Chapter 3 - Text
Management and Retrieval

Outline

- Text as a media object
- Text/Information retrieval models
  - boolean model
  - fuzzy set model
  - vector model
- Relevance feedback
- Retrieval evaluation
  - relevance, precision, recall
- Query languages
  - keyword-based
  - pattern matching
- Document preprocessing
- Indexing and Searching

Literature:
Text

- Raw data
  - sequence of characters (letters, digits, special characters) of variable length, printable
- Registration data
  - character set (ISO Latin-1, Unicode, Kanji)
  - # of bits/character (8, 16, ...)
  - character encoding (ASCII, EBCDIC, UTF-8, UTF-16)
  - length of text, or end-of-text character (0x0)
  - length of line, or end-of-line character (<LF>, <RET>)
- Descriptive data
  - structural information, e.g.
    - chapter (nr, heading)
    - section (nr, chapter-nr, position, length)
  - content information (keywords, abstract, ...)
- Operations
  - read: determine length, substring, position of a subsequence
  - write: concatenate, replace

Information Retrieval Process

query media objects

preprocessing

query representation

preprocessing

object representation

comparison (similarity)

result objects

query refinement and relevance feedback
Traditional Information Retrieval Models

- Based on index terms as a means for internal document representation
  - restricted form: keyword (usually a noun) with a meaning of its own
  - general form: any word appearing in a document in a collection of documents
- Fundamental assumption: semantics of a document (and user information need) can be expressed through a set of index terms
  - but not completely – a lot of semantics is "lost" in the process
  - matching of documents and queries is imprecise
- Central problem: predicting which documents are relevant, and which not
- An information retrieval model
  - refines the internal document/query representation
  - defines comparison function (ranking function) for two documents, or a document and a query to determine relevance of documents
- Classic models, developed originally for text retrieval
  - boolean, vector, probabilistic
  - extensions, variations

Basic Concepts

- Document collection
  - contains a set of text documents
  - scope of a retrieval operation, which returns relevant documents
- Index term
  - helps describe the main theme of a document
- Term weight
  - numerical value
  - describes the importance of an index term in describing a specific document
  - (or describes the importance of a query term for the user information need)
- Index term vocabulary
  - set of all index terms used to describe the documents in a collection/system
Boolean Model

- Simple retrieval model based on set theory and boolean algebra
  - received a lot of attention in the early IR years
  - incorporated in a lot of early commercial IR systems
- Internal document representation: a set of index terms
  - term weights are binary
- Comparison function only considers presence/absence of query keyword in the document
- Query
  - index term
  - combination of index terms using boolean operators
    AND (set intersection)
    OR (set union)
    NOT (set complement)

Boolean Model – Example

- Vocabulary: {digital library content management multimedia database}
- Documents
  - d1: {digital library multimedia}
  - d2: {digital library content management}
  - d3: {content management multimedia database}
- Queries and results

```plaintext
library      content
<table>
<thead>
<tr>
<th>library</th>
<th>content</th>
</tr>
</thead>
<tbody>
<tr>
<td>d1</td>
<td>d2 d3</td>
</tr>
</tbody>
</table>

library AND content
```

```plaintext
library | AND NOT content
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>d1</td>
<td>d2 d3</td>
</tr>
</tbody>
</table>
```

```plaintext
library | OR content
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>d1</td>
<td>d2 d3</td>
</tr>
</tbody>
</table>
```
Additional Operators

- BUT
  - motivation: 'NOT content' retrieves all documents that do not contain "content"
  - may be the whole document collection
  - negation is often restricted to the BUT operation (equivalent to AND NOT)
- OF – construct
  - search for documents that contain m out of n (m < n) terms
  - example: 2 OF (content, library, multimedia)
  - equivalent to a more complex boolean expression
    (content AND library) OR (content AND multimedia) OR (library AND multimedia)

Query Processing Steps

- Query normalization in disjunctive normal form (DNF) or conjunctive normal form (CNF)
  - query: content AND ((digital AND library) OR multimedia)
  - DNF: (content AND digital AND library) OR (content AND multimedia)
  - CNF: content AND (digital OR multimedia) AND (library OR multimedia)
- Query evaluation
  - every term determines a set of documents described by the term
  - complex query: determine result using the set operations that correspond to the operators
  - DNF will help reduce size of intermediate results (first intersection, then union)
Problems With Boolean Model

- Exact model based on binary weights
  - strong data retrieval “flavor” (instead of IR)
  - search is too precise, no notion of similarity considered
- Query result
  - all result documents are considered equally relevant
    - no ranking
    - complete presentation/inspection of result required
  - depending on the query, result is often found to be
    - too large
    - too small (empty)
- Boolean query operators
  - are considered unnatural, too difficult to use by most end-users
  - confusion with the colloquial semantics of “and”, “or”, “not”

Possible Improvements

- Attempt to address the "exact model" problem
  - introduce different levels of relevance
    - turn "AND" into "OR" and
    - present results based on the number of terms matched by the document
  - example: library AND content -> library OR content
    - first return documents containing both terms (d2)
    - then those containing any of the terms (d1, d3)
- Attempt to address the query result (size) problem using faceted query
  - two-level search
    - formulate query and refine it based on named (previous) queries and result set size
    - retrieve final query result
  - example:
    - content returns Q1: 5276
    - Q1 AND management returns Q2: 17
    - display Q2 returns 17 result documents
Fuzzy Set Model

- Alternative set-theoretic model that extends boolean retrieval with "fuzziness"
  - Generalizes the boolean operators
  - Fuzziness introduces a gradual association of documents with terms
- Based on fuzzy set theory

**Definition:**
A fuzzy subset $A$ of a universe of discourse $U$ is characterized by a membership function $\mu_A : U \rightarrow [0, 1]$ which associates with each element $u$ of $U$ a number $\mu_A(u)$ in the interval $[0, 1]$.

Fuzzy Sets in Information Retrieval
- universe is the document collection
- term $t$ defines a fuzzy subset
- membership function $\mu_t(d)$
  - 0 for no relevance
  - 1 for maximal relevance
  - value between 0 and 1 for gradual relevance

Fuzzy Set Model – Operations

- Standard operations on fuzzy sets
  - every document is a member of each fuzzy set
    - the usual set operator semantics is not applicable
  - standard operators determine a new membership value
    - AND (conjunction): $\mu_{A \wedge B}(u) = \min(\mu_A(u), \mu_B(u))$
    - OR (disjunction): $\mu_{A \vee B}(u) = \max(\mu_A(u), \mu_B(u))$
    - NOT (negation): $\mu_{\neg A}(u) = 1 - \mu_A(u)$
- Example

<table>
<thead>
<tr>
<th>$\mu$</th>
<th>$d1$</th>
<th>$d2$</th>
<th>$d3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>content</td>
<td>0.7</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>database</td>
<td>0.5</td>
<td>0.75</td>
<td>1</td>
</tr>
<tr>
<td>content $\cap$ database</td>
<td>0.5</td>
<td>0.75</td>
<td>1</td>
</tr>
<tr>
<td>content $\cup$ database</td>
<td>0.7</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$\neg$ content</td>
<td>0.3</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Query Processing Steps

- Queries and query processing
  - query is transformed into DNF
  - every query term determines a fuzzy set
  - application of fuzzy set operators on fuzzy sets
  - result: documents sorted by descending membership values

- Limiting the size of the retrieval result
  - result consists of all documents (complete universe)
  - reduce result size using
    - threshold for membership value
    - fixed result set size

Membership Function

- Numerous approaches for determining membership values
- Example: term-term correlation matrix (Ogawa, Morita & Kobayashi)
  - defines correlation $c_{i,j}$ of any pair of terms $t_i$ and $t_j$ in the vocabulary
    - rows correspond to term $t_i$, columns to term $t_j$
    - $n_i$: #docs containing $t_i$, $n_j$: #docs containing $t_j$
    - $n_{i,j}$: #docs containing both $t_i$ and $t_j$
  - membership degree for term $t_i$ and document $d_j$
    - 1, if the index term is contained in the document
    - 0, if a term closely related to the index term is contained

\[
\mu_i(d_j) = 1 - \prod_{t_j \in d_j} (1 - c_{i,t_j})
\]

- Example:
  - d1: (digital library multimedia)
  - d2: (digital library content management)
  - d3: (content management multimedia database)

<table>
<thead>
<tr>
<th></th>
<th>digital</th>
<th>library</th>
<th>content</th>
<th>management</th>
<th>multimedia</th>
<th>database</th>
</tr>
</thead>
<tbody>
<tr>
<td>digital</td>
<td>1</td>
<td>1</td>
<td>0,33</td>
<td>0,33</td>
<td>0,33</td>
<td>0</td>
</tr>
<tr>
<td>library</td>
<td>1</td>
<td>1</td>
<td>0,33</td>
<td>0,33</td>
<td>0,33</td>
<td>0</td>
</tr>
<tr>
<td>content</td>
<td>0,33</td>
<td>0,33</td>
<td>1</td>
<td>1</td>
<td>0,33</td>
<td>0,5</td>
</tr>
<tr>
<td>management</td>
<td>0,33</td>
<td>0,33</td>
<td>1</td>
<td>1</td>
<td>0,33</td>
<td>0,5</td>
</tr>
<tr>
<td>multimedia</td>
<td>0,33</td>
<td>0,33</td>
<td>0,33</td>
<td>0,33</td>
<td>1</td>
<td>0,5</td>
</tr>
<tr>
<td>database</td>
<td>0</td>
<td>0</td>
<td>0,5</td>
<td>0,5</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Vector Model

- Addresses limitations of the boolean model (i.e., binary weights) by assigning non-binary weights to index terms in queries and documents
- Assumption: fixed set of terms used for queries and as document descriptors
- Approach
  - fixed vocabulary consisting of N terms
  - document $D_i = (T_{i1}, T_{i2}, ..., T_{ik}, ..., T_{iN})$, $T_{ik}$ weight of term $k$ in document $i$
  - query $Q_j = (Q_{j1}, Q_{j2}, ..., Q_{jk}, ..., Q_{jN})$, $Q_{jk}$ weight of term $k$ in query $j$
  - both document and query are interpreted as N-dimensional vectors in the vector space defined by the set of terms
  - similarity of document $D_i$ and query $Q_j$ is defined as the correlation of the two vectors
  - cosine similarity quantifies the correlation using the cosine of the angle between vectors

\[
\text{sim}_{\cos}(D_i, Q_j) = \frac{D_i \cdot Q_j}{|D_i| \cdot |Q_j|} = \frac{\sum_{k=1}^{N} T_{ik} \cdot Q_{jk}}{\sqrt{\sum_{k=1}^{N} T_{ik}^2 \cdot \sum_{k=1}^{N} Q_{jk}^2}}
\]

Calculating Index Term Weights

- TFIDF (term frequency/inverse document frequency) method
  - Determines a weight $w_{D,t}$ for each term $t$ of a document $D$ based on
    - its frequency in the document (term frequency, $tf_{D,t}$) and
    - the inverse of its frequency in all documents (inverse document frequency, $idf(t)$)
  - Basic weight formula: $w_{D,t} = tf_{D,t} \cdot idf(t)$
  - Idea:
    - terms occurring more frequently in the document $D$ have a higher weight because they help characterize the document $D$
    - terms occurring in almost every document receive a lower weight because they don’t help to distinguish document $D$ from the other documents
  - Many different approaches to calculate $tf_{D,t}$ and $idf(t)$
    - e.g., with $n_{D,x}$ being the number of occurrences of term $x$ in document $D$, $T$ being the set of all terms in $D$, $N$ being the total number of documents, and $N_t$ being the number of documents that contain term $t$ (at least once):

\[
tf_{D,t} = \frac{n_{D,t}}{\max_x \left( n_{D,x} \right)} \quad \text{and} \quad idf_t = \log \frac{N}{N_t}
\]
Vector Model - Summary

- Very popular method
- Requires fixed set of numerical properties (e.g., term weights) per document
- Problems:
  - terms are assumed to be independent from each other (i.e., not correlated), which is unrealistic
  - doesn't work well for large term vocabularies, large documents
  - query is a vector, there are no boolean operators
    - possible combination with fuzzy set model to introduce boolean operators

Probabilistic Retrieval Models

- Parameters:
  - \( P(\text{rel}) \) – probability that a document is relevant
  - \( P(\text{nonrel}) \) – probability that a document is not relevant
  - \( a_1 \) – cost associated with returning a non-relevant document
  - \( a_2 \) – cost associated with not returning a relevant document
- Document retrieval as a cost minimization problem
  - document is included in the result set, if
    \[ a_2 P(\text{rel}) \geq a_1 P(\text{nonrel}) \]
- Main task:
  - estimation of \( P(\text{rel}) \) and \( P(\text{nonrel}) \) – see literature
Relevance Feedback

- End-user marks returned documents as relevant or irrelevant
- Query modification
  - terms present in documents marked as relevant are added to the query, or their weights are increased
  - terms present in documents marked as irrelevant are removed from the query, or their weights are decreased
  - improves retrieval quality, but only for the current user
- Example (for vector model)

\[ Q^{(n+1)} = \alpha Q^{(n)} + \beta \left( \sum_{d \in Re} d - \frac{\gamma}{|NonRe|} \sum_{d \in NonRe} d \right) \]

- Document modification
  - query terms not present in the document are added as descriptors with an initial weight
  - query terms present in relevant documents receive an increase for the document descriptors
  - document descriptor weight is reduced for descriptors which are not query terms (because document was found without the term)
  - positive effects if similar queries follow, effects questionable for other queries

Retrieval Evaluation

- Response time
- Relevance
  - Recall: fraction of relevant documents which has been retrieved
    - recall = #relevant results / #all relevant documents
  - Precision: fraction of retrieved documents which is relevant
    - precision = #relevant results / #result documents
- Contradictory goals
  - maximize precision
  - maximize recall
- Retrieval engine needs to find compromise
- Problem: determining all relevant documents
  - requires inspection of document collection
  - benchmark on small test database
Retrieval Evaluation (2)

- Precision-Recall-Graph:

  \[
  \begin{array}{c}
  \text{Precision-Recall-Graph:} \\
  \end{array}
  \]

- determines precision for increasing recall numbers
- needs to be determined for a large, representative set of queries
  - aggregated into average precision-recall figures

Retrieval Evaluation (3)

- Example:

<table>
<thead>
<tr>
<th>#results</th>
<th>R/I</th>
<th>recall</th>
<th>precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>R</td>
<td>1/N</td>
<td>1/1</td>
</tr>
<tr>
<td>2</td>
<td>R</td>
<td>2/N</td>
<td>2/2</td>
</tr>
<tr>
<td>3</td>
<td>I</td>
<td>2/N</td>
<td>2/3</td>
</tr>
<tr>
<td>4</td>
<td>I</td>
<td>2/N</td>
<td>2/4</td>
</tr>
<tr>
<td>5</td>
<td>R</td>
<td>3/N</td>
<td>3/5</td>
</tr>
<tr>
<td>6</td>
<td>R</td>
<td>4/N</td>
<td>4/6</td>
</tr>
<tr>
<td>7</td>
<td>I</td>
<td>4/N</td>
<td>4/7</td>
</tr>
<tr>
<td>8</td>
<td>I</td>
<td>4/N</td>
<td>4/8</td>
</tr>
<tr>
<td>9</td>
<td>R</td>
<td>5/N</td>
<td>5/9</td>
</tr>
<tr>
<td>10</td>
<td>I</td>
<td>5/N</td>
<td>5/10</td>
</tr>
</tbody>
</table>
Query Languages

- Different types of queries possible
  - largely dependent on the underlying information retrieval model
- Keyword-based querying
  - single-word
  - context-based
  - boolean
  - natural language
- Pattern matching

Keyword-Based Querying

- Single-word queries
  - ask for documents containing one (or more) words
  - example
    "content", "management"
- Context queries: searching for words "near" other words
  - phrase search: find a sequence of words
    - example
      "content management"
    - does not match "content and multimedia database management"
    - considers a 'normalized' phrase representation
      - number of separator/whitespace characters in the text is ignored
    - usually does not consider "uninteresting" words (stopwords)
      - previous query could match "content and management"
  - proximity search: find a set of words with a maximum distance from each other
    - distance: characters, words, sentences, paragraphs
    - examples
      "content" IN SAME SENTENCE AS "management"
      "content" NEAR "management" WITHIN 3 WORDS
Keyword-Based Querying (2)

- Boolean queries
  - involve boolean operators (AND, OR, NOT) – see previous discussion in this chapter

- Natural language queries
  - query is an enumeration of words
  - ranking may consider
    - number of words matched by the document
    - proximity of words appearing in the document vs. those in the query

- General option: query term expansion
  - to cover linguistic variations (see discussion of stemming later in this chapter), or
  - by introducing synonyms, broader terms, narrower terms (using a thesaurus)

Pattern Matching

- Retrieval of pieces of text that match a specified pattern
  - pattern: set of syntactic features that have to occur in a text segment
  - search for all documents containing a text segment that matches the pattern

- Types of patterns
  - word – string of characters
  - prefix/suffix – beginning/end of a text word
  - substring – string appearing within a text word
  - range – pair of string defining a lexical range
    - example: ["content" "continent"] will also match "context"
  - allowing errors – word with error threshold matches "similar" words
    - Edit-distance functions, e.g. Levenshtein distance:
      - Count the number of edit operations (insert, modify, delete) to turn string a into string b
      - Example:
        - kitten
        - sitting
        - 2 replacements, 1 insertion LevenshteinDist("kitten","sitting") = 3
      - Weighting of operations possible (e.g. replace more expensive than delete)
      - Normalization to interval [0,1] by dividing result through max(length(String A), length(string B))
Pattern Matching (2)

- Types of patterns (cont.)
  - wildcard characters
    - ., +, * match a single, at least one, any number of arbitrary characters in a word
  - regular expressions: general pattern built by simple strings
  - extended patterns
    - subsets of regular expressions
    - expressed with a simpler syntax
    - examples
      - wildcard characters (see above)
      - case-insensitive matching

<table>
<thead>
<tr>
<th>operator</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>.</td>
<td>any character</td>
</tr>
<tr>
<td>A*</td>
<td>any number of As</td>
</tr>
<tr>
<td>A+</td>
<td>one or more As</td>
</tr>
<tr>
<td>A?</td>
<td>single A or nothing</td>
</tr>
<tr>
<td>[a-d]</td>
<td>any of a, b, c, or d</td>
</tr>
<tr>
<td>(a)</td>
<td>matches expression a</td>
</tr>
<tr>
<td>a</td>
<td>b</td>
</tr>
</tbody>
</table>

Document Preprocessing

- Goal: produce internal logical document representation
  - set of descriptors, index terms for each document
- Manual preprocessing
  - classification (hierarchical, faceted)
  - indexing
    - words contained in the document
    - words not contained in the document
      - using thesaurus
      - descriptor attributes (aspects, roles)
      - example: name:carpenter vs. profession:carpenter
      - descriptor weights
- Automatic preprocessing
  - assign index terms based on the text content of the document
  - goal: improve retrieval quality
  - involves various types of operations on the text
    - successive transformations of text into index terms
Preprocessing Operations

- Lexical analysis
  - treat digits, hyphens, punctuation marks, case of letters to produce a sequence of words
- Stopword elimination
  - filter out words with very low discrimination values (i.e., unsuitable for retrieval)
- Stemming
  - reduce words to their word stems (e.g., remove prefixes, suffixes, inflections) to allow for syntactic variations of query terms
- Index term selection
  - select words that carry more "semantics"
  - usually based on the syntactic nature (e.g., only nouns)

and also
- Construction of term categorization structures (thesaurus), extracting document text structure

Phases of Text Preprocessing
Lexical Analysis

- Converts a stream of characters into a sequence of words
  - problem: identification of words in the text
- Subtasks
  - recognition of whitespace characters as word separators
  - treatment of numbers
    - numbers alone are usually too vague without a surrounding context, can be disregarded
      - example: "2008" could be a year, phone number, PIN, ...
    - but maybe "510B.C." or "0631 205 3279" are useful index terms
      - advanced lexical analysis to recognize and normalize specific kinds of numbers
  - hyphenated words
    - break up into words? - "state-of-the-art" vs "state of the art"
    - don't break up - "B-52", "cutting-edge", ...
  - removing punctuation marks
    - o.k. for "text, image, audio ...", "510B.C."
    - but not for "x.id" (vs. "xid") in program code
  - converting to upper/lower case for normalization
    - what about "TED" vs. "Ted", "OR" vs. "or", ...

Stopword Elimination

- Stopword: word that appears too frequently in the document collection (e.g., >80%) to be of any discriminative use
  - don't contribute to the retrieval task, can be eliminated
  - articles, prepositions, conjunctions are likely candidates
  - might also include adjectives, adverbs, verbs
- Additional benefit: reduces the size of the indexing structure
- Possible problems
  - reduction of precision
    - "War and Peace" vs. "war or peace", "war, peace"
  - reduction of recall
    - "To be or not to be" cannot be found
Stemming & Index Term Selection

- Stemming
  - stem: portion of a word left after removal of affixes (pre-/suffixes)
  - example: “connect” for “connected”, “connecting”, “connection”, “connections”
  - reduces variants of the same root word to a common concept
  - different strategies, such as table lookup or affix removal algorithms
    - Porter algorithm: rules for stripping off suffixes

- Index term selection
  - full text representation adopts all words
  - more selective approaches
    - eliminate verbs, adjectives, adverbs, … (see stopword elimination)
    - identify noun groups as index terms
      - example: “database management system”, “computer science”, “digital library”

Text Mining

- Discover knowledge (high-level information) in unstructured text for metadata creation
  - patterns
  - relationships

- Uses techniques from
  - data mining
  - natural language processing
  - machine learning
Text Mining Operations

- **Feature Extraction**
  - identifies facts and relations in the text
  - distinguishes person, place, organization, etc. for noun phrases
  - uses dictionaries, linguistic patterns (part of speech tagging)

- **Categorization**
  - classifies documents into predefined categories
    - thesaurus-based approach
      - categories are determined based on frequencies of domain-specific terms
    - machine-learning approach
      - categorizer is trained with (pre-categorized) sample documents
      - statistically analyzes linguistic patterns (word frequencies, lexical affinities)
      - builds statistical signatures for the categories
      - uses the signatures to classify new documents

Text Mining Operations (2)

- **Clustering**
  - groups together related documents into clusters
    - based on a similarity measure
      - may utilize lexical affinities, extracted features
    - done without predefined categories
  - variations
    - hierarchical
    - binary relational (flat)
    - fuzzy

- **Summarization**
  - reduces text while still keeping its key meaning
  - based on input parameters
    - number of sentences, percentage of original text to extract
  - result contains the most important sentences
Linear/Sequential Searching

- Text documents are stored in one or multiple files
  - search by scanning the text sequentially, "before" preprocessing
    - also called "online search"
- Type of search: pattern matching
- Performance time proportional to length of document (collection)
  - numerous algorithms
- Appropriate only for small document collections
  - but may be the only option if
    - text collection changes too frequently
      - indexing is done periodically, index doesn't reflect current state
    - indexing is too expensive

Signature Files

- Word-oriented index structure based on hashing
  - hash function (signature) maps words to bitmaps of length \( B \)
  - text is divided into blocks with \( b \) words each
  - a bitmap of length \( B \) is assigned to each block (plus a pointer to each block)
    - block bitmap is computed by bitwise OR-ing of the word bitmaps for all words in the block
  - idea: a word can only be present in the block, if the bits that are set in the word bitmap are also set in the block bitmap
- False positives
  - query word bitmap and block bitmap match, but the word doesn't appear in the block
  - probability for false positives depends on proportion \( b / B \)
  - tradeoff: overhead vs. false positives probability
Signature Files - Search

- Single keyword queries
  - determine signature value of query term, find matching block bitmaps
    - with $q = \text{bitmap of query term}$, $B_i = \text{bitmap of block } i$, find all blocks with $q \land B_i = q$
  - additional sequential search on block is required for determining word match
- Phrase queries
  - determine bitwise OR of bitmaps for all words in the phrase
  - allows to find blocks (potentially) containing all the words
  - need to have blocks overlap to account for phrase words on block boundaries

Signature Files - Example

<table>
<thead>
<tr>
<th>Block 1</th>
<th>Block 2</th>
<th>Block 3</th>
<th>Block 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>This is a text.</td>
<td>A text has many</td>
<td>word. Words are</td>
<td>made from letters</td>
</tr>
<tr>
<td>000101</td>
<td>110101</td>
<td>100100</td>
<td>101101</td>
</tr>
</tbody>
</table>

text signature

$\text{h(text)} = 000101$
$\text{h(many)} = 110000$
$\text{h(words)} = 100100$
$\text{h(made)} = 001100$
$\text{h(letters)} = 100001$

false positive!
Inverted Files

- Word-oriented index
  - lists for each index term the identifiers of documents to which the keyword was assigned
  - includes additional information about the word occurrence (position in the document) and term weights
- Entry (row in the inverted file)
  - (term: (doc-id-1 pos-1 ... pos-n weight) ...)
  - pos: paragraph-no, sentence-no, word-no, character-no
- Search
  - search for individual terms in the index
  - phrase and proximity queries need to be split up
  - retrieve the occurrences
  - manipulate the occurrences
    - resolve phrases, proximity, boolean operators (list union/intersection/difference)
- Search steps can be supported by appropriate access structures (B-trees, ...)

Inverted files/indexes are the best foundation for text retrieval in practice

---

Inverted Files - Example

<table>
<thead>
<tr>
<th>text</th>
<th>occurrences</th>
</tr>
</thead>
<tbody>
<tr>
<td>letters</td>
<td>60 ...</td>
</tr>
<tr>
<td>made</td>
<td>50 ...</td>
</tr>
<tr>
<td>many</td>
<td>28 ...</td>
</tr>
<tr>
<td>text</td>
<td>11, 19 ...</td>
</tr>
<tr>
<td>words</td>
<td>33, 40 ...</td>
</tr>
</tbody>
</table>

This is a text. A text has many words. Words are made from letters.
Summary

- Text as a media object
- Text retrieval - main problems
  - imprecise search
  - similarity, relevance of text documents
- Text/Information retrieval models defines internal document model, similarity rank
  - boolean model, fuzzy set model, vector model
- Relevance feedback uses user input to improve retrieval quality
  - adjusts query and/or document representation
- Retrieval evaluation based on notions of relevance, precision, recall
- Query languages
  - keyword-based (keyword, phrase, boolean operators, natural language queries)
  - pattern matching (simple patterns, error tolerance, regular expressions)
- Document preprocessing produces logical internal document representation
  - lexical analysis, stopword elimination, stemming, index term selection
- Text Mining operations extract higher-level information
  - feature extraction, categorization, clustering, summarization
- Indexing and Searching for text
  - sequential search, signature files, inverted files/indexes