Chapter IR:III

III. Indexing

- Indexing Basics
- Inverted Index
- Query Processing I
- Query Processing II
- Index Construction
- Index Compression

Inverted Index

Term-Document Matrix

	d_1	d_2	d_3	d_4	d_5	$\cdots$
t_1						
t_2						
t_3						
t_4						
t_5						
$\vdots$						$\ddots$

Inverted Index

Term-Document Matrix

	d_1	d_2	d_3	d_4	d_5	$\cdots$
t_1						
t_2						
t_3						
t_4						
t_5						
$\vdots$						$\ddots$

- Documents D

  d_1 Antony and Cleopatra
  d_2 Julius Caesar
  d_3 The Tempest
  d_4 Hamlet
  d_5 Othello

- Index terms T

  t_1 Antony
  t_2 Brutus
  t_3 Caesar
  t_4 Calpurnia
  t_5 Cleopatra

Inverted Index

Term-Document Matrix

	d_1	d_2	d_3	d_4	d_5	$\cdots$
t_1	1					
t_2	1					
t_3	1					
t_4	0					
t_5	1					
$\vdots$						$\ddots$

- Documents D
  - d_1 Antony and Cleopatra
  - d_2 Julius Caesar
  - d_3 The Tempest
  - d_4 Hamlet
  - d_5 Othello
- Index terms T
  - t_1 Antony
  - t_2 Brutus
  - t_3 Caesar
  - t_4 Calpurnia
  - t_5 Cleopatra

Inverted Index

Term-Document Matrix

	d_1	d_2	d_3	d_4	d_5	$\cdots$
t_1	1	1	0	0	0	
t_2	1	1	0	1	0	
t_3	1	1	0	1	1	
t_4	0	1	0	0	0	
t_5	1	0	0	0	0	
$\vdots$						$\ddots$

- Documents D

  d_1 Antony and Cleopatra
  d_2 Julius Caesar
  d_3 The Tempest
  d_4 Hamlet
  d_5 Othello

- Index terms T

  t_1 Antony
  t_2 Brutus
  t_3 Caesar
  t_4 Calpurnia
  t_5 Cleopatra

Inverted Index

Term-Document Matrix

	d_1	d_2	d_3	d_4	d_5	$\cdots$
t_1	382	128	0	0	0	
t_2	4	379	0	1	0	
t_3	289	272	0	2	1	
t_4	0	16	0	0	0	
t_5	271	0	0	0	0	
$\vdots$						$\ddots$

- Documents D

  d_1 Antony and Cleopatra
  d_2 Julius Caesar
  d_3 The Tempest
  d_4 Hamlet
  d_5 Othello

- Index terms T

  t_1 Antony
  t_2 Brutus
  t_3 Caesar
  t_4 Calpurnia
  t_5 Cleopatra

Inverted Index

Term-Document Matrix

	d_1	d_2	d_3	d_4	d_5	$\cdots$
t_1	$w_{1,1}$	$w_{1,2}$	$w_{1,3}$	$w_{1,4}$	$w_{1,5}$	
t_2	$w_{2,1}$	$w_{2,2}$	$w_{2,3}$	$w_{2,4}$	$w_{2,5}$	
t_3	$w_{3,1}$	$w_{3,2}$	$w_{3,3}$	$w_{3,4}$	$w_{3,5}$	
t_4	$w_{4,1}$	$w_{4,2}$	$w_{4,3}$	$w_{4,4}$	$w_{4,5}$	
t_5	$w_{5,1}$	$w_{5,2}$	$w_{5,3}$	$w_{5,4}$	$w_{5,5}$	
$\vdots$						$\ddots$

- Documents D
  - d_1 Antony and Cleopatra
  - d_2 Julius Caesar
  - d_3 The Tempest
  - d_4 Hamlet
  - d_5 Othello

- Index terms T
  - t_1 Antony
  - t_2 Brutus
  - t_3 Caesar
  - t_4 Calpurnia
  - t_5 Cleopatra

Inverted Index

Term-Document Matrix

	d_1	d_2	d_3	d_4	d_5	$\cdots$
t_1	$w_{1,1}$	$w_{1,2}$	$w_{1,3}$	$w_{1,4}$	$w_{1,5}$	
t_2	$w_{2,1}$	$w_{2,2}$	$w_{2,3}$	$w_{2,4}$	$w_{2,5}$	
t_3	$w_{3,1}$	$w_{3,2}$	$w_{3,3}$	$w_{3,4}$	$w_{3,5}$	
t_4	$w_{4,1}$	$w_{4,2}$	$w_{4,3}$	$w_{4,4}$	$w_{4,5}$	
t_5	$w_{5,1}$	$w_{5,2}$	$w_{5,3}$	$w_{5,4}$	$w_{5,5}$	
$\vdots$						$\ddots$

- Documents D
  - d_1 Antony and Cleopatra
  - d_2 Julius Caesar
  - d_3 The Tempest
  - d_4 Hamlet
  - d_5 Othello

- Index terms T
  - t_1 Antony
  - t_2 Brutus
  - t_3 Caesar
  - t_4 Calpurnia
  - t_5 Cleopatra

- Term Weights
  - Boolean
  - Term frequency
  - ...

Inverted Index

Term-Document Matrix

	d_1	d_2	d_3	d_4	d_5	$\cdots$
t_1	$w_{1,1}$	$w_{1,2}$	$w_{1,3}$	$w_{1,4}$	$w_{1,5}$	
t_2	$w_{2,1}$	$w_{2,2}$	$w_{2,3}$	$w_{2,4}$	$w_{2,5}$	
t_3	$w_{3,1}$	$w_{3,2}$	$w_{3,3}$	$w_{3,4}$	$w_{3,5}$	
t_4	$w_{4,1}$	$w_{4,2}$	$w_{4,3}$	$w_{4,4}$	$w_{4,5}$	
t_5	$w_{5,1}$	$w_{5,2}$	$w_{5,3}$	$w_{5,4}$	$w_{5,5}$	
$\vdots$						$\ddots$

Observations:

- Most retrieval models induce a term-document matrix by computing term weights $w_{i,j}$ for each pair of term $t_i \in T$ and document $d_j \in D$.
- Query-independent computations that only depend on D are done offline.
- Online, given a query q, the term weights required are looked up to score documents.

Inverted Index

Term-Document Matrix

	d_1	d_2	d_3	d_4	d_5	$\cdots$
t_1	$w_{1,1}$	$w_{1,2}$				
t_2	$w_{2,1}$	$w_{2,2}$		$w_{2,4}$		
t_3	$w_{3,1}$	$w_{3,2}$		$w_{3,4}$	$w_{3,5}$	
t_4		$w_{4,2}$				
t_5	$w_{5,1}$					
$\vdots$						$\ddots$

Observations:

- The size of the term-document matrix is $|T| \cdot |D|$.
- The term-document matrix is sparse: the vast majority of term weights are 0.
- Therefore, most of the storage space required for the full matrix is wasted.
- Using a sparse-matrix representation yields significant space savings.

Inverted Index

Data Structure

T	$\rightarrow$	**Postings** (Posting Lists, Postlists)			
t_1	$\rightarrow$	$\boxed{d_1, w_{1,1}}$	$\boxed{d_2, w_{1,2}}$		
t_2	$\rightarrow$	$\boxed{d_1, w_{2,1}}$	$\boxed{d_2, w_{2,2}}$	$\boxed{d_4, w_{2,4}}$	
t_3	$\rightarrow$	$\boxed{d_1, w_{3,1}}$	$\boxed{d_2, w_{3,2}}$	$\boxed{d_4, w_{3,4}}$	$\boxed{d_5, w_{3,5}}$
t_4	$\rightarrow$	$\boxed{d_2, w_{4,2}}$			
t_5	$\rightarrow$	$\boxed{d_1, w_{5,1}}$			
$\vdots$					

An index is implemented as a multimap (i.e., a hash table with multiple values).

Components of an externalized implementation:

- Term vocabulary file
  Lookup table which maps terms $t_i \in T$ to the start of their posting list in the postings file.
- Postings file(s)
  File(s) that store posting lists on disk.
- Index entries $\boxed{d_i, [\ldots]}$, so-called postings

Inverted Index

Data Structure

T	$\rightarrow$	**Postings** (Posting Lists, Postlists)			
t_1	$\rightarrow$	$\boxed{d_1, w_{1,1}}$	$\boxed{d_2, w_{1,2}}$		
t_2	$\rightarrow$	$\boxed{d_1, w_{2,1}}$	$\boxed{d_2, w_{2,2}}$	$\boxed{d_4, w_{2,4}}$	
t_3	$\rightarrow$	$\boxed{d_1, w_{3,1}}$	$\boxed{d_2, w_{3,2}}$	$\boxed{d_4, w_{3,4}}$	$\boxed{d_5, w_{3,5}}$
t_4	$\rightarrow$	$\boxed{d_2, w_{4,2}}$			
t_5	$\rightarrow$	$\boxed{d_1, w_{5,1}}$			
$\vdots$					

An index is implemented as a multimap (i.e., a hash table with multiple values).

Design choices:

- Information stored in a posting $\boxed{d_i, [\ldots]}$.
- Ordering of each term's posting list.
- Encoding and compression techniques for further space savings.
- Physical implementation details, such as external memory and distribution.

Inverted Index

Posting

Given term t and document d, their posting may include the following:

```
<document> [<weights>] [<position>] [...]
```

`<document>`:

- Reference to the document d in which term t occurs (or to which it applies).

`<weights>`:

- Term weight w for term t in document d.
- Often, only basic term weights are stored (e.g., term frequency *tf*(t, d)). Storing model-specific weights saves space at the expense of flexibility.

`<position>`:

- Term positions within the document, i.e., term, sentence, page, chapter, etc.
- Field information, e.g., title, author, introduction, etc.

Inverted Index

Posting

Two special-purpose entries are distinguished:

`...<list length>`

`...<skip pointer>`

`<list length>`:

- Added to the first entry of the posting list of a term t.
- Stores the length of the posting list.
- What does the length of a posting list indicate?

`<skip pointer>`:

- Used to implement a skip list in a term's posting list, when ordered by ID.
- Allows for random access to postings in $O(\log df(t, D))$.
- Second entry of a posting list, and then at random (or regular) intervals. An effective amount of skip entries has been found to be $\sqrt{df(t, D)}$.

Inverted Index

Posting

Two special-purpose entries are distinguished:

...<list length>	...<skip pointer>

`<list length>`:

- Added to the first entry of the posting list of a term t.
- Stores the length of the posting list.
- Equals the number of documents containing t (document frequency $\mathit{df}(t, D)$).

`<skip pointer>`:

- Used to implement a skip list in a term's posting list, when ordered by ID.
- Allows for random access to postings in $O(\log \mathit{df}(t, D))$.
- Second entry of a posting list, and then at random (or regular) intervals.
  An effective amount of skip entries has been found to be $\sqrt{\mathit{df}(t, D)}$.

Inverted Index

Posting List, Postlist

Example for two posting lists, where for term t_i postings $\boxed{k, \textit{tf}(t_i, d_k)}$ are stored:

T	**Postings**										
⋮											
t_i	$2, 4$	$4, 9$	$8, 2$	$16, 1$	$19, 7$	$23, 5$	$28, 6$	$41, 8$	$50, 6$	$77, 8$	$\ldots$
t_j	$1, 1$	$2, 3$	$3, 5$	$5, 2$	$8, 17$	$41, 6$	$51, 5$	$60, 5$	$71, 3$	$77, 2$	$\ldots$
⋮											

Ordering:

- by document identifier. Problem: good documents randomly distributed.
- ➜ by document quality. Early termination, but re-assigning IDs necessary.
- ➜ by term weight. Early termination, but renders skip lists useless.

Compression:

- The size of an index is in $O(|D|)$, where $|D|$ denotes the disk size of D.
- Posting lists can be effectively compressed with tailored techniques.

Remarks:

- The name “inverted index” is redundant: an index always maps terms to (parts of) documents where they occur. Better suited, but used less often, is “inverted file”, which conveys that a (document) file is inverted to form an index.
- Some retrieval models do not assign zero weights, but default to non-zero weights instead. Such weights can be omitted from an inverted index as well; they can be stored as a constant and used whenever a term weight for a given term-document pair is required that is not present in the inverted index.
- There is a tradeoff between the amount of information stored in a posting, and the time it takes to process a posting list in search of a document. The more information is stored in a posting, the more must be decoded or at least loaded into memory during postlist traversal.
- A skip entry may include more than one pointer, allowing for skip steps of various lengths.
- Dependent on the search scenario, constructing more than one index with different properties may be beneficial.