Chapter IR:IV

IV. Indexes

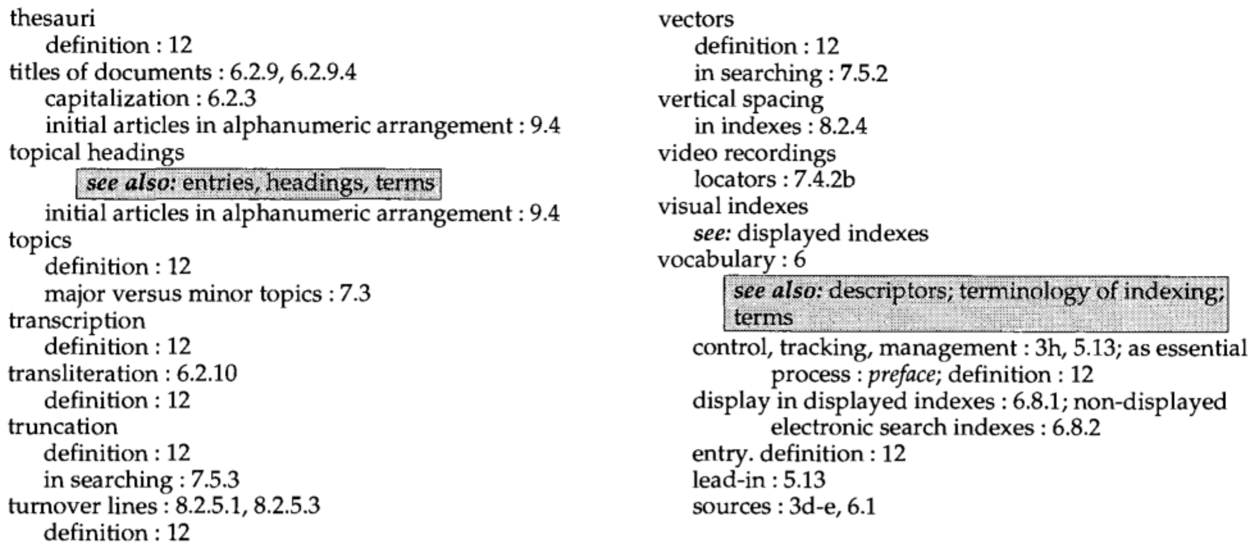
- ❑ Inverted Indexes
- ❑ Query Processing
- ❑ Index Construction
- ❑ Index Compression

Inverted Indexes

Index [ANSI/NISO 1997]

An index is a systematic guide designed to indicate topics or features of documents in order to facilitate retrieval of documents or parts of documents.

The function of an index is to provide users with an effective and systematic means for locating documentary units (complete documents or parts of documents) that are relevant to information needs or requests.



thesauri
- definition : 12

titles of documents : 6.2.9, 6.2.9.4
- capitalization : 6.2.3
- initial articles in alphanumeric arrangement : 9.4

topical headings
- ***see also:*** entries, headings, terms
- initial articles in alphanumeric arrangement : 9.4

topics
- definition : 12
- major versus minor topics : 7.3

transcription
- definition : 12

transliteration : 6.2.10
- definition : 12

truncation
- definition : 12
- in searching : 7.5.3

turnover lines : 8.2.5.1, 8.2.5.3
- definition : 12

vectors
- definition : 12
- in searching : 7.5.2

vertical spacing
- in indexes : 8.2.4

video recordings
- locators : 7.4.2b

visual indexes
- *see:* displayed indexes

vocabulary : 6
- ***see also:*** descriptors; terminology of indexing; terms
- control, tracking, management : 3h, 5.13; as essential process : *preface*; definition : 12
- display in displayed indexes : 6.8.1; non-displayed electronic search indexes : 6.8.2
- entry. definition : 12
- lead-in : 5.13
- sources : 3d-e, 6.1

Inverted Indexes

Index [ANSI/NISO 1997]

Requirements:

1. identify documentary units that treat particular topics or possess particular features.

2. indicate all important topics or features of documentary units in accordance with the level of exhaustivity appropriate for the index.

3. discriminate between major and minor treatments of particular topics or manifestations of particular features.

4. provide access to topics or features using the terminology of prospective users.

5. provide access to topics or features using the terminology of verbal texts being indexed whenever possible.

Inverted Indexes

Index [ANSI/NISO 1997]

Requirements: (continued)

6. use terminology that is as specific as documentary units warrant and the indexing language permits.

7. provide access through synonymous and equivalent terms.

8. guide users to terms representing related concepts (narrower terms, other related terms, and if possible, broader terms).

9. provide for the combination of terms to facilitate the identification of particular types or aspects of topics or features and to eliminate unwanted types or aspects.

10. provide a means for searching for particular topics or features by means of a systematic arrangement of entries in displayed indexes or, for non-displayed indexes, by means of a clearly documented and displayed method for entering, combining, and modifying terms to create search statements and for reviewing retrieved items.

Remarks:

- ❑ Several standards worldwide govern the (manual) construction and formatting of indexes. [Gibbs 2015]
- ❑ ANSI = American National Standards Institute
- ❑ NISO = National Information Standards Organization

Inverted Indexes

Data Structure

Term-document matrix:

$$\begin{array}{lcccccc} & d_1 & d_2 & d_3 & d_4 & d_5 & \dots \\ \hline t_1 & \boxed{w_{1_1}} & \boxed{w_{1_2}} & \boxed{w_{1_3}} & \boxed{w_{1_4}} & \boxed{w_{1_5}} & \\ t_2 & \boxed{w_{2_1}} & \boxed{w_{2_2}} & \boxed{w_{2_3}} & \boxed{w_{2_4}} & \boxed{w_{2_5}} & \\ t_3 & \boxed{w_{3_1}} & \boxed{w_{3_2}} & \boxed{w_{3_3}} & \boxed{w_{3_4}} & \boxed{w_{3_5}} & \\ t_4 & \boxed{w_{4_1}} & \boxed{w_{4_2}} & \boxed{w_{4_3}} & \boxed{w_{4_4}} & \boxed{w_{4_5}} & \\ t_5 & \boxed{w_{5_1}} & \boxed{w_{5_2}} & \boxed{w_{5_3}} & \boxed{w_{5_4}} & \boxed{w_{5_5}} & \\ \vdots & & & & & & \end{array}$$

Observations:

- Most information 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 Indexes

Data Structure

Term-document matrix:

	d_1	d_2	d_3	d_4	d_5	$\ldots$
t_1	w_{1_1}	w_{1_2}	w_{1_3}	w_{1_4}	w_{1_5}	
t_2	w_{2_1}			w_{2_4}	w_{2_5}	
t_3		w_{3_2}	w_{3_3}			
t_4			w_{4_3}	w_{4_4}		
t_5		w_{5_2}				
$\vdots$						

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 Indexes

Data Structure

Term-document matrix as inverted index data structure:

T	**Postings**				
t_1	$\boxed{d_1, w_{1_1}}$	$\boxed{d_2, w_{1_2}}$	$\boxed{d_3, w_{1_3}}$	$\boxed{d_4, w_{1_4}}$	$\boxed{d_5, w_{1_5}}$
t_2	$\boxed{d_1, w_{2_1}}$	$\boxed{d_4, w_{2_4}}$	$\boxed{d_5, w_{2_5}}$		
t_3	$\boxed{d_2, w_{3_2}}$	$\boxed{d_3, w_{3_3}}$			
t_4	$\boxed{d_3, w_{4_3}}$	$\boxed{d_4, w_{4_4}}$			
t_5	$\boxed{d_2, w_{5_2}}$				
$\vdots$					

Components:

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

Inverted Indexes

Data Structure

Term-document matrix as inverted index data structure:

T	**Postings**				
t_1	$\boxed{d_1, w_{1_1}}$	$\boxed{d_2, w_{1_2}}$	$\boxed{d_3, w_{1_3}}$	$\boxed{d_4, w_{1_4}}$	$\boxed{d_5, w_{1_5}}$
t_2	$\boxed{d_1, w_{2_1}}$	$\boxed{d_4, w_{2_4}}$	$\boxed{d_5, w_{2_5}}$		
t_3	$\boxed{d_2, w_{3_2}}$	$\boxed{d_3, w_{3_3}}$			
t_4	$\boxed{d_3, w_{4_3}}$	$\boxed{d_4, w_{4_4}}$			
t_5	$\boxed{d_2, w_{5_2}}$				
$\vdots$					

Design choices:

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

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.

Inverted Index

Postings

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

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

`<document>`:

- Reference to the document d in which a given term t occurs.

`<weights>`:

- Term weight w for term t in document d.
- Term weights can be stored, say, *tf*(t, d) or term weights of the retrieval model. Saves query processing time, but predetermines a retrieval model.

`<position>`:

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

Inverted Index

Postings

Two special-purpose entries are distinguished:

```
...  <list length>
```

```
...  <skip pointer>
```

`<list length>`:

- First entry of the postings list of a term t.
- Stores the length of the postings list, excluding special-purpose entries.
- What does the length of a postings list indicate?

`<skip pointer>`:

- Used to implement a skip list in a term's postings list.
- Allows for random access to postings in $O(\log n)$.
- Second entry of a postings 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

Postings

Two special-purpose entries are distinguished:

```
...   <list length>
```

```
...   <skip pointer>
```

`<list length>`:

- First entry of the postings list of a term t.
- Stores the length of the postings list, excluding special-purpose entries.
- Corresponds to the document frequency *df*(t, D) of term t in collection D.

`<skip pointer>`:

- Used to implement a skip list in a term's postings list.
- Allows for random access to postings in $O(\log n)$.
- Second entry of a postings 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

Postings Lists

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

T	**Postings**										
$\vdots$											
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	$1, 1$ 🟧🟩	$2, 3$	$3, 5$	$5, 2$ 🟩	$8, 17$	$41, 6$	$51, 5$ 🟩	$60, 5$	$71, 3$	$77, 2$ 🟩	$\ldots$
$\vdots$											

Ordering:

- by document identifier. Problem: good documents may end up last.
- by term weight. Early termination, but renders skip lists useless.
- by document quality. 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.
- Postings lists can be effectively compressed with tailored techniques.

Remarks:

- ❑ There is a tradeoff between the amount of information stored in a posting, and the time it takes to process a postings 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.