Inverted Indexes:
   (a) Dictionary determines search
   (b) Any string outside dictionary or that can not be reduced to dictionary string?
   (c) full text search?

3. Solution: suffix trees and arrays
   (a) Allow searching on any string
   (b) Matching string and document locations can be updated
   (c) Efficient file structure
   (d) Based on concept of Semi Infinite Strings or SIS
Example Suffix Trie

Once upon a midnight dreary while I pondered, weak and weary, over many a quaint
midnight dreary while I pondered, weak and weary, over many a quaint
dreary while I pondered, weak and weary, over many a quaint
while I pondered, weak and weary, over many a quaint
pondered, weak and weary, over many a quaint
weak and weary, over many a quaint

Suffix Trie

```
1  6  13  22  29  37  47  52  56  63  68  75
```

Suffix Tree

```
aro  i  midnight  13
     w  a  many  68
      1  k  weak  47
       3  y  weary  56
```

Suffix Array

```
13  68  47  56
```
Suffix array retrieval

1. Search operations:
   (a) Prefix searching
   (b) Range searching

2. Prefix searching:
   (a) Simply traverse PAT tree from root
   (b) Subtree leafs are all documents with prefix

3. Range Searching
   (a) Determine limiting pattern
   (b) Retrieve subtree leads defined by limiting patterns

4. Does this work for suffix arrays?
Signature Files

Use of hashing functions

1. Storage
   (a) Use of hashing function
   (b) Text is cut in blocks
   (c) All words in blocks are mapped to bit masks
   (d) Bit masks for block = bitwise OR of all word bit masks

2. Retrieval
   (a) Query term is mapped to bit mask
   (b) Query term bit mask is bitwise

   AND with all block bitmaps
   (c) match? text block is retrieved and checked

   Issue of false drops

1. Block bitmaps may correspond to query bitmap
   (a) Result of bitwise OR of all word bitmaps
   (b) Same bit pattern may be created
   (c) False match

2. Careful tuning of parameters required
Signature Files

Once upon a midnight dreary

while I pondered, weak and weary,
over many a quaint

Block 1

111011

h(midnight) = 111011
h(dreary) = 010010

q = "midnight"

Block 2

101010

h(while) = 101010
h(pondered) = 101000

Block 3

110101

h(weak) = 000101
h(weary) = 110100

Block 4

111111

h(over) = 011110
h(many) = 010000
h(quaint) = 111110
Signature Files: False drops

**Bit masks**

1. Determined number of bits
2. Corresponds to expected false drop probability

**Balance**

1. Large bit masks: overhead
2. Can be significant problem for large text blocks
3. Or Large data sets
4. Choice = accepted levels of false drops

and overhead

**Empirical Values**

1. 10% overhead $\sim$ 2% false drop rate
2. 20% overhead $\sim$ 0.046% false drop rate

**General Conclusion**

1. Efficient production
2. Easy updating
3. Inverted indexes are most often used