2 Retrieval of Text Information

Motivation

- Information Retrieval is a field of activity for many years
- IR was long seen as an area of narrow interest. Advent of the Web changed this perception
  - universal repository of knowledge
  - free (low cost) universal access
  - no central editorial board
- Objective: representation, storage, organization of, and access to information items
- Emphasis on the retrieval of information (not data)
- Focus is on the user information need
- The information need is expressed as a query
Information Retrieval systems do not search through the documents but through the representation (also called index, meta-data or description).

source: (Ferber 2004)

Example

D1

Heavy accident
Because of a heavy car accident 4 people died yesterday morning in Vienna.

D2

More vehicles
In this quarter more cars became registered in Vienna.

D3

Truck causes accident
In Vienna a trucker drove into a crowd of people. Four people were injured.

Information need: documents containing information about car accidents in Vienna where heavy vehicles were involved

Query: accident heavy vehicles vienna

Expected result: document D3

but:

+ not all terms of the query occur in the document
+ the occurring terms „accident“ and „heavy“ also occur in D1


### Retrieval System

- Each document represented by a set of representative keywords or index terms
- An index term is a document word useful for remembering the document main themes
- The index is stored in an efficient system or data structured
- Queries are answered using the index
- With the ID, the document can be retrieved

### Indexing

- Manual indexing – key words
  - User specifies key words, he/she assumes useful
  - Usually, key words are nouns because nouns have meaning by themselves
  - There are two possibilities
    1. User can assign any terms
    2. User can select from a predefined set of terms (controlled vocabulary)

- Automatic indexing – full text search
  - Search engines assume that all words are index terms (full text representation)
  - System generates index terms from the words occurring in the text
**Automatic Indexing:**

1. **Decompose a Document into Terms**

- **D1:** heavy accident because of a heavy car accident
- **D2:** more vehicles in this quarter more cars became registered in vienna
- **D3:** Truck causes accident in vienna a trucker drove into a crowd of people four people were injured

- Rules determine how texts are decomposed into terms by defining separators like:
  - punctuation marks, blanks or hyphens
- Additional preprocessing, e.g.:
  - exclude specific strings (stop words, numbers)
  - generate normal form
    - stemming
    - substitute characters (e.g. upper case – lower case, Umlaut)

2. **Index represented as an inverted list**

- For each term:
  - list of documents in which the term occurs
  - additional information can be stored with each document like:
    - frequency of occurrence
    - positions of occurrence

*In inverted list is similar to an index in a book*
Index as Inverted List with Frequency

In this example the inverted list contains the document identifier and the frequency of the term in the document.

<table>
<thead>
<tr>
<th>term</th>
<th>(document; frequency)</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>(D1,1) (D3,2)</td>
</tr>
<tr>
<td>accident</td>
<td>(D1,2) (D3,1)</td>
</tr>
<tr>
<td>became</td>
<td>(D2,1)</td>
</tr>
<tr>
<td>because</td>
<td>(D1,1)</td>
</tr>
<tr>
<td>car</td>
<td>(D1,1)</td>
</tr>
<tr>
<td>cars</td>
<td>(D2,1)</td>
</tr>
<tr>
<td>died</td>
<td>(D1,1)</td>
</tr>
<tr>
<td>heavy</td>
<td>(D1,2)</td>
</tr>
<tr>
<td>in</td>
<td>(D1,1) (D2,1) (D3,1)</td>
</tr>
<tr>
<td>more</td>
<td>(D2,1)</td>
</tr>
<tr>
<td>of</td>
<td>(D1,1)</td>
</tr>
<tr>
<td>people</td>
<td>(D1,1) (D3,2)</td>
</tr>
<tr>
<td>quarter</td>
<td>(D2,1)</td>
</tr>
<tr>
<td>registered</td>
<td>(D2,1)</td>
</tr>
<tr>
<td>truck</td>
<td>(D3,1)</td>
</tr>
<tr>
<td>vehicles</td>
<td>(D2,1)</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

Problems of Information Retrieval

- **Word form**
  - A word can occur in different forms, e.g. singular or plural.
  - Example: A query for „car“ should find also documents containing the word „cars“

- **Meaning**
  - A singular term can have different meanings; on the other hand the same meaning can be expressed using different terms.
  - Example: when searching for „car“ also documents containing „vehicle“ should be found.

- **Wording, phrases**
  - The same issue can be expressed in various ways
  - Example: searching for „motorcar“ should also find documents containing „motorized car“
**Word Forms**

- **Flexion:** Conjugation and declension of a word
  - car - cars
  - run - ran - running

- **Derivations:** words having the same stem
  - form - format - formation

- **Compositions:** statements
  - information management - management of information

In German, compositions are written as single words, sometimes with hyphen
  - Informationsmanagement
  - Informations-Management

---

**Word Meaning and Phrases**

*Dealing with words having the same or similar meaning*

- **Synonyms**
  - record - file - dossier
  - seldom - not often

- **Variants in spelling (e.g. BE vs. AE)**
  - organisation - organization
  - night - nite

- **Abbreviations**
  - UN - United Nations

- **Polyseme:** words with multiple meanings
  - Bank
2.1 Dealing with Word Forms and Phrases

In principle, we distinguish two ways to deal with word forms and phrases:

- **Indexing without preprocessing**
  - All occurring word forms are included in the index
  - Different word forms are unified at search time
    - string operations

- **Indexing with preprocessing**
  - Unification of word forms during indexing
  - Terms normal forms of occurring word forms
  - Index is largely independent of the concrete formulation of the text
    - computerlinguistic approach

2.1.1 Indexing Without Preprocessing – String Operations

- Index: contains all the word forms occurring in the document
- Query:
  - Searching for specific word forms is possible (e.g. searching for “cars” but not for “car”)
  - To search for different word forms string operations can be applied
    - Operators for truncation and masking, e.g.
      - ? covers exactly one character
      - * covers arbitrary number of characters
    - Context operators, e.g.
      - {n} exact distance between terms
      - <n> maximal distance between terms
Index Without Preprocessing and Query

<table>
<thead>
<tr>
<th>Term</th>
<th>Dokument-IDs</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>D1,D3</td>
</tr>
<tr>
<td>accident</td>
<td>D1,D3</td>
</tr>
<tr>
<td>became</td>
<td>D2</td>
</tr>
<tr>
<td>because</td>
<td>D1</td>
</tr>
<tr>
<td>car</td>
<td>D1</td>
</tr>
<tr>
<td>cars</td>
<td>D2</td>
</tr>
<tr>
<td>died</td>
<td>D1</td>
</tr>
<tr>
<td>heavy</td>
<td>D1</td>
</tr>
<tr>
<td>in</td>
<td>D1,D2,D3</td>
</tr>
<tr>
<td>more</td>
<td>D2</td>
</tr>
<tr>
<td>of</td>
<td>D1</td>
</tr>
<tr>
<td>people</td>
<td>D1,D3</td>
</tr>
<tr>
<td>quarter</td>
<td>D2</td>
</tr>
<tr>
<td>registered</td>
<td>D2</td>
</tr>
<tr>
<td>truck</td>
<td>D3</td>
</tr>
<tr>
<td>vehicles</td>
<td>D2</td>
</tr>
</tbody>
</table>

Truncation and Masking: Searching for Different Word Forms

- **Truncation**: Wildcards cover characters at the beginning and end of words – prefix or suffix
  - `schreib*` finds `schreiben, schreibt, schreibt, schreibe, …`
  - `??schreiben` finds `anschreiben, beschreiben, but not verschreiben`

- **Masking** deals with characters in words – in particular in German, declensions and conjugation affect not only suffix and prefix
  - `schr??b*` can find `schreiben, schrieb`
  - `h??s*` can find `Haus, Häuser`

- **Disadvantage**: With truncation and masking not only the intended words are found
  - `schr??b*` also finds `schrauben`
  - `h??s*` also finds `Hans, Hanse,hausen, hassen` and also words in other languages like `horse`
**Context Operators**

Context operators allow searching for variations of text phrases

- **exact word distance**
  
  | Bezug [3] Telefonat |
  | Bezug nehmend auf unser Telefonat |

- **maximal word distance**
  
  | text <2> retrieval |
  | text retrieval |
  | text and fact retrieval |

For context operators to be applicable, the positions of the words must be stored in the index.

---

**Indexing Without Preprocessing**

- **Efficiency**
  
  - Efficient Indexing
  - Overhead at retrieval time to apply string operators

- **Wort forms**
  
  - User has to codify all possible word forms and phrases in the query using truncation and masking operators
  - No support given by search engine
  - Retrieval engine is language independent

- **Phrases**
  
  - Variants in text phrases can be coded using context operators
2.1.2 Preprocessing of the Index – Computerlinguistic Approach

- Each document is represented by a set of representative keywords or index terms
- An index term is a document word useful for remembering the document’s main themes
- Index contains standard forms of useful terms
  1. Restrict allowed terms
  2. Normalisation: Map terms to a standard form

Restricting allowed Index Terms

- Objective:
  - increase efficiency effectivity by neglecting terms that do not contribute to the assessment of a document’s relevance
- There are two possibilities to restrict allowed index terms
  1. Explicitly specify **allowed** index terms
     - controlled vocabulary
  2. Specify terms that are **not allowed** as index terms
     - stopwords
Stop Words

- Stop words are terms that are not stored in the index
- Candidates for stop words are
  - words that occur very frequently
    - A term occurring in every document is useless as an index term, because it does not tell anything about which document the user might be interested in
    - a word which occurs only in 0.001% of the documents is quite useful because it narrows down the space of documents which might be of interest for the user
  - words with no/little meanings
  - terms that are not words (e.g. numbers)
- Examples:
  - General: articles, conjunctions, prepositions, auxiliary verbs (to be, to have)
    - occur very often and in general have no meaning as a search criteria
  - application-specific stop words are also possible

Normalisation of Terms

- There are various possibilities to compute standard forms
  - N-Grams
  - stemming: removing suffixes or prefixes
**N-Grams**

- Index: sequence of characters of length N
  - Example: “persons”
  - 3-Grams (N=3): per, ers, rso, son, ons
  - 4-Grams (N=4): pers, erso, rson, sons
- N-Grams can also cross word boundaries
  - Example: “persons from switzerland”
  - 3-Grams (N=3):
    - er, ers, rso, son, ons, ns_, s_f, _fr, rom, om_, m_s, _sw,
    - swi, wit, itz, tze, zer, erl, rla, lan, and

**Stemming**

- Stemming: remove suffixes and prefixes to find a common stem, e.g.
  - remove –ing and –ed for verbs
  - remove plural –s for nouns
- There are a number of exceptions, e.g.
  - –ing and –ed may belong to a stem as in red or ring
  - irregular verbs like go - went - gone, run - ran - run
- Approaches for stemming:
  - rule-based approach
  - lexicon-based approach
Rules for Stemming in English

Kuhlen (1977) derived a rule set for stemming of most English words:

<table>
<thead>
<tr>
<th>Ending</th>
<th>Replacement</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ies</td>
<td>y</td>
<td>XY = Co, ch, sh, ss, zz oder Xx</td>
</tr>
<tr>
<td>XYes</td>
<td>XY</td>
<td>XY = VC</td>
</tr>
<tr>
<td>XYs</td>
<td>y</td>
<td></td>
</tr>
<tr>
<td>ies'</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Xes'</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>X's</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>X'</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>XYing</td>
<td>XY</td>
<td>XY = CC, XV, Xx</td>
</tr>
<tr>
<td>XYing</td>
<td>XYe</td>
<td>XY = VC</td>
</tr>
<tr>
<td>ied</td>
<td>y</td>
<td></td>
</tr>
<tr>
<td>XYed</td>
<td>XY</td>
<td>XY = CC, XV, Xx</td>
</tr>
<tr>
<td>XYed</td>
<td>XYe</td>
<td>XY = VC</td>
</tr>
</tbody>
</table>

Source: (Ferber 2003)

Problems for Stemming

- In English a small number of rules cover most of the words
- In German it is more difficult because also stem changes for many words
  - insertion of Umlauts, e.g. Haus - Häuser
  - new prefixes, e.g. laufen - gelaufen
  - separation/retaining of prefix, e.g.
    - mitbringen - er brachte den Brief mit
    - überbringen - er überbrachte den Brief
  - Irregular insertion of „Fugen“ when building composita
    - Schwein-kram, Schwein-s-haxe, Schwein-e-braten
- These problems that cannot be easily dealt with by general rules operating only on strings
**Lexicon-based Approaches for Stemming**

Principle idea: A lexicon contains stems for word forms

- complete lexicon: for each possible form the stem is stored
  - persons – person
  - running – run
  - ran – run
  - went – go
  - going – go
  - gone – go

- word stem lexicon: to each stem all the necessary data are stored to derive all word forms
  - Distinction of different flexion classes
  - specification of anomalies
  - Example: To compute the stem of *Flüssen*, the last characters are removed successively and the Umlaut is exchanged until a valid stem is found (Lezius 1995)

| Fall/Endung | n | en | sen | ...
|-------------|---|----|-----|---
| normal      | Flüssen- | Flüss-en | Flüs-sen | ...
| Umlaut      | Flüssen- | Flusse-n | Fluss-en | Flussen-

**Index with Stemming and Stop Word Elimination**

Index:

<table>
<thead>
<tr>
<th>Terms</th>
<th>Document IDs</th>
</tr>
</thead>
<tbody>
<tr>
<td>accident</td>
<td>D1, D3</td>
</tr>
<tr>
<td>car</td>
<td>D1, D2</td>
</tr>
<tr>
<td>cause</td>
<td>D3</td>
</tr>
<tr>
<td>crowd</td>
<td>D3</td>
</tr>
<tr>
<td>die</td>
<td>D1</td>
</tr>
<tr>
<td>drive</td>
<td>D3</td>
</tr>
<tr>
<td>four</td>
<td>D3</td>
</tr>
<tr>
<td>heavy</td>
<td>D1</td>
</tr>
<tr>
<td>injur</td>
<td>D3</td>
</tr>
<tr>
<td>more</td>
<td>D2</td>
</tr>
<tr>
<td>morning</td>
<td>D1, D3</td>
</tr>
<tr>
<td>people</td>
<td>D1, D3, D2</td>
</tr>
<tr>
<td>quarter</td>
<td>D2</td>
</tr>
<tr>
<td>register</td>
<td>D2</td>
</tr>
<tr>
<td>truck</td>
<td>D3</td>
</tr>
<tr>
<td>trucker</td>
<td>D3</td>
</tr>
<tr>
<td>vehicle</td>
<td>D2</td>
</tr>
<tr>
<td>vienna</td>
<td>D1, D2, D3</td>
</tr>
<tr>
<td>yesterday</td>
<td>D1</td>
</tr>
</tbody>
</table>
Indexing Variants

The index can be represented as a matrix. The values of the matrix represent, whether a term occurs in a document.

- Each column represents a document vector:
  - $v_{ec}(d_j) = (w_{j1}, w_{j2}, ..., w_{jn})$
  - the document $d_j$ contains a term $t_i$ if $w_{ij} > 0$

- Each row represents a term vector:
  - $v_{ec}(t_i) = (w_{1i}, w_{2i}, ..., w_{ni})$
  - the term $t_i$ is in document $d_j$ if $w_{ij} > 0$
**Boolean Document Vectors**

<table>
<thead>
<tr>
<th></th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
</tr>
</thead>
<tbody>
<tr>
<td>accident</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>car</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>cause</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>crowd</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>die</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>drive</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>four</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>heavy</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>injur</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>more</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>morning</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>people</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>quarter</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>register</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>truck</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>trucker</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>vehicle</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>vienna</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>yesterday</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

d1: heavy accident because of a heavy car accident 4 people died yesterday morning in vienna

d2: more vehicles in this quarter more cars became registered in vienna

d3: Truck causes accident in vienna a trucker drove into a crowd of people four people were injured

**Term weights**

- Not all terms are equally useful for representing the document contents
- The importance of the index terms is represented by weights associated to them
- Let \( t_i \) be an index term, \( d_j \) be a document, \( w_{ij} \) is a weight associated with \((t_i,d_j)\)

  - The weight \( w_{ij} \) quantifies the importance of the index term for describing the content of the document
    - A simple weight could be the frequency of the term, i.e. how often the term occurs in the document
  - (Stop words can be regarded as terms where \( w_{ij} = 0 \) for every document)  
    
    (Baeza-Yates & Ribeiro-Neto 1999)
**Classic IR Models - Basic Concepts**

- $t_i$ is an index term
- $d_j$ is a document
- $N$ is the total number of docs
- $T = (t_1, t_2, \ldots, t_k)$ is the set of all index terms
- $w_{ij} \geq 0$ is a weight associated with $(t_i, d_j)$
- $w_{ij} = 0$ indicates that term $t_i$ does not belong to $d_j$
- $\text{vec}(d_j) = (w_{1j}, w_{2j}, \ldots, w_{kj})$ is a weighted vector associated with the document $d_j$
- $g_i(\text{vec}(d_j)) = w_{ij}$ is a function which returns the weight associated with pair $(t_i, d_j)$
- $f_i$ is the number of documents containing term $t_i$

**Representing the Index**

- Some thoughts about the size of the Index:
  - Each document typically contains only a small fraction of all the terms
    - Assume the document collection (often also called corpus) consists of 1 million documents
    - It is not unrealistic to assume that there are 500,000 different terms
    - Assume each document contains 5000 words (about 10 pages) than even if each would occur only once in a document, 10% of the cells would be zero
    - Realistically, about 99% of the cells or more are zero.
    - A better representation of the matrix would to record only the things that do occur, that is the 1 position

*source: teaching material of Ribeiro-Neto*

*following (Manning et al. 2008, p. 4)*
**Representing the Index as an Inverted Index**

- The vector space model is usually implemented with an inverted index.
- The inverted index maps from the terms to the documents where they occur.
- The basic idea of the inverted index is shown in the figure:
  - The dictionary is a sorted set of terms (sometimes also called lexicon or vocabulary).
  - For each term, there is a sorted list that records which documents the term occurs in.
  - Each item in the list is called "posting", the list is called "posting list".

```
<table>
<thead>
<tr>
<th>term</th>
<th>postings</th>
</tr>
</thead>
<tbody>
<tr>
<td>accident</td>
<td>1 3 14 35 46 76</td>
</tr>
<tr>
<td>car</td>
<td>3 7 36 46 78 85 93 102</td>
</tr>
<tr>
<td>vienna</td>
<td>2 3 17 35 56 62 67</td>
</tr>
</tbody>
</table>
```

**Representing the Index as an Inverted List (cont.)**

- The dictionary can have additional information for each term, e.g., the length of the posting (i.e., the number of documents the term occurs in, usually called document frequency).
- The posting can simply be the ID of the document. However, it can also contain additional information, e.g., the frequency of the term in the document or its positions.
**Data Structures for the Inverted Index**

- The dictionary is usually kept in memory
  - For each index term a pointer references to a posting list
  - The posting lists can be stored on disk
- The posting lists can be implemented as
  - linked lists or
  - more efficient data structures that reduce the storage requirements (index pruning)
- To answer a query, efficient retrieval of posting lists is essential
  - Sorting of dictionary and posting lists can be useful for efficient query processing

---

**Implementing the Term Structure as a Trie**

- Sequentially scanning the index for query terms/posting lists is inefficient
- A trie is a tree structure
  - each node is an array, one element for each character
  - each element contains a link to another node

---


*Source: G. Saake, K.-U. Sattler: Algorithmen und Datenstrukturen – Eine Einführung mit Java. dpunkt Verlag 2004*
**The Index as a Trie**

- The leaves of the trie are the index terms, pointing to the corresponding position lists.
- Searching a term in a trie:
  - search starts at the root
  - subsequently for each character of the term the reference to the corresponding subtree is followed until
    - a leave with the term is found
    - search stops without success

(Saake, Sattler 2004)

**Patricia Trees**

- Idea: Skip irrelevant parts of terms
- This is achieved by storing in each node the number of characters to be skipped.
- Example:

(Saake, Sattler 2004)