Text Analysis and Information Retrieval

P. Héroux
University of Rouen Normandy

Data Science and Engineering
Data Science specialization

Introduction

- We are living in the era of big data
- One of the goal of data science is to extract knowledge embedded in the huge amount of data that is generated everyday
- However...

Introduction

- We are living in the era of big data
- One of the goal of data science is to extract knowledge embedded in the huge amount of data that is generated everyday
- However...
- Since 3000 BC, humans directly write down their knowledge through texts
- Texts and documents have been the support to store and transfert information and knwoledge

Introduction

- Hieroglyphs, Codices, Papyri, silk paper, paper
- Books (copyist Manuscripts, Printing)
- Correspondance (letter, telegraphs...),
- Journals, Documentation, Patents
- Libraries, Lessons, Encyclopedias

Introduction

- Hieroglyphs, Codices, Papyri, silk paper, paper
- Books (copyist Manuscripts, Printing)
- Correspondance (letter, telegraphs...),
- Journals, Documentation, Patents
- Libraries, Lessons, Encyclopedias
- and their digital counterparts

Introduction

- Digital files (txt, word processing, postscript/pdf files...)
- Web pages
- On-line journals
- Emails
- Social network posts

- It is difficult to imagine that an information / a knowledge has not been written as a text

Introduction

- Numerous initiatives to convert/encode “old” documents into digital documents
- Manual transcriptions
- Digitization campaigns / OCR

Introduction

- Huge amount of data
- Universal knowledge
- Computation power
- but...

Introduction

- Huge amount of data
- Universal knowledge
- Computation power
- but...
- how to handle text ?

What is a text (file)

- Sequence of bytes
- Encode characters
- Form words
- Which combines into sentences
- Paragraphs, sections, chapters, Titles
- TOC, Index, References, Links...

What is a text (file)

- Sequence of bytes Linear structure
- Encode characters
- Form words
- Which combines into sentences
- Paragraphs, sections, chapters, Titles
- TOC, Index, References, Links...

What is a text (file)

- Sequence of bytes Linear structure
- Encode characters Alphabet, character encoding, file format. . .
- Form words
- Which combines into sentences
- Paragraphs, sections, chapters, Titles
- TOC, Index, References, Links...

What is a text (file)

- Sequence of bytes Linear structure
- Encode characters Alphabet, character encoding, file format. . .
- Form words Vocabulary, spelling, meaning(s), synonyms, language. . .
- Which combines into sentences
- Paragraphs, sections, chapters, Titles
- TOC, Index, References, Links...

What is a text (file)

- Sequence of bytes Linear structure
- Encode characters Alphabet, character encoding, file format. . .
- Form words Vocabulary, spelling, meaning(s), synonyms, language. . .
- Which combines into sentences Grammatical structure, Relations between entities
- Paragraphs, sections, chapters, Titles
- TOC, Index, References, Links...

What is a text (file)

- Sequence of bytes Linear structure
- Encode characters Alphabet, character encoding, file format. . .
- Form words Vocabulary, spelling, meaning(s), synonyms, language. . .
- Which combines into sentences Grammatical structure, Relations between entities
- Paragraphs, sections, chapters, Titles
- TOC, Index, References, Links...
- All this is expressed in natural language
- Which is not easily manipulated by computers
- Lack of structure

Expected use

- Question answering :
  - Who wrote Romeo and Juliet?
  - What is the GDP of Brasil?
  - In which year ended the war of Indochina?

Expected use

- Question answering :
  - Who wrote Romeo and Juliet?
  - What is the GDP of Brasil?
  - In which year ended the war of Indochina?

All these questions could be answered

- by a human who knows
  - where to search
  - to read

Expected use

- Question answering :
  - Who wrote Romeo and Juliet?
  - What is the GDP of Brasil?
  - In which year ended the war of Indochina?

All these questions could be answered

- by a human who knows
  - where to search
  - to read
- automatically through SQL queries if we have a DBMS with the required informations.

Expected use

- Document classification (incoming mail)
- Filtering according to several criteria
- Review, Abstract
- Translation
- Detection of event in a document flow (trending topics, security issues. . .)

Outline

- Some tasks dealing with text analysis
  - Information Retrieval
  - Document classification
  - Information extraction
- Processing tasks

Information retrieval

- Considering an information need expressed by a user
- Considering a document collection
- Information retrieval aims at selecting the documents in the collection that contains information that satisfy the information need

Information retrieval (history)

- Initially performed by professionals (librarians, domain expert...)
- . . . on rather small corpora
- Nowadays, performed by anybody, even without knowing
- Not only web searches (emails, file systems. . .)
- #IR queries > #DBMS queries since 90’s

Boolean model

- grep based search is only tractable for small collections
- Ineffective for
  - large collections (#documents, `len(doc)`)
  - flexible queries (NEAR operator)
  - document ordering based on relevance
- Aim: Prevent the sequential search of a pattern in a sequence of files
- Solution: towards a constant time indexing ⇒ word based document indexing

Boolean model

term-document incidence matrix

- **M**
- $m_{i,j} = 1$ if t_i occurs in d_j
- $m_{i,j} = 0$ otherwise
- $M[i,:]$ gives the set of documents where t_i occurs
- $M[:,j]$ gives the set of terms contained in d_j

Boolean model

Exemple

	d_1	d_2	d_3	d_4	d_5
t_1	1	1	0	0	1
t_2	1	0	0	1	0
t_3	0	0	1	1	0
t_4	1	0	1	0	1
t_5	1	1	0	1	0

- what is the set of documents that contain t_1 and t_4 but do not contain t_3?
- 11001 AND 10101 AND NOT (00110) = 10001
- d_1 and d_5

Boolean model

Memory space

- a collection with 10^6 documents
- each document contains about 1000 terms
- average term length is 6
- a vocabulary of 500 000 terms
- results in a 6 Go collection
- Incidence matrix
  - of size $500\,000 \times 1\,000\,000$
  - contains at most 1 billion 1 (if each term occurs once at most)
  - $P(M[i,j] = 0) = 0.998$

Boolean model

Memory space

- Let us encode only the positive information
- Inverted index
- Each term t_i of the vocabulary is associated a list of documents that contain t_i

$$t_1 \longrightarrow \begin{array}{|c|c|c|c|c}\hline d_1 & d_3 & d_6 & d_{10} & \ldots \\ \hline\end{array}$$

$$t_2 \longrightarrow \begin{array}{|c|c|c|c|c}\hline d_1 & d_2 & d_4 & d_6 & \ldots \\ \hline\end{array}$$

$$t_3 \longrightarrow \begin{array}{|c|c|c|c|c}\hline d_2 & d_3 & d_5 & d_8 & \ldots \\ \hline\end{array}$$

Boolean model

Occurrence

(t, d) ou (t, d, pos)

posting list

- $(t, list(d))$
- $(t, list(d, pos))$
- $(t, list(d, liste(pos))$

Inverted index

- indexed on terms t ("constant time access")
- posting lists are ordered
  - on documents d (linear time access)
  - then, on positions *pos* (linear time access)

Boolean model

Inverted index building

1. For each *d* in the collection,
   1.1 find the terms in *d* ⇒ `list((d,t,pos))`
   1.2 sort this list according to *t*
   1.3 merge duplicates ⇒ `list(t,d,list(pos))`
2. For each *t* in the vocabulary
   2.1 merge the set of lists $\{\texttt{list(t,d,list(pos))}\}_{d \in c}$ into an index entry `(t,list((d,list(pos)))`

Boolean model

AND based queries

- q: t1 AND t2
- locate t1 in the key index (dictionnary)
- Load its corresponding posting list
- locate t2 in the key index (dictionnary)
- Load its corresponding posting list
- find the intersection between the two lists
- return the result

Boolean model

t1 $\longrightarrow$ $\boxed{1} \to \boxed{2} \to \boxed{4} \to \boxed{11} \to \boxed{31} \to \boxed{45} \to \boxed{173} \to \boxed{1$

t2 $\longrightarrow$ $\boxed{2} \to \boxed{31} \to \boxed{54} \to \boxed{101}$

Intersection $\Longrightarrow$

- Linear complexity if the posting lists are ordered

Boolean model

$$\begin{tikzcd}
t1 \arrow[r] & \boxed{1} \arrow[r] & \boxed{2} \arrow[r] & \boxed{4} \arrow[r] & \boxed{11} \arrow[r] & \boxed{31} \arrow[r] & \boxed{45} \arrow[r] & \boxed{173} \arrow[r] & \boxed{1} \\
t2 \arrow[r] & \boxed{2} \arrow[r] & \boxed{31} \arrow[r] & \boxed{54} \arrow[r] & \boxed{101}
\end{tikzcd}$$

Intersection $\Longrightarrow$

- Linear complexity if the posting lists are ordered

Boolean model

$$\begin{tikzcd}
\text{t1} \arrow[r] & \boxed{1} \arrow[r] & \boxed{2} \arrow[r] & \boxed{4} \arrow[r] & \boxed{11} \arrow[r] & \boxed{31} \arrow[r] & \boxed{45} \arrow[r] & \boxed{173} \arrow[r] & 1 \\
\text{t2} \arrow[r] & \boxed{2} \arrow[r] & \boxed{31} \arrow[r] & \boxed{54} \arrow[r] & \boxed{101}
\end{tikzcd}$$

Intersection $\Longrightarrow$

- Linear complexity if the posting lists are ordered

Boolean model

$$\begin{tikzcd}
\text{t1} \arrow[r] & \boxed{1} \arrow[r] & \boxed{2} \arrow[r] & \boxed{4} \arrow[r] & \boxed{11} \arrow[r] & \boxed{31} \arrow[r] & \boxed{45} \arrow[r] & \boxed{173} \arrow[r] & \boxed{1} \\
\text{t2} \arrow[r] & \boxed{2} \arrow[r] & \boxed{31} \arrow[r] & \boxed{54} \arrow[r] & \boxed{101} \\
\text{Intersection} \arrow[r, Rightarrow] & \boxed{2}
\end{tikzcd}$$

- Linear complexity if the posting lists are ordered

Boolean model

t1 $\longrightarrow$ $\boxed{1}\rightarrow\boxed{2}\rightarrow\boxed{4}\rightarrow\boxed{11}\rightarrow\boxed{31}\rightarrow\boxed{45}\rightarrow\boxed{173}\rightarrow\boxed{1}$

t2 $\longrightarrow$ $\boxed{2}\rightarrow\boxed{31}\rightarrow\boxed{54}\rightarrow\boxed{101}$

Intersection $\Longrightarrow$ $\boxed{2}$

- Linear complexity if the posting lists are ordered

Boolean model

$$\begin{tikzcd}
\text{t1} \arrow[r, long] & \boxed{1} \arrow[r] & \boxed{2} \arrow[r] & \boxed{4} \arrow[r] & \boxed{11} \arrow[r] & \boxed{31} \arrow[r] & \boxed{45} \arrow[r] & \boxed{173} \arrow[r] & \boxed{1} \\
\text{t2} \arrow[r, long] & \boxed{2} \arrow[r] & \boxed{31} \arrow[r] & \boxed{54} \arrow[r] & \boxed{101} \\
\text{Intersection} \arrow[r, Rightarrow] & \boxed{2}
\end{tikzcd}$$

- Linear complexity if the posting lists are ordered

Boolean model

$$\text{t1} \quad \longrightarrow \quad \boxed{1} \rightarrow \boxed{2} \rightarrow \boxed{4} \rightarrow \boxed{11} \rightarrow \boxed{31} \rightarrow \boxed{45} \rightarrow \boxed{173} \rightarrow \boxed{1$$

$$\text{t2} \quad \longrightarrow \quad \boxed{2} \rightarrow \boxed{31} \rightarrow \boxed{54} \rightarrow \boxed{101}$$

$$\text{Intersection} \quad \Longrightarrow \quad \boxed{2}$$

- Linear complexity if the posting lists are ordered

Boolean model

$$\begin{tikzcd}
\text{t1} \arrow[r] & \boxed{1} \arrow[r] & \boxed{2} \arrow[r] & \boxed{4} \arrow[r] & \boxed{11} \arrow[r] & \boxed{31} \arrow[r] & \boxed{45} \arrow[r] & \boxed{173} \arrow[r] & \boxed{1} \\
\text{t2} \arrow[r] & \boxed{2} \arrow[r] & \boxed{31} \arrow[r] & \boxed{54} \arrow[r] & \boxed{101} \\
\text{Intersection} \arrow[r, Rightarrow] & \boxed{2} \arrow[r] & \boxed{31}
\end{tikzcd}$$

- Linear complexity if the posting lists are ordered

Boolean model

$$t1 \quad \longrightarrow \quad \boxed{1} \rightarrow \boxed{2} \rightarrow \boxed{4} \rightarrow \boxed{11} \rightarrow \boxed{31} \rightarrow \boxed{45} \rightarrow \boxed{173} \rightarrow \boxed{1}$$

$$t2 \quad \longrightarrow \quad \boxed{2} \rightarrow \boxed{31} \rightarrow \boxed{54} \rightarrow \boxed{101}$$

$$\text{Intersection} \quad \Longrightarrow \quad \boxed{2} \rightarrow \boxed{31}$$

- Linear complexity if the posting lists are ordered

Boolean model

$$\begin{tikzcd}
\text{t1} \arrow[r] & \boxed{1} \arrow[r] & \boxed{2} \arrow[r] & \boxed{4} \arrow[r] & \boxed{11} \arrow[r] & \boxed{31} \arrow[r] & \boxed{45} \arrow[r] & \boxed{173} \arrow[r] & \boxed{1} \\
\text{t2} \arrow[r] & \boxed{2} \arrow[r] & \boxed{31} \arrow[r] & \boxed{54} \arrow[r] & \boxed{101} \\
\text{Intersection} \arrow[r, Rightarrow] & \boxed{2} \arrow[r] & \boxed{31}
\end{tikzcd}$$

- Linear complexity if the posting lists are ordered

Boolean model

t1 $\longrightarrow$ $\boxed{1} \to \boxed{2} \to \boxed{4} \to \boxed{11} \to \boxed{31} \to \boxed{45} \to \boxed{173} \to \boxed{1}$

t2 $\longrightarrow$ $\boxed{2} \to \boxed{31} \to \boxed{54} \to \boxed{101}$

Intersection $\Longrightarrow$ $\boxed{2} \to \boxed{31}$

- Linear complexity if the posting lists are ordered

Boolean model

```
Intersect($p_1, p_2$)
 1  $answer \leftarrow \langle\, \rangle$
 2  **while** $p_1 \neq$ nil and $p_2 \neq$ nil
 3  **do if** $docID(p_1) = docID(p_2)$
 4          **then** Add($answer, docID(p_1)$)
 5                 $p_1 \leftarrow next(p_1)$
 6                 $p_2 \leftarrow next(p_2)$
 7          **else** **if** $docID(p_1) < docID(p_2)$
 8                   **then** $p_1 \leftarrow next(p_1)$
 9                   **else** $p_2 \leftarrow next(p_2)$
10  **return** $answer$
```

Boolean model

Complex queries

- Boolean queries are built with logical operators AND, OR, NOT
- Optimization strategies
  - Process conjunctive queries with shorter posting lists first
  - Length of disjunctive queries results estimated as the sum of posting list length
- NEAR operator examine the position lists in order to filter the result

Boolean model

Overview

- documents are modeled as term sets
- queries are built with boolean operator
- The matching of a document is boolean
- The results is not structured/ordered
- The boolean model
  - has been used 40 years ago in the first IR sytems
  - is still used by some systems

Boolean model

Overview

- Some prefer this model because you know why a document is in the result
- The query session is an alternance of
  - query (re)formulations
  - result examinations
- It may be long to get a satisfying result
- It may result in long and complex queries
- It may require expertise in query formulation:
  - manipulation of boolean operators
  - how to built queries including polysemous terms

Vector model

Limits of the boolean model

- Only based on occurrence of query terms in the document
- The result is a set of documents that may be difficult to examine

Interested in

- Priority suggestion for the user
- Result not only based on incidence but also on the number of occurrences
- d_1 should be presented before d_2 if the number of occurrences of the query term t is greater in d_1 than in d_2
- a score $s(d, q)$

Vector model

Parametric index – zone index

- What are the documents written by W. Shakespeare published in 1603 containing the terms “Prince” and “Love” ?
- This query requires to determine the value of meta-data (author, publishing date, content)
- Need for dedicated inverted index for each type of meta-data
- It would also be possible an inverted index for each zone of a document (title, abstract, introduction, body, conclusion. . .)
- Alternative: A unique index in which occurrences indicate the meta-data or the document zone

Vector model

Parametric index – zone index

- Occurrences of a query term in some zones (or meta-data) may have a greater importance thant occurences in other type of zones
- Title, abstract, introduction may be considered more important zone than document body
- Each zone may be associated a weight.

$$\sum_i w_i = 1$$

- For a given document d
  - possibility to give a 0/1 score s_i for the query q in each zone
  - total score: $s(d, q) = \sum_i s_i w_i$

Vector model

Score $s(d, q)$

- The matching between the document and the query is no more boolean
- Compute the score between q and every document d
- $s(d, q)$ is a scalar real value
- Present the list of documents d to the user in the decreasing order $s(d, q)$

Vector model

weighting based on term frequency

- score: increasing function of the number of occurrences of query terms
- let q be a query built as a sequence of terms (without connector)

$$q = t_1 t_2 \ldots t_n$$

- for a given document $s(d, q) = \sum_{t_i \in q} s(t_i, q)$
- the score for a query term can be weighted according to the term frequency $s(d, q) = \sum_{i=1}^{n} tf_{t_i,d}$
- Other weighting schemes can be adopted

$$s(d, q) = \sum_{t_i \in q}^{n} wf_{t_i,d}$$

$$wf_{t,d} = \begin{cases} 1 + \log tf_{t,d} & \text{if} tf_{t,d} > 0 \end{cases}$$

Vector model

weight matrix

- $tf_{t,d}$: frequency of t in d
- $wf_{t,d}$: a weight which is an increasing function of $tf_{t,d}$
- vectorial bag-of-word description of documents
- a document is described by a vector where the i^{th} dimension corresponds to the weight associated to t_i
- The weight increases with (and only depends on) $tf_{t_i,d}$
- The term order has no influence on the description
- “`A is greater than B`” and “`B is greater than A`” have the same description
- “`B is less than or equal to A`” has a different description.

Vector model

Discriminative power of terms

- *tf* and *wf* weighting schemes assume that all terms have the same contribution to the score computation
- This contribution only depends on *tf* and whether t_i belongs or not to the query
- However this assuption does not hold
- Common terms should have less influence than specific terms
- A term has a low discriminative power if it occurs with the same frequency in every document
- A term has a high discriminative power for some documents if it only occurs those few documents

Vector model

Discriminative power of terms

1. Normalization by cf_t
   - number of occurrences of t in the whole collection
   - only distinguish frequent terms from non frequent terms
   - does not integrate the term distribution over the collection
2. Normalization by df_t
   - number of documents where t occurs
   - better indicator of the discriminative power

Vector model

Discriminative power of terms

t	cf_t	df_t
A	10000	250
B	10000	2500

- A and B have the same collection frequency
- A occurs in 250 documents
- B occurs in 2500 documents
- A has a greater discriminative power

Inverse document frequency : *idf*

$$idf_t = \log \frac{N}{df_t}$$

- N: collection size
- df_t: number of documents where t occurs
- idf_t is high for terms that occur in few documents
- idf_t is low for terms that occur in many documents
- idf_t is zero if t occurs in every document

Example for a collection with 100 000 documents

t	df_t	idf_t
the	100 000	0
health	10 000	1
disease	1 000	2
kidney	100	3
nephritic	10	4

Vector model

tfidf weighting

$$tfidf_{t,d} = tf_{t,d} \times idf_t$$

- given a term t_i, it increases with $tf_{t_i,d}$
- for a given document d_i, it decreases with df_{t,d_i}

Vector model

Other weighting schemes

- $wfidf_{t,d} = wf_{t,d} \times idf_t$
- Normalization by tf_{max}
- Integrate the length of the document.
  - $tf_{t_i,d}$ increases with $len(d)$
  - each term of the vocabulary has a higher probability to occur when $len(d)$ increases
  - $s(q,d)$ increases with $len(d)$
- Should we consider that a long document is more relevant for the user ?

Vector model

Learning the weighting scheme

- a document collection
- a parametric model for the weighting scheme
- document feature set
- a training set $\{(q, list(d))\}$
- an objective function = loss function + a regularization term
- loss function: a metric that measure the difference between the list of returned document and the expected ones

A machine learning algorithm is used to find the parameter set that minimizes the objective function on the training set

Vector model

Okapi BM25

$$wf_{t,d} = \log \frac{N - df_t + 0.5}{df_t + 0.5} . \frac{(k_1 + 1)tf_{t,d}}{k_1((1 - b) + b\frac{l_d}{l_{ave}}) + tf_{t,d}}$$

- l_d: length of d
- l_{ave}: average length
- N collection size
- k_1 and b are the parameters
- $1 \leq k_1 \leq 2$ and $b = 0.75$ is a setting that have given quite good results on a variety of collections

Vector model

document similarity

- each document is represented by a vector
- each dimension is weight for a term of the vocabulary
- Are we able to compute “distance” between d_1 et d_2 given $\vec{V}(d_1)$ et $\vec{V}(d_2)$?

$$s(d_1, d_2) = \left| \vec{V}(d_1) - \vec{V}(d_2) \right|$$

- Need to normalize, because documents may have different wf_t but the same relative term distribution.
- the document similarity is a function of the angle between $\vec{V}(d_1)$ and $\vec{V}(d_2)$

Vector model

document similarity

$$\begin{aligned} s(d_1, d_2) &= \frac{\vec{V}(d_1)}{|\vec{V}(d_1)|} \cdot \frac{\vec{V}(d_2)}{|\vec{V}(d_2)|} \\ &= \vec{v}(d_1).\vec{v}(d_2) \end{aligned}$$

$$\begin{aligned} \vec{v}(d_1) &= \frac{\vec{V}(d_1)}{|\vec{V}(d_1)|} \\ \vec{v}(d_2) &= \frac{\vec{V}(d_2)}{|\vec{V}(d_2)|} \end{aligned}$$

$s(d_1, d_2)$ is the dot product between $\vec{v}(d_1)$ and $\vec{v}(d_2)$, normalized versions of $\vec{V}(d_1)$ and $\vec{V}(d_2)$.

Vector model

document similarity

- $s(d_1, d_2) = 1$
  - $\vec{v}(d_1) = \vec{v}(d_2)$
  - $\vec{V}(d_1)$ and $\vec{V}(d_2)$ are colinear and same direction
  - d_1 et d_2 have the same relative term distribution
- $s(d_1, d_2) = 0$
  - $\vec{v}(d_1)$ and $\vec{v}(d_2)$ are orthogonal
  - $\vec{V}(d_1)$ and $\vec{V}(d_2)$ are orthogonal
  - d_1 and d_2 have no common terms

Vector model

document similarity

- k-NN search
  - the collection $\{d_i\}$ can be ordered according to the similarity $s(d, d_i)$.
  - supervised classification of documents (binary, multi-class, multi-label)
  - unsupervised classification

Vector model

Vector representation of queries

Queries are represented like documents

- Each query term is associated a weight
- The weight depends on its frequency (almost always 1)
- The vector representation is normalized by its length
- Missing terms have a 0 weight
- The query is a very short document
- Search for the nearest document in the collection with the cosinus similarity

Vector model

Vector representation of queries

1. For each document d_i, $s(q, d_i) = \vec{v}(q).\vec{v}(d_i)$ is computed
2. documents are returned in the order of decreasing $s(q, d_i)$
3. The returned list is truncated in order to return only the k nearest documents

Vector model

Computational cost

- N similarity computations
- Each similarity computation needs $|V|$ multiplications

Optimization

- The similarity value is not important, but only their ranking

$$\vec{v}(q) \propto \vec{V}(q)$$
$$\vec{v}(q).\vec{v}(d_1) < \vec{v}(q).\vec{v}(d_2) \Leftrightarrow \vec{V}(q).\vec{v}(d_1) < \vec{V}(q).\vec{v}(d_2)$$

- $\vec{V}(q)$ contains almost $|V|$ 0 weights

$$\vec{V}(q).\vec{v}(d_2) = \sum_t V_t(q).v_t(d) = \sum_{t \in q} V_t(q).v_t(d)$$

Vector model

Optimization

- $tf_{t,q} = 1$ most of the time

$$\vec{V}(q).\vec{v}(d) = \sum_{t \in q} v_t(d)$$

- Sorting a list N documents ⇒ manage a ranked list of k documents

Vector model

Approximating the *k* nearest documents

Objective: Reduce the computational cost

Motivation: *k* nearest document vector are not those that best fit the user need

Principle: Filter the collection in order to reduce the number of similarity computation

Vector model

Several strategies

- threshold on idf_t: terms with small idf_t have
  - a low contribution to $s(q, d)$
  - long posting lists
- Only consider documents that contain at least n query terms
- Champion sets
  - a set of M nearest documents is precomputed for each t in V
  - candidate documents for a query q are those from the union of the champion sets for $t \in q$

Vector model

Hierarchical search

Preprocessing:
- $\sqrt{N}$ documents are selected as *leaders*
- All others documents are linked to their nearest leader

Query:
- First round to select the leader that is the most similar to the query
- Second round to query the document registered to the nearest leader

Extensions:
- Documents register to a leaders
- The query is applied to documents registered to the b nearest leaders

Probabilistic model

Principles

- Information need
- Expressed as a query (uncomplete, inaccurate. . .)
- Internal representation of documents (boolean model, vector model. . .)
- IR system
  - has a partial, inaccurate and finally uncertain view of the user need
  - states an (uncertain) hypothesis converning document relevance wrt. user's information need
- The probabilistic model attempts to model these uncertainties through probability theory
- The document ranking is not built upon $s(q, d)$ but is made with $P(R|q, d)$

Probabilistic model

Bayes law

$$P(A,B) = P(A \cap B) = P(A|B)P(B) = P(B|A)P(A)$$

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} = \frac{P(B|A)P(A)}{P(B|A)P(A) + P(B|\bar{A})P(\bar{A})}$$

$P(A)$: *a priori* probability

$P(A|B)$: *a posteriori* probability

Probabilistic model

Probability based Ranking Principle

van Rijsbergen 79

If a reference retrieval system's respons to each request is a ranking of the documents in the collection in order of the decreasing probability of relevance to the user who submitted the request, where the probability are estimated as accurately as possible on the basis on the basis of whatever data have been made available to the system for this purpose, the overall effectiveness of the system to its user will be the best that is obtainable on the basis of those data

Probabilistic model

Probability based Ranking Principle

d:	a document
q:	a query
R:	Bernoulli random variable

- $R = 1$: relevant
- $R = 0$: non relevant

- Bayes decision rule. A document is considered as relevant if

$$P(R = 1|q, d) > P(R = 0|q, d)$$

$$P(R = 1|q, d) > \frac{1}{2}$$

- Documents with higher $P(R = 1|q, d)$ should be presented first
- Ordering based on decreasing values of

Probabilistic model

Probability computation

- Ranking based on $P(R = 1|q, d)$
- Unknown
- We need to estimate $P(R = 1|q, d)$ for each document
- Estimation based on available information
- Statistics

Probabilistic model

Binary Independant Model (BIM)

Hypothesis

- Relevance is binary
- Relevance for a document is independant for an other document
- terms occurrences are independant

Representation

A document d is représented by a binary vector $\vec{x}$ for term occurrences

$$d \rightarrow \vec{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix} \text{ with } \begin{cases} x_i = 1 \text{ if } t_i \text{ occurs in } d \\ x_i = 0 \text{ otherwise} \end{cases}$$

Probabilistic model

Binary Independant Model (BIM)

Hypothesis

- Relevance is binary
- Relevance for a document is independant for an other document
- terms occurrences are independant

Representation

A query q is représented e by a binary vector $\vec{q}$ for term occurrences

$$q \rightarrow \vec{q} = \begin{bmatrix} q_1 \\ \vdots \\ q_m \end{bmatrix} \text{ with } \begin{cases} q_i = 1 \text{ if } t_i \text{ occurs in } q \\ q_i = 0 \text{ otherwise} \end{cases}$$

Probabilistic model

Binary Independant Model(BIM)

Document ranking

based on $O(R|d,q)$

$$\begin{aligned}
O(R|d,q) &= O(R|\vec{x},\vec{q}) \\
&= \frac{P(R=1|\vec{x},\vec{q})}{P(R=0|\vec{x},\vec{q})} \\
&= \frac{\frac{P(R=1|\vec{q}).P(\vec{x}|R=1,\vec{q})}{P(\vec{x}|\vec{q})}}{\frac{P(R=0|\vec{q}).P(\vec{x}|R=0,\vec{q})}{P(\vec{x}|\vec{q})}} \\
&= \frac{P(R=1|\vec{q})}{P(R=0|\vec{q})}.\frac{P(\vec{x}|R=1,\vec{q})}{P(\vec{x}|R=0,\vec{q})} \\
&\propto \frac{P(\vec{x}|R=1,\vec{q})}{P(\vec{x}|R=0,\vec{q})}
\end{aligned}$$

Probabilistic model

Binary Independant Model(BIM)

Document ranking

$$
\begin{aligned}
O(R|d,q) &\propto \frac{P(\vec{x}|R=1,\vec{q})}{P(\vec{x}|R=0,\vec{q})} \\
&\propto \prod_{t=1}^{m} \frac{P(x_t|R=1,\vec{q})}{P(x_t|R=0,\vec{q})} \\
&\propto \prod_{t:x_t=1} \frac{P(x_t=1|R=1,\vec{q})}{P(x_t=1|R=0,\vec{q})} \cdot \prod_{t:x_t=0} \frac{P(x_t=0|R=1,\vec{q})}{P(x_t=0|R=0,\vec{q})}
\end{aligned}
$$

Probabilistic model

Binary Independant Model(BIM)

Document ranking

$$
\begin{aligned}
O(R|d,q) &\propto \prod_{t:x_t=1} \frac{P(x_t=1|R=1,\vec{q})}{P(x_t=1|R=0,\vec{q})} \cdot \prod_{t:x_t=0} \frac{P(x_t=0|R=1,\vec{q})}{P(x_t=0|R=0,\vec{q})} \\
&\propto \prod_{t:x_t=1} \frac{p_t}{u_t} \cdot \prod_{t:x_t=0} \frac{1-p_t}{1-u_t}
\end{aligned}
$$

$p_t = P(x_t = 1|R = 1, \vec{q})$: probability that t occurs in a relevant document d for the query q

$u_t = P(x_t = 1|R = 0, \vec{q})$: probability that t occurs in a non relevant document d for the query q

Probabilistic model

Binary Independant Model(BIM)

Document ranking

We assume that the terms that are not in the query have the same probability to occur in relevant and non relevant documents

$$
\begin{aligned}
O(R|d,q) &\propto \prod_{t:x_t=1} \frac{p_t}{u_t} . \prod_{t:x_t=0} \frac{1-p_t}{1-u_t} \\
&\propto \prod_{t:x_t=q_t=1} \frac{p_t}{u_t} . \prod_{t:x_t=0,q_t=1} \frac{1-p_t}{1-u_t} \\
&\propto \prod_{t:x_t=q_t=1} \frac{p_t(1-u_t)}{u_t(1-p_t)} . \prod_{t:q_t=1} \frac{1-p_t}{1-u_t} \\
&\propto \prod_{t:x_t=q_t=1} \frac{p_t(1-u_t)}{u_t(1-p_t)}
\end{aligned}
$$

Probabilistic model

Binary Independant Model(BIM)

Retrieval Status Value

$$
\begin{aligned}
RSV &= \log \prod_{t:x_t=q_t=1} \frac{p_t(1-u_t)}{u_t(1-p_t)} \\
&= \sum_{t:x_t=q_t=1} \log \frac{p_t(1-u_t)}{u_t(1-p_t)} \\
&= \sum_{t:x_t=q_t=1} c_t
\end{aligned}
$$

with

$$c_t = \log \frac{p_t}{1-p_t} + \log \frac{1-u_t}{u_t}$$

Probabilistic model

Binary Independant Model(BIM)

Retrieval Status Value

$$c_t = \log \frac{p_t}{1-p_t} + \log \frac{1-u_t}{u_t}$$

- $c_t = 0$ if t $P(t|R=1,q) = P(t|R=0,q)$
- How can we estimate c_t ?

Probabilistic model

Binary Independant Model(BIM)

$c_t estimate$

Documents	Relevant	Non relevant	Total
$x_t = 1$	s	$df_t - s$	df_t
$x_t = 0$	$S - s$	$(N - df_t) - (S - s)$	$N - df_t$
Total	S	$N - S$	N

$$p_t = \frac{s}{S} \quad u_t = \frac{df_t - s}{N - S}$$

$$c_t = \log \frac{s}{S - s} - \log \frac{df_t - s}{(N - df_t) - (S - s)}$$

$$\hat{c}_t = \log \frac{s + 0.5}{S - s + 0.5} - \log \frac{df_t - s + 0.5}{(N - df_t) - (S - s) + 0.5}$$

Probabilistic model

Binary Independant Model(BIM)

c_t estimate

- Non relevant documents are majoritary
- Statistics for non relevant documents are approximated by statistics on the whole collection

$$\log \frac{N - df_t}{df_t} \approx \log \frac{N}{df_t}$$

- Turns into the *idf* weighting scheme

Probabilistic model

Binary Independant Model(BIM)

- The same reasoning can not be used to estimate p_t
- Several approaches:
  - Approximéated by a frequency on documents known as relevant
  - p_t approximated by a constant (Croft et Harper, 1979)
  - p_t iteratively estimated (probabilistic approach of relevance feedback)

Probabilistic model

Binary Independant Model(BIM)

1. A first proposal (based on another approach) ⇒ provides a document set V
2. User interaction

$$VR \subset V \quad VNR \subset V$$

3. p_t (and u_t) are estimated again by examining VR et VNR.

$$p_t = \frac{|VR_t|}{|VR|} \quad p_t^{(k+1)} = \frac{|VR_t| + \kappa p_t^{(k)}}{|VR| + \kappa}$$

Probabilistic model

Binary Independant Model(BIM)

$$RSV = \sum_{t:x_t=q_t=1} c_t \text{ with } c_t = \log\left[\frac{|V_t|}{|V|-|V_t|} \cdot \frac{N}{df_t}\right]$$

- term occurrences are assumed to be independant
- Only query terms are considered
- Relevance is assumed as binary
- Relevance for a document is assumed to be independant from the relevance of other documents

Probabilistic model

Extension

OkapiBM25

- OkapiBM25 is not a binary model
  - term frequency
  - document length
- Several formulas

Probabilistic model

Okapi BM25

Formulas

- The simplest turns into the *idf* weighting scheme

$$RSV_d = \log\left[\frac{N}{df_t}\right]$$

Probabilistic model

Okapi BM25

Formulas

- Term frequency and document length are considered

$$RSV_d = \log \left[\frac{N}{df_t} \cdot \frac{(k_1+1)tf_{t,d}}{k_1((1-b)+b\frac{l_d}{l_{ave}})+tf_{t,d}} \right]$$

Probabilistic model

Okapi BM25

Formulas

- Query length is considered

$$RSV_d = \log \left[\frac{N}{df_t} \cdot \frac{(k_1+1)tf_{t,d}}{k_1((1-b)+b\frac{l_d}{l_{ave}})+tf_{t,d}} \cdot \frac{(k_3+1)tf_{t,q}}{k_3+tf_{t,q}} \right]$$

Probabilistic model

Okapi BM25

Formulas

- Relevance feedback

$$RSV_d = \log \left[\frac{\frac{|VR_t|+0.5}{|VR|-|VR_t|+0.5}}{\frac{df_t-|VR_t|+0.5}{N-df_t-|VR|+|VR_t|+0.5}} \cdot \frac{(k_1+1)tf_{t,d}}{k_1((1-b)+b\frac{l_d}{l_{ave}})+tf_{t,d}} \cdot \frac{(k_3+1)tf_{t,q}}{k_3+tf_{t,q}} \right]$$

OkapiBM25

Paramètres

- k_1 and k_3 are positive real values that allow to take term frequency in document and query respectively into account.
- A null value correspond to a binary model
- $0 \leq b \leq 1$ allows to take the document length into account
- These parameters can be optimized on a training collection
- $k_1 = k_3 = 2$ et $b = 0.75$ are common default values

Language models

Introduction

- The document ranking in probalistic model is based on the relevance probability considering a query
- Language models are an other kind of probabilistic models
- They use a generative approach
- A language model $\mathcal{M}_d$ s built for each document d.
- The document ranking is based on $P(q|\mathcal{M}_d)$, the probability that $\mathcal{M}_d$ the model representing the document d could have generated the query q

Language models

Introduction

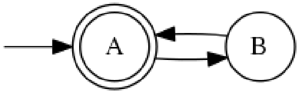
- This model assumes that user that want to find some relevant document use query with terms and syntaxes similar to relevant documents
- A document should have a good matching with the query if the language model used by the document might also have generated the query

Language models

A language model expresses conditional probability distributions

Exemple

- A deterministic automaton



- This automaton can generate the following sequences:
  - A B
  - A B A B
  - A B A B A B
  - . . .

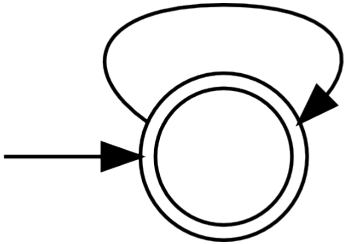
Language models

- In language models, each vertex of the automaton is a probability distribution
- A probabilistic (or stochastic) language model is a function that allows to measure the probability of generating a term sequence
- The simplest example is a finite automaton with a single vertex that integrates a term probability distribution

Language models

Simple example

Graphic model



Probability distribution

the	0.1
a	0.1
is	0.1
in	0.05
this	0.05
my	0.05
your	0.05
phrase	0.0005
sentence	0.0005
⋮	⋮

Language models

- The language model allows to compute the probability that it may have generated a term sequence

$$P(\text{"this is my sentence"}) = 0.05 \times 0.1 \times 0.05 \times 0.0005$$

- Given several language models, it is possible to rank them in the decreasing order of probability for a sequence.

Language models

Term sequence probability

- The probability of a term sequence "$t_1 t_2 t_3 t_4$" is

$$P(t_1 t_2 t_3 t_4)$$

- It might be split into

$$P(t_1 t_2 t_3 t_4) = P(t_1)P(t_2|t_1)P(t_3|t_1 t_2)P(t_4|t_1 t_2 t_3)$$

Laguage models

Term sequence probability

Unigram model

- Term probability is assumed to be independant from the context

$$P(t_1 t_2 t_3 t_4) = P(t_1)P(t_2)P(t_3)P(t_4)$$

- Term order has no influence
- Bag-of-word description

Laguage models

Probabilité de production d'une séquence de termes

Bigram model

- The context is limited to the fromer term

$$P(t_1 t_2 t_3 t_4) = P(t_1)P(t_2|t_1)P(t_3|t_2)P(t_4|t_3)$$

- More complex language models can be learned (Deep recurrent Neural Networks, LSTM) to describe a collection. A single document does not provide enough data.

Language models

Query likelihood

- Given q, we want a ranking based on $P(d|q)$

$$P(d|q) = \frac{P(q|d)P(d)}{P(q)}$$

- $P(q)$ is a constant for all document
- $P(d)$ can be stated as $\frac{1}{N}$ which is similar for all documents
- Documents can then be ranked based on $P(q|d)$
- d is modeled by $\mathcal{M}_d$
- Finally documents are ranked based on $P(q|\mathcal{M}_d)$, the probability that the query q might have been producted by the language model of de d.
- However, $P(d)$ can also be set according to other prior information (source reliability, prior relevance, hits, ...)

Language model

Multinomial Unigram Laguage Model

- Similar to a bayesian model where each document is considered as a distinct class
- The aim is then to classify queries
- Product of term occurrence probability in the language model of the document

$$P(q|\mathcal{M}_d) = \prod_{t \in Vocabulary} P(t|\mathcal{M}_d)^{c(t)}$$

Corpus identification

- Identify the document set to index
- Define the notion of document:
  - A file ?
  - What about an archive / mailbox file ?
  - What about an attached file in a mail ?
  - A set of html files (electronic book) ?
  - A long file containing several books ?
- (Un)bounded document set ? ⇒ crawling
- Static / evolving document set ⇒ Index update

Text extraction

- Document encoding of raw text file
  - Given
  - Heuristics
  - Estimated through classification
- Extracting the raw text content
  1. Determinine the file format (html, postscript, pdf, wordprocessing) (given, heuristics, estimated through classification)
  2. Use the dedicated reader (pdf2txt, beautiful soup, entity characters. . .)
- Language identification can also be useful as it might influence subsequent processing tasks

Text segmentation

- Aims at spliting a character sequence into tokens
- Tokens are words as they occur in the document
- Heuristic rules:
  - Remove punctuation (replace by a space)
  - Split on blank characters (space, tabulation, newline)
- Special cases:
  - Acronyms, hyphens, hyphened words (end of line), intra-word splitting
  - Language specific rules
- Towards index entries
- Trained tokenizer with contextual features

Text segmentation

Examples

San Francisco-Los Angeles flights	B-52
C#	C++
www.univ-rouen.fr	193.52.145.35
Un laissez-passer	Comment va-t-il ?
30/11/2017	U.S.A.
Lebenversicherungsgesellsaftangstellter	和尚
aren’t / don’t	won’t

Normalization

- From tokens to term occurrences
- Token: a word as it occurs in the document
- Term: Index entry
- Several steps:
  - Lowercase ?
  - Diacritics ?
  - Stemming ?
  - Lemmatization ?

Lowercase / Diacritics

Diacritics

- Removed most of the times
- Difficult keyboard manipulations / often omitted in queries
- Introduce ambiguities

Lowercase, but. . .

- Language dependant
- Introduce ambiguities
- Help to determine names entities (people, title, companies, organizations. . .)

Lowercase

Examples

- CAT / a cat
- FED / fed
- George Bush / a bush
- President of the United States
- United Kingdom

Stemming / Lemmatization

Aim

Reduce the derived forms of a word to a unique term (index entry)

Stemming

- Applies a series of (empirical/heuristic) rules that replace suffix patterns
- Until no more rule can be applied
- Priority level concurrent rules management
- The reduced form is not a “valid” word

Lemmatization

- Uses external linguistic ressources (thesaurus)
- Ambiguities resolved by contextual information (POST)
- The reduced form is a “valid” word

Stemming / Lemmatization

- Language dependant
- Affixal approach (mostly suffix)
- Tries to resolve grammatical rules such as
  - conjugation
  - agreement
  - difficult management of special cases by automatic processing
  - a token can correspond to several terms

Stemming / Lemmatization

Stemming for english

- Lovins stemmer
- Paice stemmer
- Porter stemmer

Lemmatization

- Wordnet

How to rate the quality of an IR system?

Several criterions

- Several points of view
  - Document editors: SEO
  - IR system manager: Indexing speed
  - Information need: number of indexed documents, latency, information need satisfaction, interface, extra functionalities
- Not always easy to measure

Information need satisfaction

Measures based on binary relevance

- A set of document
- A set of (representative) queries
- For each query, the set of relevant documents (binary classification problem)
- Classification accuracy is not a good indicator (imbalanced class)
- Given a set of “returned” documents
  - TP: number of returned documents that are relevant
  - FP: number of returned documents that are not relevant
  - TN: number of unreturned documents that are not relevant
  - FN: number of unreturned documents that are relevant

Information need satisfaction

Precision / Recall

$$P = \frac{TP}{TP + FP}$$

$$R = \frac{TP}{TP + FN}$$

$$F_\alpha = \frac{1}{\frac{\alpha}{P} + \frac{1-\alpha}{R}}$$

Information need satisfaction

Precision-Recall curve

1. For a given query, the IR system returns a ranking of the document collection
2. Precision and recall are computed for the k first element of the ranking
3. k range from 1 to N
4. At each step,
   - if the k^{th} is not relevant, P decreases and R is constant
   - else both P and R increase
5. Interpolated precision:

$$IP(R) = \max_{r \geq R} P(r)$$

6. Mean of P-R curve over the set of representative queries

Information need satisfaction

Other measures

- Area under the ROC Curve (TPR(FPR))
- Break Event Point (P=R)
- $P(k)$ Mean of precision computed of the first k documents
- Mean Average Precision: Mean of P after the last relevant document is returned

Other functionalities

- Query terms / index terms alignment
- Wildcards
- Relevance feedback
- Query rephrasing

Alignement between query and index terms

- The normalized form of a query term is not an index entry
- Try to interpret the user meaning
  - Run the query with “nearest” index term entries
  - Edit (Levenstein) distance between query and index terms
    - Minimum number of edit operations needed to transform the query term into the index entry
    - Adjustable costs (keyboard character proximity)
  - Soundex term indexing
    - Assumes a phonetic correspondance between the query term and the index term
    - Each word is coded with a 4 character code (initial letter, the 3 numbers that express the consonant classes)

Alignement between query and index terms

Documents or terms can be indexed by n-grams

n-grams indexing of terms

- The query term is decomposed into n-grams
- Index terms whose n-gram decomposition overlaps the query n-gram list by at least a threshold are kept and used in a disjuctive query

n-gram indexing of documents

Query terms with wildcards

abc*

- Binary tree of terms
- A disjunctive query is launched with all terms in the subtree of terms beginning with “abc”

*abc

- Binary tree of terms built on the right to left reading order
- A disjunctive query is launched with all terms in the subtree of terms ending with “abc”

Query terms with wildcards

abc*efg

- Each term is indexed several times in a permuterm index
- Every rotation of the term is inserted in a binary tree
- abc → `abc$, bc$a, c$ab, $abc`
- Each rotation is associated to the original term
- a query a∗c is rotated so that the '*' character is a the end → c$a∗
- We look in the permuterm index the terms in the subtree of c$a
- A disjuctive query based on these terms is launched

Relevance feedback

1. The query is first launched
2. The IR system present some documents
3. The user label some of them as relevant or non relevant
4. A new query (representation) is build based on relevance feedback

Relevance feedback

Rocchio's algorithm

- Finds a query representation that:
  - maximizes its similarity with relevant documents
  - minimizes its similarity with non relevant documents

$$\vec{q}_m = \alpha \vec{q}_0 + \frac{1}{|D_p|} \sum_{\vec{d}_j \in D_p} \vec{d}_j - \gamma \frac{1}{|D_{np}|} \sum_{\vec{d}_j \in D_{np}} \vec{d}_j$$

Query rephrasing

Aim

- Resolve ambiguous terms
- Query expansion (synonyms)

Use of a thesaurus

- Polysemy: Suggestion of conjunctive query with topic related terms that resolve polysemy
- Synonyms: Suggestion of disjunctive query build with synonyms

Query rephrasing

- History of query sessions: initial queries and their rephrased forms
- Topic-based index through equivalence class
  - Equivalence class: set of synonym terms
- Topic modeling
  - Statistical learning / Probability estimation of the contribution of a term to a topic
  - Determined through supervised classification
- Automatic thesaurus: term collocations
- Latent Semantic Indexing

Latent Semantic Indexing

term-document occurrence matrix

$$X = \begin{pmatrix} x_{1,1} & \dots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \dots & x_{m,n} \end{pmatrix}$$

- $x_{i,j}$: frequency of term i in document j
- $t_i^T = \begin{pmatrix} x_{i,1} & \dots & x_{i,n} \end{pmatrix}$: vector that gives the importance of term i over the documents
- $d_j = \begin{pmatrix} x_{1,j} \\ \vdots \\ x_{m,j} \end{pmatrix}$: vector that gives the importance of each term in document j

Latent semantic Indexing

Correlation

- $t_i^T t_p$ gives the correlation between terms i and p over the collection
- $d_j^T d_q$ gives the correlation between documents j and q over the lexicon
- XX^T gathers all dot products $t_i^T t_p$
- $X^T X$ gathers all dot products $d_j^T d_q$

Singular Value Decomposition

$$X = U\Sigma V^T$$

$$\begin{aligned}
XX^T &= (U\Sigma V^T)(U\Sigma V^T)^T \\
&= (U\Sigma V^T)(V^{T^T}\Sigma^T U^T) \\
&= U\Sigma V^T V\Sigma^T U^T \\
&= U\Sigma\Sigma^T U^T
\end{aligned}$$

$$\begin{aligned}
X^T X &= (U\Sigma V^T)^T(U\Sigma V^T) \\
&= (V^{T^T}\Sigma^T U^T)(U\Sigma V^T) \\
&= V\Sigma^T U^T U\Sigma V^T \\
&= V\Sigma^T\Sigma V^T
\end{aligned}$$

Décomposition en valeurs singulières

- $\Sigma\Sigma^T$ and $\Sigma^T\Sigma$ are diagonal
- U eigenvectors of XX^T
- V eigenvectors of X^TX

$$X = \left(\begin{pmatrix} u_1 \end{pmatrix} \dots \begin{pmatrix} u_l \end{pmatrix}\right) . \begin{pmatrix} \sigma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_l \end{pmatrix} . \begin{pmatrix} (& v_1 &) \\ & \vdots & \\ (& v_l &) \end{pmatrix}$$

Concept space

- U gives the principal axis system in the term space
- V gives the principal axis system in the document space
- k most greatest values of Σ are selected and the corresponding axis in U and V
- Projection on the most meaningful subspace of dimension k
$$X_k = U_k \Sigma_k V_k^T$$
- X_k is the approximation of X
- Term vectors and document vector are projected in the "automatically" determined concept space
- $\hat{t}_i$ has k dimensions that gives the weight of term i in each concept k
- $\hat{d}_j$ has k dimensions that gives the weight of document j in each concept k

Latent Semantic Indexing

Using

- We can evaluate the correlation between d_i and d_j in the concept space

$$\cos\theta = \frac{\hat{d}_i.\hat{d}_j}{\left\|\hat{d}_i\right\|.\left\|\hat{d}_j\right\|}$$

- We can also compare the correlation between t_i and t_j in the concept space
  - polysemy
  - synonimy
  - translation
- A query can be processed in the concept space

$$\hat{q} = \Sigma_k^{-1} U_k^T q$$

Document classification

- Usual classification tasks where items to be classified are documents
- Use of textual features:
  - Characters: distributions, occurrences, sequences, special types of characters (uppercase, punctuation, consonnants, emoticons. . .)
  - Words: distribution, occurrences, sequences, grammatical type, length
  - Sentences: length, structure, grmmatical components
  - Documents: length, #paragraph #sentence, meta-data (title, author, format, date). . .

Document classification

- Unsupervised classification: collection structuration, discovery of trending topics. . .
- Binary classification: Spam detection, very unbalanced classes, (IR, security issues. . .)
- Multi-class problem: sort a document stream several channels, encoding/language identification, sentiment analysis. . .
- Multi-label problem: several topics may be affected to each document
- Regression: review/comments analysis

Document classification

- Given a collection of document
- Provides a segmentation of the collection into groups
- Documents inside a group should be similar from the point of view of the user
- Groups should be “meaningful”

Unsupervised classification

- Flat clustering
  - k-means
  - EM
- Hierarchical clustering
  - Agglomerative
  - Divisive
- Incremental / Evolution

Supervised classification

- Given a set of labels

$$\mathbb{C} = \{c_1, \ldots, c_j\}$$

- Document space $\mathbb{X}$
- Given $\mathbb{D}$, a training dataset of labeled documents

$$\mathbb{D} = \{(d, c)\}$$

$$(d, c) \in \mathbb{X} \times \mathbb{C}$$

- Learn a function $\gamma : \mathbb{X} \to \mathbb{C}$

Naive Bayes Classifier

$$\gamma(d) = \arg\max_{c \in \mathbb{C}} P(c|d)$$

Multinomial Naive Bayes Classifier

$$\begin{aligned}
\gamma(d) &= \arg\max_{c \in \mathbb{C}} P(c|d) \\
&= \arg\max_{c \in \mathbb{C}} \hat{P}(c) \prod_{t \in Dict} \hat{P}(t|c) \\
&= \arg\max_{c \in \mathbb{C}} (\log \hat{P}(c) + \sum_{t \in Dict} \log \hat{P}(t|c))
\end{aligned}$$

Other classifiers

- k Nearest Neighbours
- Decision Trees / Random Forests
- Perceptron, Logistic regression
- Support vector Machines
- Multi Layer Perceptron, (Deep) Neural Networks

information extraction

- A document is given
- The aim is to locate and identify a specific type of information:
  - Normalized patterns:
    - Numeric fields (phone number, IP address, client ids)
    - email address, URL. . .
    - dates
  - Unnormalized pattern:
    - Named entities: people designatd by their nouns or their functions, organizations, companies, locations. . .

Information extraction

- Normalized patterns:
  - Handcrafted regular expressions
  - Exactly finds what is coded in the regular expression
  - Not robust to slight violations of rules constraints
- Unnormalized patterns:
  - Heuristics / Machine Learning
  - Prone to errors (false positives and false negatives, segmentation errors)
  - Better generalization capabilities
  - Rules based on capitalization, context (POST)

Part of Speech Tagging

- Aims at identifying the grammatical function of each token in sentences
  - Determiner
  - Nouns
  - Adjective
  - Verbs
  - Prepositions
  - ...
- POS Taggers
  - use
    - Linguistic resources
    - Heuristics rules (token endings / context)
  - can be learned through ML

Chunking / Parsing

- Chunking
  - Extract segments of sentences: e.g. Noun phrases
  - Classifier trained for begining / ending of chunks
- Parsing
  - Given a grammar
  - Determines the parsing tree of sentences

Knowledge extraction

- Exploits information provided by
  - Named entity extraction
  - Part of Speech Tagging
- in order to feed relational knowledge dataset
- (Subject) (Verb) (Object)
- Used in information monitoring (e.g. business intelligence)