Data structures in Information Retrieval

Max Gubin
mail@maxgubin.com
Information Retrieval History

4000 BC

1950

2000
Information Retrieval Tasks

Types of information:

- Text
- Sound
- Image

Types of tasks:

- Search
- Classification/clustering
- Extraction/Summarization

Mixes…
Toy project

Let’s create a toy search engine:
Course Outline

• Introduction (the problem definition)
• Basics (structures and environments)
• Building index
• Search!
• Other data: Language Models and Link Graphs
Hierarchy of data in text IR

Collection

Document

Field1

Field2

Field3

Word A  Word B  Word C

Word D  Word E
Linearization (word extraction)

“To”, 1, Body, Document1
“BE”, 2, Body, Document1
“or”, 3, Body, Document1
“not”, 4, Body, Document1
“to”, 5, Body, Document1
“be”, 6, Body, Document1
Document formats

- Presentation oriented (PDF, RTF)
- Structure Oriented (SGML, HTML, XML)
Encodings

• **Present all letters** of the alphabet
• **Collation** (case) – can be complex in some languages: a A ä Ä;

Official standard Unicode
Latest version 5.10 about 100000 characters:
Character codes (codepoints 0 10FFFF)
Encoding rules (utf-8, utf-16, utf-32)
Algorithms
Words

• Morphology agglutinative, multiroot,
• Abbreviations
• Spelling variants
• Stop-words

How to handle:
1. During document analysis
2. During search
Linearization (complex)
Naïve Scan (grep approach)

- Have the whole context for analysis
- Match current hardware architecture
- Usually can be easily parallelized
Adding index

Two meanings of index:

- Taxonomy that accelerates human search
- **Special data structure** that accelerate data access
Using Standard Database

<table>
<thead>
<tr>
<th>Word</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>to</td>
<td>1</td>
</tr>
<tr>
<td>be</td>
<td>2</td>
</tr>
<tr>
<td>not</td>
<td>4</td>
</tr>
<tr>
<td>or</td>
<td>3</td>
</tr>
</tbody>
</table>

```
SELECT DocTable.Document FROM Dictionary, Doctable, Positions
```
Bag of words

<table>
<thead>
<tr>
<th>Word</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>to</td>
<td>1</td>
</tr>
<tr>
<td>be</td>
<td>2</td>
</tr>
<tr>
<td>not</td>
<td>4</td>
</tr>
<tr>
<td>or</td>
<td>3</td>
</tr>
</tbody>
</table>

Dictionary

<table>
<thead>
<tr>
<th>WordID</th>
<th>DocID</th>
<th>Flags</th>
<th>Fields</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>CAP</td>
<td>BODY</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>CAP</td>
<td>BODY</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td></td>
<td>BODY</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td></td>
<td>BODY</td>
<td>1</td>
</tr>
</tbody>
</table>

Doctable

<table>
<thead>
<tr>
<th>Document</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hamlet</td>
<td>1</td>
</tr>
<tr>
<td>Introduction to...</td>
<td>2</td>
</tr>
<tr>
<td>Dive into Python</td>
<td>3</td>
</tr>
</tbody>
</table>
Problems with General Purpose Databases

1. Size
2. Speed build
3. Speed search

This is a tool for another task
Matrix representation

Simple example

1. Dad is reading a book
2. Mom is watching TV
3. Dad and Mom are at home

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>and</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>are</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>at</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>book</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Dad</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Mom</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>is</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>reading</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>home</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>TV</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Main IR structure

A sparse n-dimensional matrix in different presentations is “THE MAIN IR STRUCTURE”

Search – inverted index
Language models – table of probabilities
Link analysis – Adjacency matrix
Spareness of the matrix

Example:
N - 1 mln documents
Ds - 1000 words/document
D – 500 000 words in dictionary

\[ |\text{Word/Document matrix}| = D*N = 500 \text{ bln} \]
Words in collection = 1 mln * 1000 = 1 bln
Only 0.2% elements in the matrix are not 0
Inverted file

Dictionary

Dad

Mom

Posting lists

1,3

2,3

TV

2
Signature file

Signatures for words (function)

<table>
<thead>
<tr>
<th>Word</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dad</td>
<td>000000001</td>
</tr>
<tr>
<td>Mom</td>
<td>00001000</td>
</tr>
<tr>
<td>TV</td>
<td>10000000</td>
</tr>
<tr>
<td>watching</td>
<td>00001000</td>
</tr>
<tr>
<td>football</td>
<td>00001000</td>
</tr>
</tbody>
</table>

Doc Signature = OR words

<table>
<thead>
<tr>
<th></th>
<th>00110001</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>00110001</td>
</tr>
<tr>
<td>2</td>
<td>01011000</td>
</tr>
<tr>
<td>3</td>
<td>10001001</td>
</tr>
</tbody>
</table>
Signature file (Search)

Query = “Mom Dad”
q_s = 00001001

for doc in Document_Signatures:
    if doc.signature & q_s = q_s:
        ScanDocument(doc.id)

An old structure = hash + bloom filter + scan
IR Packages

• Lucene (http://lucene.apache.org/)
• Terrier (http://ir.dcs.gla.ac.uk/terrier/)
• Lemur & Indri (http://www.lemurproject.org/)
• Zettair (http://www.seg.rmit.edu.au/zettair/)
• Zebra (http://www.indexdata.dk/zebra/)
Search speed

- Inverted File
- Signature file
- Naïve Scan

Collection size vs. Search speed
Speed (Size) depends on

- Algorithm
- Size of data
- Hardware
Algorithm complexity

• Storage complexity (How much memory we need)
• Time complexity (How many operations we need)
$O(f(n))$ notation

$x(n)$ is $O(f(n))$ if $x(n) \leq C \cdot f(n)$, $C$ – const $n \to \infty$
Structure characteristics

• **Theoretical**: Processing algorithm complexity

• **Practical**:  
  – Memory access pattern  
  – Parallelization
Summary

• IR is old 😊
• Main Structure is sparse matrix
• Index = Inverted file
• Speed & Size
Q&A