Overview

- Large picture: document processing, storage, search
- Indexing
- Term significance and term weighting
  - Zipf’s law, TF*IDF, Signal to Noise Ratio
- Document similarity
- Processing: stop lists and stemming
- Other problems of text analysis
Documents and Surrogates

- Digitally stored, used for search, presentation, and selection
- Digitally stored, used for presentation and selection, not used for search
- Externally stored, not used for search

Document Processing

- The focus of document processing is
  - Extracting useful information from a document
  - Creating searchable document surrogates
Document processing and search
Indexing

- Act of assigning index terms to a document
- Identify important information and represent it in a useful way
- Indexing in traditional books
  - Book index (term index, topic index)
  - Figure index, citations, formula index

Indexing: From text to index

Intelligent Miner for Text turns unstructured information into business knowledge for organizations of any size, from small businesses to global corporations. This knowledge-discovery “toolkit” includes components for building advanced text-mining and text-search applications.

Intelligent Miner for Text offers system integrators, solution providers, and application developers a wide range of text-analysis tools, full-text retrieval components, and Web-access tools to enrich their business intelligence and knowledge management solutions. With Intelligent Miner, you can unlock the business information that is “trapped” in email, insurance claims, news feeds, and Lotus Notes, and analyze patent portfolios, customer complaint letters, even competitor Web pages.
Why indexing?

- Need some representation of content
- Can not use the full document for search
- Using plain surrogates in inefficient
  - We want to avoid a “brute force” approach to searching (string searching, pattern matching)
- Used in:
  - Find documents by topic
  - Define topic areas, relate documents to each other
  - Predict relevance between documents and information needs

Indexing language (vocabulary)

- A set of index terms
  - words, phrases
- Controlled vocabulary
  - Indexing language is restricted to a set of terms predefined by experts
- Uncontrolled vocabulary
  - Any term satisfying some broad criteria is legible for indexing
Characteristics of an Indexing Language

- **Exhaustivity** refers to the breadth coverage
  - The extent to which all topics are covered
- **Specificity** refers to the depth of coverage
  - The ability to express specific details
- Domain dependent - snow example

Indexing: Choices and problems

- Who does the indexing
  - Humans (manual)
  - Computers (automatic)
- Problems and trade-offs
  - Presence of digital documents
  - Cost
  - Consistency
  - Precision
Manual indexing

- High precision (human understanding)
- Supports advance forms of indexing
  - Role-based indexing, phrase indexing
- Problems
  - Expensive
  - Inherently inconsistent
  - Indexer-user mismatch
- Addressing problems
  - Indexing rules
  - Precoordinated indexing
    - (vodka, gin, rum) -> liquor

Thesauri

- Roget Thesaurus vs. IR thesaurus
- IR thesaurus provides a controlled vocabulary and connections between words. It specifies:
  - Standard words that has to be used for indexing (vodka, see liquor)
  - Relationships between words (broader, narrower, related, opposite terms)
Features of thesauri

- Coordination level
  - Precoordination, postcoordination
- Represented term relationships
- Number of entries for each term
- Specificity of vocabulary
- Control on term frequency
- Normalization of vocabulary

Working with thesauri

- Construction
  - User, automated, or automatic
- Usage
  - Using a thesaurus for indexing
  - Using a thesaurus for search
- Some years ago a thesaurus was a handbook for an IR system
Automatic indexing

- Inexpensive
  - The only practical solution for large volume of data
- Consistent
- Requires digital documents
- Problems
  - Less precise (computer does not understand text!)
  - Typically supports simple forms of indexing

Document processing for search

Intelligent Miner for Text turns unstructured information into business knowledge for organizations of any size, from small businesses to global corporations. This knowledge-discovery "toolkit" includes components for building advanced text-mining and text-search applications. Intelligent Miner for Text offers system integrators, solution providers, and application developers a wide range of text-analysis tools, full-text retrieval components, and Web-access tools to enrich their business-intelligence and knowledge management solutions. With Intelligent Miner, you can unlock the business information that is "trapped" in email, insurance claims, news feeds, and Lotus Notes, and analyse patent portfolios, customer complaint letters, even competitors' Web pages.
From Indexing to Search

- The results of indexing are used to create a searchable data structure:
  - an inverted file
  - a term document matrix

Inverted File

- Also known as a *Posting file* or *concordance*
  Contains, for each term of the lexicon, an inverted list that stores a list of pointers to all the occurrences of that term in the document collection

**Lexicon** (or vocabulary) is a list of all terms that appear in the document collection
Intelligent Miner for Text turns unstructured information into business knowledge for organizations of any size, from small businesses to global corporations. This knowledge discovery "toolbox" includes components for building advanced text mining and text search applications. Intelligent Miner for Text offers custom integrators, solution providers, and enterprises the means to meet the most demanding requirements, including text retrieval components, and Web-access tools to enrich their business intelligence and knowledge management solutions. With Intelligent Miner, you can unlock the business information that is "trapped" in email, insurance claims, news feeds, and Lotus Notes, and analyze patent portfolios, customer complaint letters, even competitors' Web pages.

Inverted File

- Document file and inverted file

```
intelligent

intelligent, text miner
business, knowledge management
```

Inverted file

**Doc1:** the cat is on the mat  
**Doc2:** the mat is on the floor

Inverted file

- `cat:doc1,1`
- `floor:doc2,5`
- `mat:doc1,5;doc2,1`
Granularity

- The granularity of an index is the accuracy to which it identifies the location of a term.
- The granularity depends on the document collection.
- The usual granularity is to individual documents.

Matrix representation

- Many-to-many relationship
- Term-document matrix
  - indexing
- Term-term matrix
  - co-occurrence
- Document-document matrix
  - Similarity
**Term-Document matrix**

- Rows represent document terms
- Columns represent documents

**Doc1:** the cat is on the mat  
**Doc2:** the mat is on the floor

<table>
<thead>
<tr>
<th></th>
<th>Doc1</th>
<th>Doc2</th>
</tr>
</thead>
<tbody>
<tr>
<td>cat</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>floor</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>mat</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

The word *floor* is present in the document 2

**Term-Document matrix**

- The cells can also represent word counts or other frequency indicator
- Storage problems  
  - n. of cells=n. of terms X n. of documents
- Matrix is sparse (i.e. many terms are 0 )
- Practically use topologically equivalent representations
Term-term matrix

- Square matrix whose rows and columns represent the vocabulary terms
- A nonzero value in a cell $t_{ij}$ means that the two terms occur together in some document or have some relationship

Document-document matrix

- Square matrix whose rows and columns represent the documents
- A nonzero value in a cell $d_{ij}$ means that the two documents have some terms in common or have some relationship (e.g. an author in common)
Principles of automatic indexing

- Grammatical and content-bearing words
- Specific vs. generic
- Frequent vs. non frequent
  - The more often the word is found in the document - the better term is it
  - The less often the word is found in other documents - the better term is it
- Words of phrases?

Zipf’s Law

- If the words that occur in a document collection are ranked in order of decreasing frequency, they follow the Zipf’s law
  \[
  \text{rank} \times \text{frequency} \approx \text{constant}
  \]

*If this law holds strictly the second most common word would occur only half as often as the most frequent one*
Optimal Term Selection

- The most frequently occurring words are those included by grammatical necessity (i.e. stopwords)

  the, of, and, a

- The words at the other end of the scale are poor index terms: very few documents will be retrieved when indexed by these terms

Thresholds

- Two thresholds can be defined when an automatic indexing algorithm is used:
  - high-frequency terms are not desirable because are often not significant
  - very low frequency terms are not desirable because their inability to retrieve many documents
Term Selection with Thresholds

What is a term?

- “bag of words”
  - In simple indexing we are neglecting the relationships among different words just considering the frequency

- Term Association
  - If two or more words occur often together then the pair should be included in the vocabulary (e.g. “information retrieval”)
  - It can be useful to consider the word proximity (e.g. “retrieval of information” and “information retrieval”)
Term Weighting

- With the term weighting we try to understand the importance of an index term for a document.
- A simple mechanism can be the use of the frequency of the term (tf) in the document, but it also necessary to consider the length of the documents and the kind of the documents.

Advanced Term Weighting

- Taking document into account
  - The frequency of a term in a documents should be compared with the length of the document
  - Relative frequency (frequency / length)
- Taking collection into account
  - Depending on the kind of document collection the same term can be more or less important.
  - The term computer can be very important in a collection of medical papers, but very common in a collection of document about programming
A relatively successful approach to automatic indexing uses TF*IDF term weighting.

Calculate the frequency of each word in the text, assign a weight to each term in each document which is
- proportional to the frequency of the word in the document (TF)
- inversely proportional to the frequency of the word in the document collection (IDF)

$w_{ij} \geq 0$ is a weight associated with $(k_i, d_j)$

Assumption of mutual independence ("bag of words" representation)
Calculating TF*IDF

\[ w_{ik} = f_{ik} \times \left( \log_2 \frac{N}{D_k} + 1 \right) \]

Where:
- \( N \) number of documents in the collection
- \( D_k \) number of documents containing term \( k \) (at least once)
- \( f_{ik} \) frequency of term \( k \) in document \( i \)

TF*IDF matrix

<table>
<thead>
<tr>
<th></th>
<th>term₁</th>
<th>term₂</th>
<th>( \ldots )</th>
<th>termₙ</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc₁</td>
<td>( w_{11} )</td>
<td>( w_{12} )</td>
<td>( w_{13} )</td>
<td>( \ldots )</td>
</tr>
<tr>
<td>doc₂</td>
<td>( w_{21} )</td>
<td>( w_{22} )</td>
<td>( w_{23} )</td>
<td>( \ldots )</td>
</tr>
<tr>
<td>( \ldots )</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>docₘ</td>
<td>( w_{m1} )</td>
<td>( w_{m2} )</td>
<td>( w_{m3} )</td>
<td>( \ldots )</td>
</tr>
</tbody>
</table>
Term Weighting with Signal to Noise Ratio

- Based on Shannon’s information theory
- In information theory information has nothing to do with meaning but refers to the unexpectedness of a word
  - If a word is easy to forecast the information carried is very little. There is no information in something that can be precisely predicted
- Common words do not carry much information (e.g. stopwords).
- Less common words are much more informative

Information as messages

- Suppose that we have a set of n possible messages (words) \( i=1,2,3,\ldots,n \) with probabilities of occurring \( p_i \)
- Since some message will occur,

\[
\sum_{i=1}^{n} p_i = 1
\]
We would like to define the information content $H$ of the sequence of messages. The entropy function satisfies some necessary assumptions:

$$H = \sum_{i=1}^{n} p_i \log_2 \left( \frac{1}{p_i} \right)$$

The information content of the single word $i$ is calculated as:

$$\log_2 \left( \frac{1}{p_i} \right)$$

The more probable is the word less information it carries.

$H$ is an average information content.
Noise of an Index Term

The noise associated to an index term $K$ for a collection of $N$ documents is calculated as

$$n_k = \sum_{i=1}^{N} \frac{f_{ik}}{t_k} \log_2 \left( \frac{t_k}{f_{ik}} \right)$$

Where $t_k = \sum_{i=1}^{N} f_{ik}$ is the total frequency of the word $k$ in the document collection.

Noise of an Index Term

Note that if $f_{ik} = 0$ for a particular document then

$$\frac{f_{ik}}{t_k} \log_2 \left( \frac{t_k}{f_{ik}} \right) = 0$$
Noise of an Index Term

- If a term appears just in one document $K$ (repeated $a$ times) then the noise is minimal: $t_k = a$

$$n_k = \frac{a}{a} \log_2 \frac{a}{a} = \log_2 1 = 0$$

- On the contrary the noise is max if the term do not carry any information (appears in many documents)

Signal to Noise Ratio

- The signal of term $k$ is

$$s_k = \log_2 t_k - n_k$$

- The weight $w_{ik}$ of the term $k$ in the document $i$ is

$$w_{ik} = f_{ik} \cdot s_k = f_{ik} \cdot [\log_2 t_k - n_k]$$
Term Discrimination Value TDV

- Measures the degree to which the use of a term will help to distinguish the document from one to another.
- A measure of how much a given term \( k \) contributes to separating a set of documents into distinct subsets.
- \( AVSIM = \) average similarity for the documents in the collection.
  \[ TDV = AVSIM_N - AVSIM_{N(no\ k)} \]

Add a good discriminator

Remove good discriminator

Set of documents
Term Discrimination Value TDV

- If TDV >>0 term is a good discriminator
- If TDV << 0 term is a poor discriminator
- If TDV ≅ 0 term is a mediocre discriminator
- TDV can be used as a term weight (together with term frequency) or used to select terms for indexing (as a threshold)

Simple Automatic Indexing

- Every character string not a stopword can be considered an index term
- Positional index: include information on filed and location
- Use some normalized form of the word
- Use of a threshold: eliminate high and low frequency terms as index terms
- Assign a term weight using statistics or some other mechanism
Automatic indexing

Stop lists

- Language-based stop list: words that bear little meaning (stopwords) and dropped from further processing
  - 20-500 English words (an, and, by, for, of, the, ...)
  - Subject-dependent stop lists
- Improve storage efficiency
- May cause problems
  - “to be or not to be”, AT&T, programming
- Removing stop words
  - From document
  - From query
Stoplist examples

C ACM text collection:
a, about, above, accordingly, across, after, afterwards, again, against, all, almost, alone, along, already, also, although, always, am, among, amongst, an, and, another, any, anybody, anyhow, anyone, anything, anywhere, apart, are, around, as, aside, at, away, awfully, b, be, became, because, become, becomes, becoming, been, before, beforehand, behind, being, below, beside, besides, best, better, between, beyond, both, brief, but, by, c, can, cannot, cant, certain, co, consequently, could, d, did, do, does, .....x, y, yet, you, your, yours, yourself, yourselves, z, zero, */, manual, unix, programmer's, file, files, used, name, specified, value, given, return, use, following, current, using, normally, returns, returned, causes, described, contains, example, possible, useful, available, associated, would, cause, provides, taken, unless, sent, followed, indicates, currently, necessary, specify, contain, indicate, appear, different, indicated, containing, gives, placed, uses, appropriate, automatically, ignored, changes, way, usually, allows, corresponding, specifying.

see also
http://www.dcs.gla.ac.uk/idom/ir_resources/linguistic_utils/stop_words

Stemming

- Are there different index terms?
  - retrieve, retrieving, retrieval, retrieved, retrieves…
- Stemming algorithm:
  - (retrieve, retrieving, retrieval, retrieved, retrieves) ⇒ retriev
  - Strips prefixes of suffixes (-s, -ed, -ly, -ness)
  - Morphological stemming
Porter’s stemming algorithm

- Based on a measure of vowel-consonant sequences:
  - measure m for a string \( [C](VO)^n[V] \) where \( C \) is a sequence of consonants and \( V \) is a sequence of vowels (including "y") (\( [\] \) indicates optional)
  - \( m=0 \) (see, by), \( m=1 \) (problem, eat, trees,ivy), \( m=2 \) (problems, privacy)

- Some Notation:
  - \( *<X> \) \( \rightarrow \) stem ends with letter X
  - \( *v \) \( \rightarrow \) stem contains a vowel
  - \( *d \) \( \rightarrow \) stem ends in double consonant
  - \( *o \) \( \rightarrow \) stem ends with a ce sequence where the final consonant is not w, x, y

- Algorithm is based on a set of condition-action rules:
  - old suffix \( \rightarrow \) new suffix
  - rules are divided into steps and are examined in sequence

- Good average recall and precision


---

Porter’s stemming algorithm

- A selection of rules from Porter’s algorithm:

<table>
<thead>
<tr>
<th>STEP</th>
<th>CONDITION</th>
<th>SUFFIX REPLACEMENT</th>
<th>EXAMPLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a</td>
<td>NULL</td>
<td>red \rightarrow red</td>
<td>create \rightarrow create</td>
</tr>
<tr>
<td></td>
<td>NULL</td>
<td>red \rightarrow red</td>
<td>stress \rightarrow stress</td>
</tr>
<tr>
<td></td>
<td>NULL</td>
<td>red \rightarrow red</td>
<td>possess \rightarrow possess</td>
</tr>
<tr>
<td>1b</td>
<td>&quot;y&quot;</td>
<td>NULL</td>
<td>making \rightarrow make</td>
</tr>
<tr>
<td>1c</td>
<td>&quot;y&quot;</td>
<td>NULL</td>
<td>happy \rightarrow hap</td>
</tr>
<tr>
<td>2</td>
<td>m &gt; 0</td>
<td>sell \rightarrow sell</td>
<td>remain \rightarrow remain</td>
</tr>
<tr>
<td>3</td>
<td>m &gt; 0</td>
<td>leave \rightarrow leave</td>
<td>digit \rightarrow digit</td>
</tr>
<tr>
<td>4a</td>
<td>m &gt; 1</td>
<td>sell \rightarrow sell</td>
<td>adjustable \rightarrow adjust</td>
</tr>
<tr>
<td>4b</td>
<td>m &gt; 1</td>
<td>leave \rightarrow leave</td>
<td>microscopic \rightarrow micro</td>
</tr>
<tr>
<td>5a</td>
<td>m &gt; 1</td>
<td>NULL</td>
<td>inflect \rightarrow infet</td>
</tr>
<tr>
<td>6a</td>
<td>m &gt; 1</td>
<td>NULL</td>
<td>control \rightarrow contol</td>
</tr>
<tr>
<td>6b</td>
<td>m &gt; 1</td>
<td>NULL</td>
<td>write \rightarrow write</td>
</tr>
</tbody>
</table>
Connections between document preparation and search

- If case conversion was used - can’t distinguish lower and upper cases in a query
- If stop list was used - can’t search by stop words
- If stemming is used can’t distinguish different forms of the same word

Document similarity

- Similarity measure is a key IR problem
- How to calculate document similarity?
- Lexical measures
  - Count term occurrences
  - Count term frequencies
- Document as a vector of terms
  - 0-1 vector
  - Weighted vector
Document Similarity: 0-1 Vector

任何文档都可以表示为一个向量或列表，其中包含在其内出现的术语。

\[ D = \langle t_1, t_2, t_3, \ldots, t_N \rangle \]

其中，成分 \( t_i \) 对应词汇中的第 \( i \) 个术语。

- \( t_i = 0 \) 如果术语不出现。
- \( t_i = 1 \) 或 \( w_i \) 如果术语出现。

Document Similarity

让 \( D_1 \) 和 \( D_2 \) 两个文档向量，其中成分 \( t_{1i} \) 和 \( t_{2i} \) 对于 \( i = 1, 2, \ldots, N \)。

我们定义:

- \( w \) = 数量的术语，其中 \( t_{1i} = t_{2i} = 1 \) (在两者中出现的)
- \( x \) = 数量的术语，其中 \( t_{1i} = 1 \) 和 \( t_{2i} = 0 \) (仅在第一个中出现的)
- \( y \) = 数量的术语，其中 \( t_{1i} = 0 \) 和 \( t_{2i} = 1 \) (仅在第二个中出现的)
- \( z \) = 数量的术语，其中 \( t_{1i} = t_{2i} = 0 \) (未在两者中出现的)

\[ n_1 = w + x \]
\[ n_2 = w + y \]
Matching document terms

\[ n_1 = w + x \]
\[ n_2 = w + y \]
\[ N = w + x + y + z \]

- **w** - terms present in both
- **z** - terms absent in both
- **x** and **y** - terms present in one of the documents

Measures

- **Basic measure:**
  \[ \delta = w - \frac{n_1 \times n_2}{N} \]

- **Measures of similarity:**
  \[ C(D_1, D_2) = \frac{\delta(D_1, D_2)}{\alpha} \]
  Where \( \alpha \) is:
  \[ \alpha(S) = \frac{N}{2} \] - separation
  \[ \alpha(R) = \text{max}(n_1, n_2) \] - rectangular distance
Document Similarity

- Define the *basic comparison unit*

\[ \delta(D_1, D_2) = \delta(D_2, D_1) = w - \frac{n_1n_2}{N} \]

- The basic comparison unit can be used as a measure of similarity defining a *coefficient of association*

\[ C_\alpha(D_1, D_2) = \frac{\delta(D_1, D_2)}{\alpha} \]

Document Similarity

- There are many different definition of \( \alpha \) and so many “similarity” definitions

- Some typical examples:
  - \( \alpha \) is:
    - \( \alpha(S) = N/2 \) - separation coefficient
    - \( \alpha(R) = \max(n_1, n_2) \) - rectangular distance
Document Similarity: Separation

\[
\text{z} \quad \begin{array}{c}
\text{x} \\
\text{w} \\
\text{y} \\
\text{D}_1 \\
\text{D}_2
\end{array}
\]

Document Similarity: Weighted Vector

- Similarity measures that depend on the frequency with which terms occur in a document can be based on a metric (distance measure)
- The greater the distance between documents, the less similar they are
Properties of a Metric

- A metric has three defining properties
  - its values are nonnegative, the distance between two points is 0 iff the points are identical \( d(A,B)=0 \iff A=B \)
  - it is symmetric \( d(A,B)=d(B,A) \)
  - it satisfies the triangle inequality \( d(A,B)+d(B,C) \geq d(A,C) \) for any points \( A, B \) and \( C \)

\[ L_p \text{ Metrics} \]

- Let \( D_1 \) and \( D_2 \) two document vectors with components \( t_{1i}, t_{2i} \) for \( i=1,2,\ldots,N \)
  \( D_1=<t_{11}, t_{12}, t_{13}, \ldots t_{1N}> \)
  \( D_2=<t_{21}, t_{22}, t_{23}, \ldots t_{2N}> \)
- The \( L_p \) metrics can be defined

\[
L_p(D_1,D_2) = \left[ \sum_i (t_{1i} - t_{2i})^p \right]^{1/p}
\]
Three Popular $L_p$ Metrics

- City block distance if $p=1$
- Euclidean distance if $p=2$
- Maximal direction if $p=\infty$

$$L_\infty(D_1, D_2) = \max_i \left| t_{1i} - t_{2i} \right|$$

City Block Distance

![Diagram of City Block Distance](image)
Euclidean Distance

Maximal Direction
Analysis beyond counting words?

- Natural Language Processing
- Pragmatics processing
  - Weighting sources, authors
- User-depending factors
  - User adaptation

Multi-language retrieval

- Most progress with English, but now there are IR systems for every language
- English is simple!
  - Separated verbs in German
  - Suffixes in Russian and Turkish
  - Vowels in Hebrew and Arabic
- Translation and multi-language systems
Homework

Exercise 1
Given the document representations
\[ D_1 = <4, 2, 0, 4> \]
\[ D_2 = <0, 3, 1, 0> \]
\[ D_3 = <1, 2, 0, 5> \]
\[ D_4 = <2, 0, 4, 3> \]
- calculate the distances between all the documents pairs for the three L metrics

Exercise 2
For a set of N=20 documents, calculate the noise associated to a term that appears twice in documents 1, 2, 3, ..., 19 and once in document 20.
Compare it with the noise associated to a term that appears 2 times in ALL documents.
Explain the results