Lecture 7:
Text Retrieval: 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: \( d(s_1, s_2) = d(s_2, s_1) \)

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)

(c) Triangular inequality:

\[
d(s_1, s_3) \leq d(s_1, s_2) + d(s_2, s_3)
\]
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

   (b) Allows user to specify class of index terms

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
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, tapping at my chamber door.
5) “’Tis some visitor, ” 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</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>

#### Postings

<table>
<thead>
<tr>
<th>DocID</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>6,7,8</td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></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. nothing 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

3. Tries
   (a) Digital decomposition
   (b) Patricia Tree replaces B-tree
   (c) Internal node indicate bit skips
Example prefix B-tree

```
IR
  / | |
 nf nt  etr tur
 /     /   |
information 2 retrieval 3
```

posting file
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
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 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
IR evaluation

Why we must evaluate?
1. Myriad of possible IR systems
2. Theoretical and logical appeal
3. User preferences and communities
4. Need for metric and benchmarks to determine relative effectiveness

Golden standard: relevancy
1. IR: query → results → relevancy?
2. Who determines relevancy?

(a) User?
(b) Expert?

3. What is relevancy?

Need for formalization of metrics based on relevancy
1. Able to deal with different queries, collections
2. Objective and quantitative metric
3. Simple and intuitive
Concept of Precision and Recall

Two accepted meanings

1. Your favorite DJ’s
2. Two metrics of IR performance
   Collection v. query
   1. Given we have a query
   2. Given we have a collection
   3. We have a set of documents that are all relevant to query

(a) Relevancy remains to be defined
(b) Let’s assume we have an adequate understanding

4. Our IR system returns its results
5. Two questions
   (a) How many irrelevant results?
   (b) How well do results cover set of all relevant results in collection?
Relevancy formalized
Precision and Recall

Let us denote a user information request as \( I \) and the set of document relevant to that request as \( R \).

For any request \( I \), a set of results is returned: \( A \)

The cardinality of \( A \) and \( R \) is denoted: \(|A|\) and \(|R|\).

We denote the set of relevant results in \( A \) as \( R_a \), so \( R_a \subseteq A \).

\[
\text{Recall} \\
\text{recall} = \frac{|R_a|}{|R|}
\]

or the ratio of relevant results in the answer set over the total number of possibly relevant results for \( I \) in the collection

\[
\text{Precision} \\
\text{precision} = \frac{|R_a|}{|A|}
\]

or the ratio of relevant results over all results in the answer set
Precision and Recall

Precision and Recall
Precision and Recall

<table>
<thead>
<tr>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Represents how many relevant results returned by IR system</td>
<td>1. Represents how many relevant items were captured in toto by IR system</td>
</tr>
<tr>
<td>2. Indicates quality of results</td>
<td>2. Indicates coverage of results</td>
</tr>
<tr>
<td>3. Corresponds to false positives</td>
<td>3. Corresponds to false negatives</td>
</tr>
</tbody>
</table>

**Precision v. Recall**

1. IR system is a little like fishing
2. Fine net catches a lot of small and large fish: many of them not good catch, but you catch most fish
3. Coarse net catches only large fish: but many fish not caught
4. You can not have both fine and coarse net!
Determining Precision and Recall figures

Assume set of queries: for each query we can determine precision and recall.
Assume a set of queries $Q = \{q_1, q_2, \ldots, q_t\}$. The ideal and complete answer set of all relevant documents for query $q_1$ is:

$$R_{q_1} = \{d_7, d_5, d_6, d_{20}, d_{25}\}$$

We invent an IR model, and apply it to the collection $R, R_{q_1} subset R$ using It produces a ranking of results:

1* $d_7$
2   $d_3$
3* $d_{20}$
4   $d_1$
5   $d_{662}$
6   $d_{429}$
7* $d_{25}$
8   $d_{99}$
9   $d_{723}$
10* $d_6$

We take into account the ranking produced by the IR system because the user examines results in this order.

Result 1: $d_7 \in R_{q_1}$, precision=100%, recall=20%
Result 3: $d_{20} \in R_{q_1}$, precision=66%, recall=40%
Result 5: $d_{25} \in R_{q_1}$, precision=43%, recall=60%
Result 9: $d_6 \in R_{q_1}$, precision=40%, recall=80%

Above 80% recall, precision=0
Precision at standard recall levels

1. Notion of precision at recall levels
   (a) Given that we accept a certain recall level
   (b) What will the precision of a given IR system be?
   (c) Standard levels: 0, 10, 20, ..., 100 (11 levels)

2. PR calculated over range of queries
   (a) Requirement to extrapolate values
   (b) Aggregate different precision values for level of recall

3. Averaging procedures and aggregate PR numbers
Average precision

Assume $P(r)$ indicates average precision at recall level $r$

$P_i(r)$ represents precision at recall level $r$ for query $i$.

then:

$$\overline{P(r)} = \sum_{i=1}^{N_q} \frac{P_i(r)}{N_q}$$

Need to extrapolate recall levels

For any given query, recall level may not be precisely within standard level.

Use maximum of neighboring recall levels.
Single Value Summaries

1. We often need to know how an IR system performs for single queries
   (a) Reveal exceptions and specific problems
   (b) Compare different IR systems over set of given queries

2. Averaging at relevant results for given query
   (a) Sample precision at relevant results in ranking
   (b) Average values
   (c) Encourages systems that rank relevant results highly

3. R-precision
   (a) Look at total number of relevant documents, $|R_q|$ for query $q$
   (b) Sample precision at ranking in R-th position

4. Precision histograms
   (a) Compare R-precision for different queries across different IR systems
   (b) Subtract R-precision system 1 from R-precision system 2
   (c) Allows comparison of two system for same set for queries
Notes on Precision and Recall

1. excellent DJs

2. validity problems
   (a) Relevance:
      i. Subjective
      ii. Depends on more than term matches
      iii. Different to consistently evaluate across queries
   (b) All or nothing
      i. Set of relevant documents is not so crisp
      ii. Precision and Recall do not take into account degree of relevancy
          iii. Actual objective should be to compare degree of relevancy for query or IR system and for external relevancy estimator
          iv. Novelty and serendipity: butterfly effect on user satisfaction
   (c) Scale
      i. Large collections: difficult to determine all relevant documents
      ii. Difficult to construct IR evaluation framework
Standardized evaluation framework: Reference collections

1. Lack of consistent frameworks for PR evaluation

2. Different systems, collections, queries
   - **TREC - Text Retrieval Conference**
     - 1. Started in 1992, Conference format
     - 2. Test-collection sampled from variety of sources, 5.8gb and retrieval tasks

3. Set of information requests

4. Set of relevant results determined by “pooling method”

5. Set of statistics collected for each system
   - **CACM and ISI collections**

   1. **CACM**
      - (a) Communications of the ACM (1979)
      - (b) Data includes categories, references, etc
      - (c) 52 test information requests

   2. **ISI**
      - (a) Sample of Institute of Scientific Information collection
      - (b) 25 test information requests
Readings for next class