Chap 2: Classical models for information retrieval

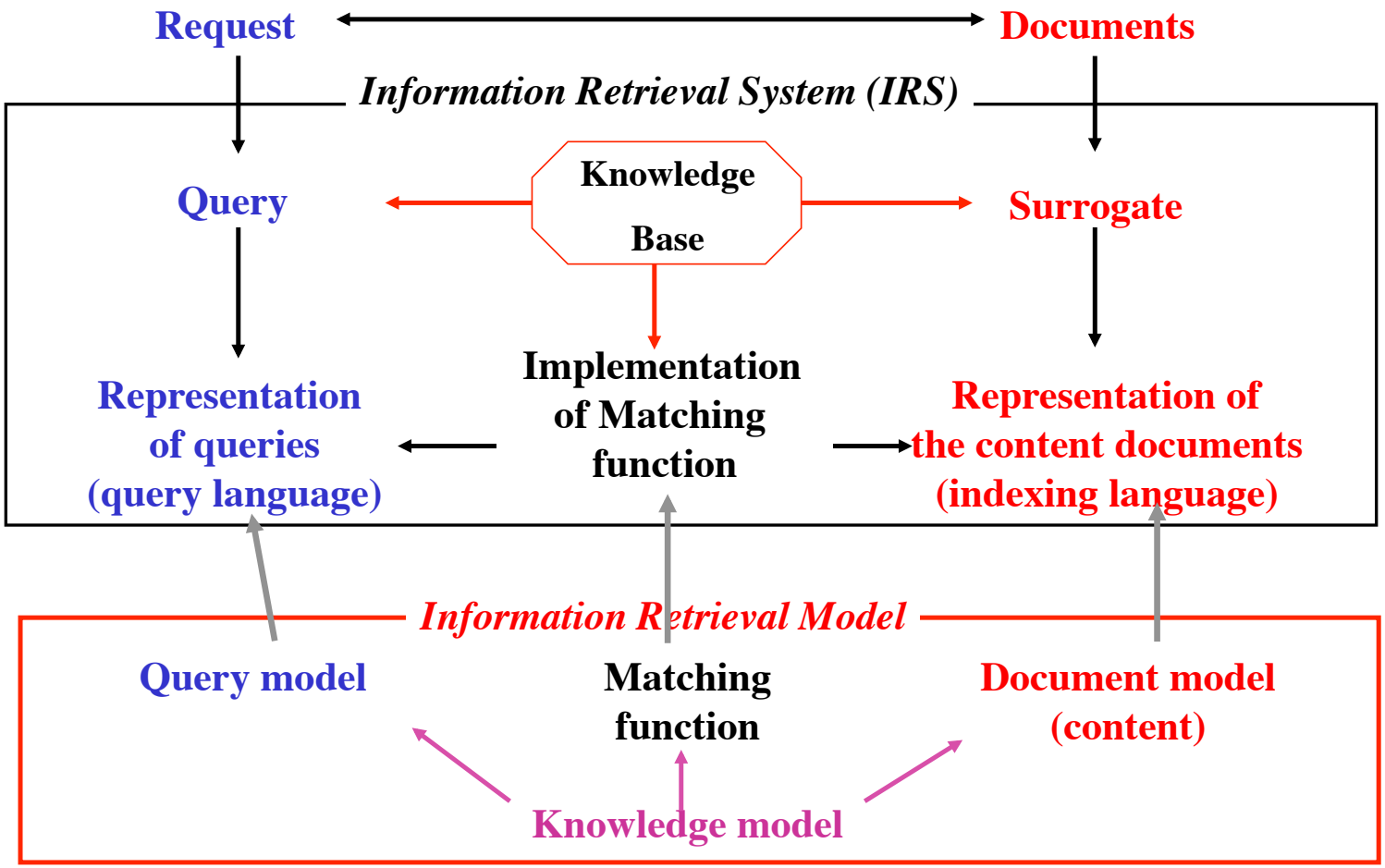
Jean-Pierre Chevallet & Philippe Mulhem

LIG-MRIM

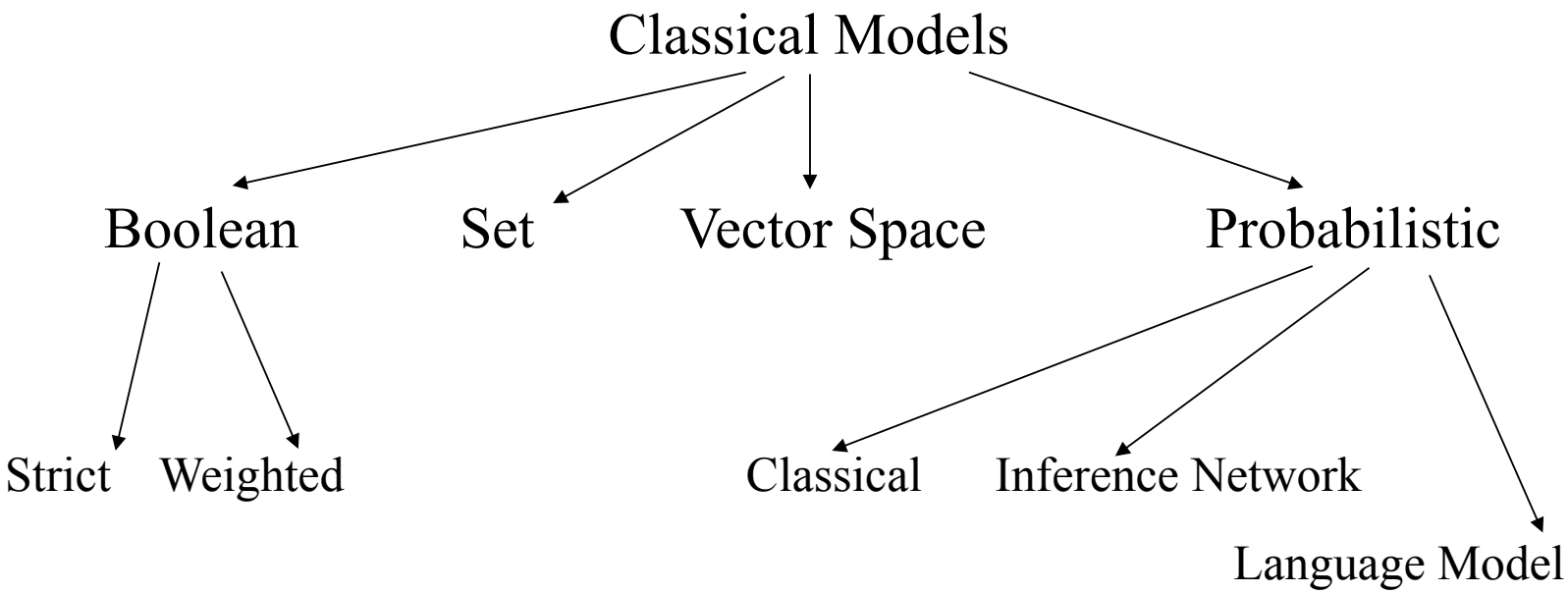
Sept 2016

Outline

IR System


Request
Documents
Information Retrieval System (IRS)
Query
Knowledge Base
Surrogate
Representation of queries (query language)
Implementation of Matching function
Representation of the content documents (indexing language)
Information Retrieval Model
Query model
Matching function
Document model (content)
Knowledge model

IR Models


Classical Models
Boolean
Set
Vector Space
Probabilistic
Strict
Weighted
Classical
Inference Network
Language Model

Outline

Set Model

Membership of a set

- Requests are single descriptors
- Indexing assignments : a set of descriptors
- In reply to a request, documents are either retrieved or not (no ordering)
- Retrieval rule : if the descriptor in the request is a member of the descriptors assigned to a document, then the document is retrieved.

This model is the simplest one and describes the retrieval characteristics of a typical library where books are retrieved by looking up a single author, title or subject descriptor in a catalog.

Example

Request

"information retrieval"

Doc1

{"information retrieval", "database" , "salton"}
$--> $ RETRIEVED $<$ $--$

Doc2

{ "database" , "SQL"}
$--> $ NOT RETRIEVED $<$ $--$

Set inclusion

- Request: a set of descriptors
- Indexing assignments : a set of descriptors
- Documents are either retrieved or not (no ordering)
- Retrieval rule : document is retrieved if ALL the descriptors in the request are in the indexing set of the document.

This model uses the notion of inclusion of the descriptor set of the request in the descriptor set of the document

Set intersection

- Request: a set of descriptors PLUS a cut off value
- Indexing assignments : a set of descriptors
- Documents are either retrieved or not (no ordering)
- Retrieval rule : document is retrieved if it shares a number of descriptors with the request that exceeds the cut-off value

This model uses the notion of set intersection between the descriptor set of the request with the descriptor set of the document.

Set intersection plus ranking

- Request: a set of descriptors PLUS a cut off value
- Indexing assignments : a set of descriptors
- Retrieved documents are ranked

Retrieval rule : documents showing with the request more than the specified number of descriptors are ranked in order of decreasing overlap

Outline

Logical Models

Based on a given formalized logic:

- Propositional Logic: boolean model
- First Order Logic : Conceptual Graph Matching
- Modal Logic
- Description Logic: matching with knowledge
- Concept Analysis
- Fuzzy logic
- ...

Matching : a deduction from the query Q to the document D.

Boolean Model

- Request is any boolean combination of descriptors using the operators AND, OR and NOT
- Indexing assignments : a set of descriptors
- Retrieved documents are retrieved or not

Retrieval rules:

- if Request $= t_a \wedge t_b$ then retrieve only documents with both t_a and t_b
- if Request $= t_a \vee t_b$ then retrieve only documents with either t_a or t_b
- if Request $= \neg t_a$ then retrieve only documents without t_a.

Boolean Model

Knowledge Model : $T = \{t_i\}, i \in [1, ..N]$

- Term t_i that index the documents

The document model (content) I a Boolean expression in the proposition logic, with the t_i considered as propositions:

- A document $D_1 = \{t_1, t_3\}$ is represented by the logic formula as a conjunction of all terms direct (in the set) or negated.
  $t_1 \wedge \neg t_2 \wedge t_3 \wedge \neg t_4 \wedge ... \wedge \neg t_{N-1} \wedge \neg t_N$
- A query Q is represented by any logic formula
- The matching function is the logical implication: $D \models Q$

Boolean Model: relevance value

No distinction between relevant documents:

- $Q = t_1 \wedge t_2$ over the vocabulary $\{t_1, t_2, t_3, t_4, t_5, t_6\}$
- $D_1 = \{t_1, t_2\} \equiv t_1 \wedge t_2 \wedge \neg t_3 \wedge \neg t_4 \wedge \neg t_5 \wedge \neg t_6$
- $D_2 = \{t_1, t_2, t_3, t_4, t_5\} \equiv t_1 \wedge t_2 \wedge t_3 \wedge t_4 \wedge t_5 \wedge \neg t_6$

Both documents are relevant because : $D_1 \supset Q$ and $D_2 \supset Q$.
We "feel" that D_1 is a better response because "closer" to the query.
Possible solution : $D \supset Q$ and $Q \supset D$. (See chapter on Logical Models)

Boolean Model: complexity of queries

- $Q = ((t_1 \wedge t_2) \vee t_3) \wedge (t_4 \vee \neg(\neg t_5 \wedge t_6))$

Meaning of the logical $\vee$ (inclusive) different from the usual "or" (exclusive)

Conclusion for boolean model

- Model use till the the 1990.
- Very simple to implement
- Still used is some web search engine as a basic document fast filter with an extra step for document ordering.

Outline

Beyond boolean logic: choice of a logic

A lot of possible choices to go beyond the boolean model. The choice of the underlying logic L to models the IR matching process:

- Propositional Logic
- First order logic
- Modal Logic
- Fuzzy Logic
- Description logic
- ....

Beyond boolean logic: matching interpretation

For a logic L, different possible interpretations of matching:

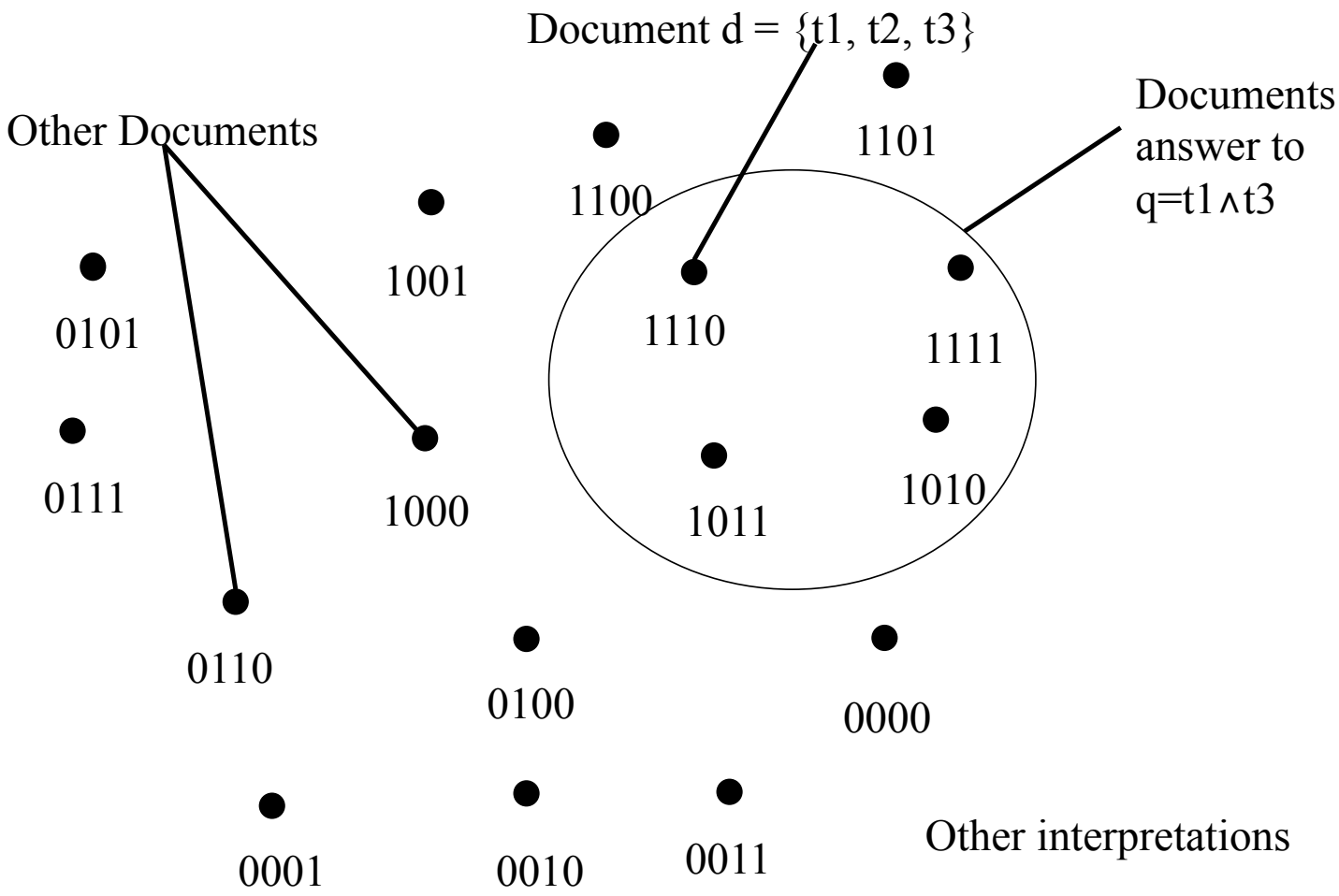
Deduction theory

- $D \vdash_L Q$
- $\vdash_L D \supset Q$

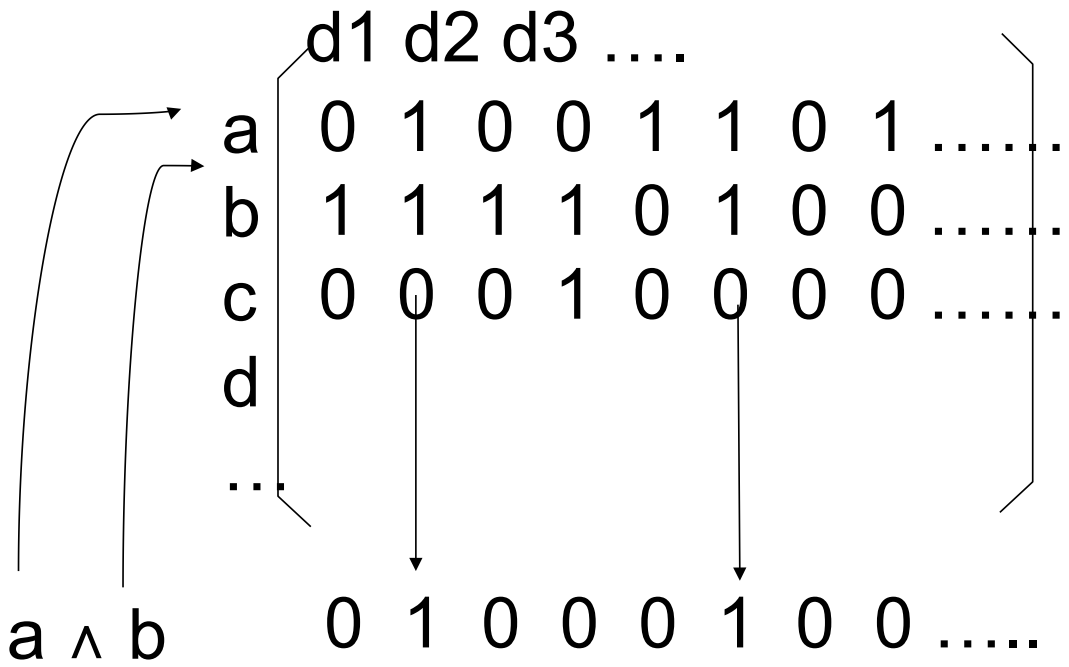
Model theory

- $D \models_L Q$
- $\models_L D \supset Q$

Interpretations



Inverted File



Implementing the boolean model

Posting list:

- List of *unique* doc id associates with an indexing term
- If no ordering, this list is in fact a set
- If ordering, nice algorithm to fuse lists

Main set (list) fusion algorithms:

- Intersection, usion and complement
- Algorithm depend on order.
- If no order complexity is $O(n * m)$
- if order, complexity is $O(n + m)$! So maintain posting list ordered.

Implementing the boolean model

Left to exercise : produces the 3 algorithms, of union, intersection and complement, with the following constraints:

- The two lists are read only
- The produced list is e new one and is write only
- Using sorting algorithms is not allowed

Deduce from the algorithms the compexity.

Outline

Weighted Boolean Model

Extension of Boolean Model with weights.
Weight denoting the representativity of a term for a document).
Knowledge Model : $T = \{t_i\}, i \in [1, ..N]$
Terms t_i that index the documents.

A document D is represented by

- A logical formula D (similar to Boolean Model)
- A function $W_D : T \rightarrow [0, 1]$, which gives, for each term in T the weight of the term in D. The weight is 0 for a term not present in the document.

Weighted Boolean Model: matching

Non binary matching function based on Fuzzy logic

- $RSV(D, a \vee b) = Max[W_D(a), W_D(b)]$
- $RSV(D, a \wedge b) = Min[W_D(a), W_D(b)]$
- $RSV(D, \neg a) = 1 - W_D(a)$
- Limitation : this matching does not take all the query terms into account.

Weighted Boolean Model: matching

Non binary matching function based on a similarity function which take more into account all the query terms.

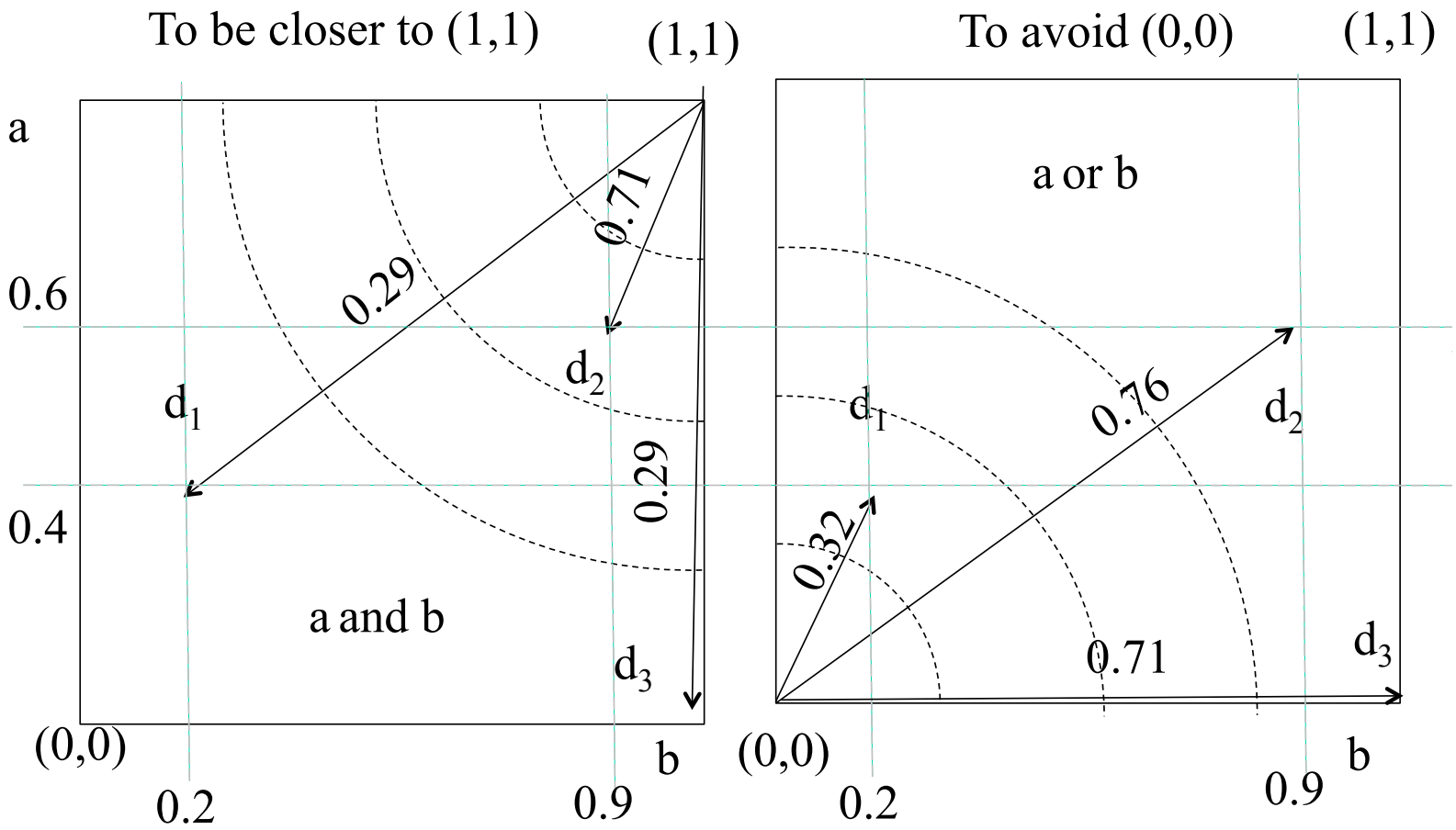
- $RSV(D, a \vee b) = \sqrt{\frac{W_D(a)^2 + W_D(b)^2}{2}}$
- $RSV(D, a \wedge b) = 1 - \sqrt{\frac{(1-W_D(a))^2 + (1-W_D(b))^2}{2}}$
- $RSV(D, \neg a) = 1 - W_D(a)$
- Limitation : query expression for complex needs

Weighted Boolean Model: matching example

			Boolean		Weighted Boolean	
Query / Document	a	b	a ∨ b	a ∧ b	a ∨ b	a ∧ b
D_1	1	1	1	1	1	1
D_2	1	0	1	0	$1/\sqrt{2}=0.71$	$1-1/\sqrt{2}=0.29$
D_3	0	1	1	0	$1/\sqrt{2}$	$1-1/\sqrt{2}$
D_4	0	0	0	0	0	0

Weighted Boolean Model: matching example

d1=(0.2, 0.4) , d2=(0.6, 0.9), d3=(0,1)



Outline

Upper bounds for results

Let t_i indexing term, let $\#t_i$ the size of its posting list and let $|Q|$ the number of documents that answer the boolean query Q.

1. Propose an upper bound value in term of posting list size for the queries $t_1 \vee t_2$ and $t_1 \wedge t_2$
2. If we suppose that $\#t_1 > \#t_2 > \#t_3$, compute the upper bound value of $|t_1 \wedge t_2 \wedge t_3|$, then propose a way to reduce global computation of posting list merge[1].
3. If we suppose that $\#t_1 > \#t_2 > \#t_3$, compute the upper bound value of $|t_1 \vee t_2|$ and of $|(t_1 \vee t_2) \wedge t_3|$, then propose a way to reduce global computation of posting list merge of query $(t_1 \vee t_2) \wedge t_3$

[1] Use the complexity value of merge depending of list sizes

Outline

Vector Space Model

- Request: a set of descriptors each of which has a positive number associated with it
- Indexing assignments : a set of descriptors each of which has a positive number associated with it.
- Retrieved documents are ranked

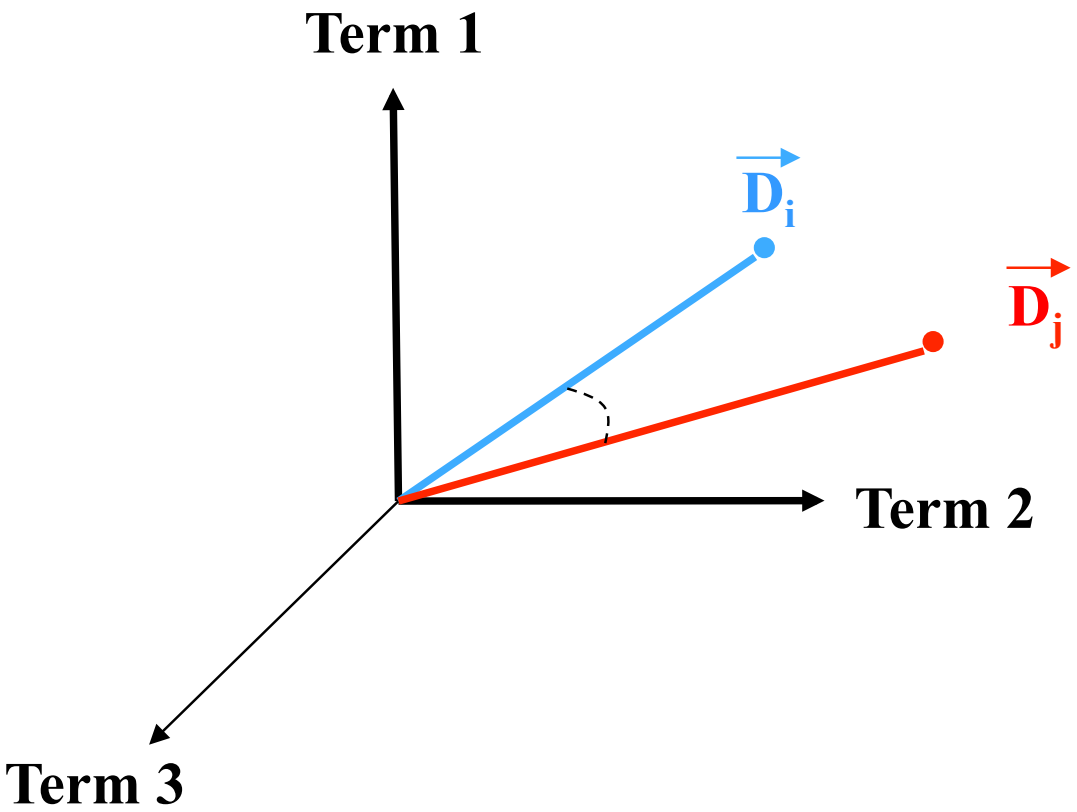
Retrieval rule : the weights of the descriptors common to the request and to indexing records are treated as vectors. The value of a retrieved document is the cosine of the angle between the document vector and the query vector.

Vector Space Model

Knowledge model: $T = \{t_i\}, i \in [1, .., n]$
All documents are described using this vocabulary.
A document D_i is represented by a vector d_i described in the R^n vector space defined on T
$d_i = (w_{i,1}, w_{i,2}, ..., w_{i,j}, ..., w_{i,n})$, with $w_{k,l}$ the weight of a term t_l for a document.
A query Q is represented by a vector q described in the same vector space $q = (w_{Q,1}, w_{Q,2}, ..., w_{Q,j}, ..., w_{Q,n})$

Vector Space Model

The more two vectors that represent documents are ?near?, the more the documents are similar:



Vector Space Model: matching

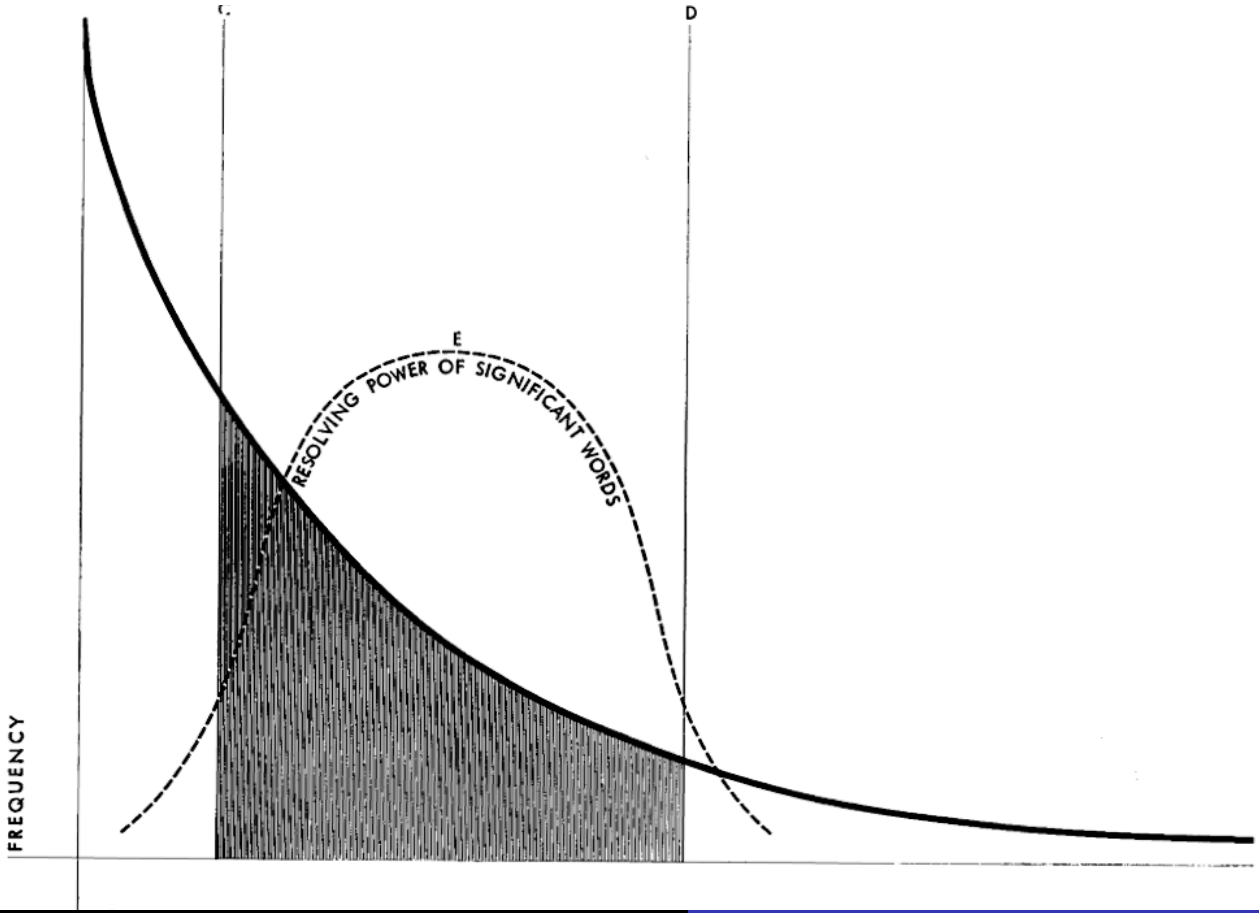
Relevance: is related to a vector similarity.
$RSV(D, Q) =_{def} SIM(\vec{D}, \vec{Q})$

- Symmetry: $SIM(\vec{D}, \vec{Q}) = SIM(\vec{Q}, \vec{D})$
- Normalization: $SIM : V \rightarrow [min, max]$
- Reflectivity : $SIM(\vec{X}, \vec{X}) = max$

Outline

Vector Space Model: weighting

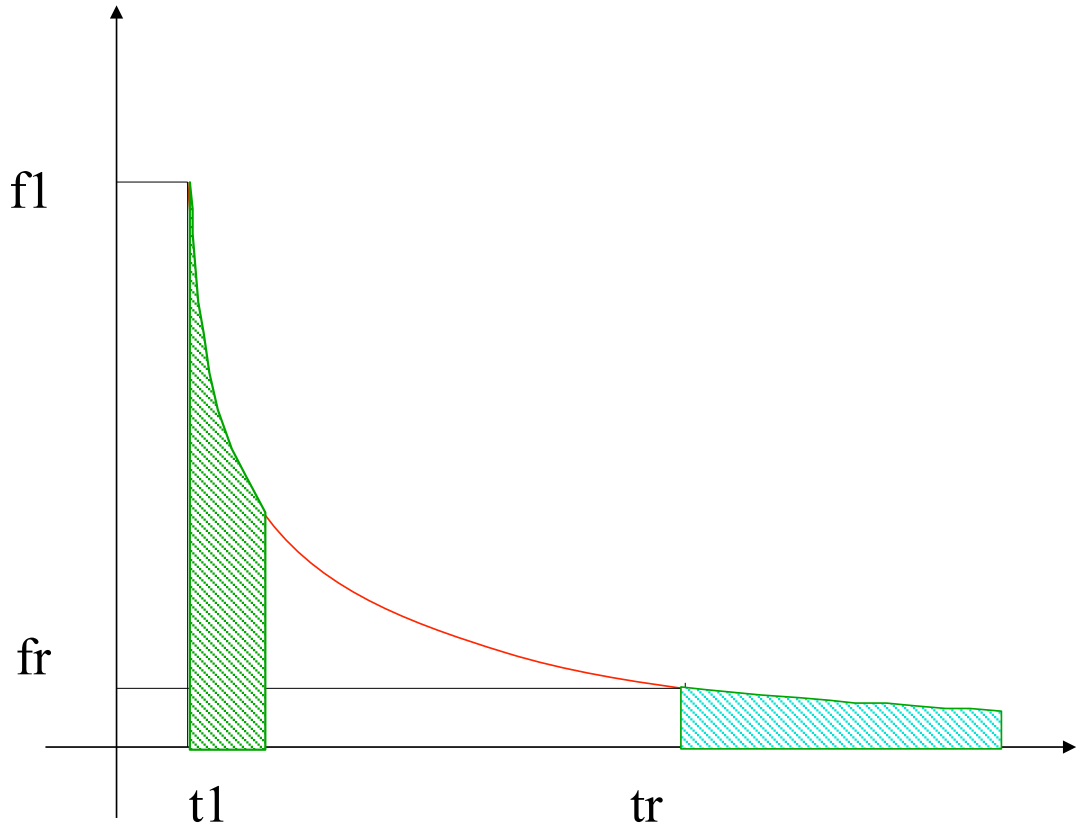
Based on counting the more frequent words, and also the more significant ones.



Zipf Law

Rank r and frequency f:

$$r \times f = const$$

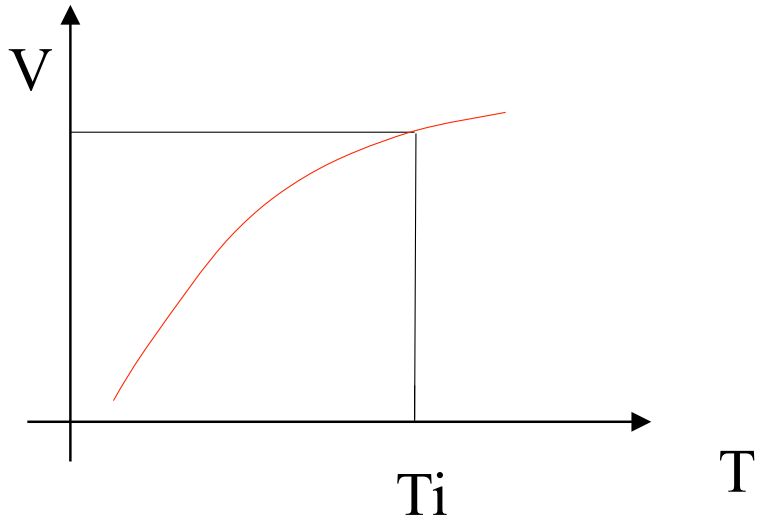


Heap Law

Estimation of the vocabulary size $|V|$ according to the size $|C|$ of the corpus:

$$|V| = K \times |C|^{\beta}$$

for English : $K \in [10, 100]$ and $\beta \in [0.4, 0.6]$ (e.g., $|V| \approx 39\ 000$ for $|C|$=600 000, K=50, β=0.5)

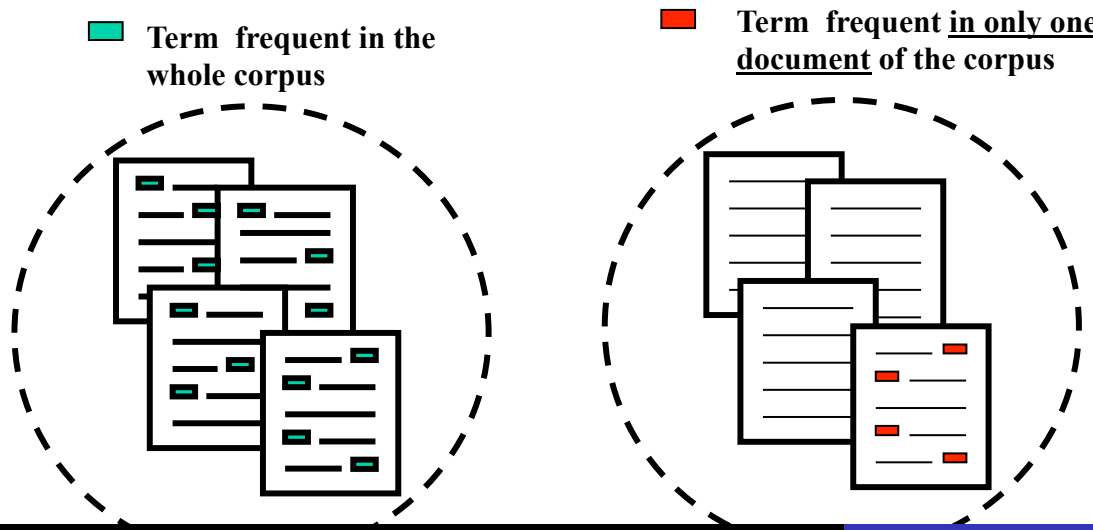


Term Frequency

Term Frequency

The frequency $tf_{i,j}$ of the term t_j in the document D_i equals to the number of occurrences of t_j in D_i.

Considering a whole corpus (document database) into account, a term that occurs a lot does not discriminate documents:



Document Frequency: tf.idf

Document Frequency

The document frequency df_j of a term t_j is the number of documents in which t_j occurs.

The larger the df, the worse the term for an IR point of view... so, we use very often the inverse document frequency idf_j:

Inverse Document Frequency

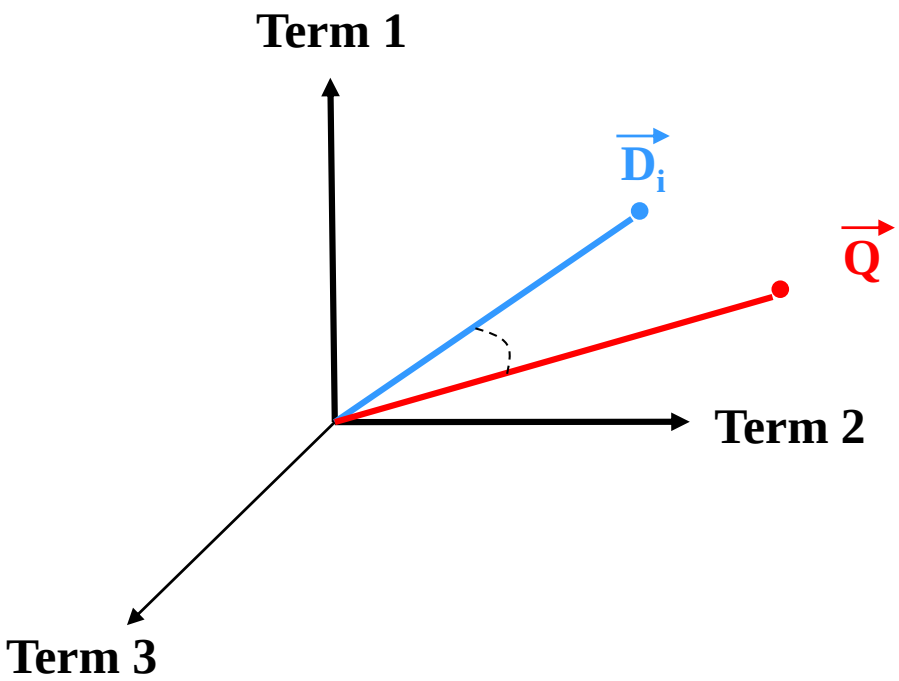
$idf_j = \frac{1}{df_j}$

$idf_j = \log(\frac{|C|}{df_j})$ with $|C|$ is the size of the corpus, i.e. the number of documents.

The classical combination : $w_{i,j} = tf_{i,j} \times idf_j$.

matching

Matching function: based on the angle between the query vector $\vec{Q}$ and the document vector $\vec{D_i}$.
The smaller the angle the more the document matches the query.



matching: cosine

One solution is to use the cosine the angle between the query vector and the document vector.

Cosine

$$SIM_{cos}(\vec{D_i}, \vec{Q}) = \frac{\sum_{k=1}^{n} w_{i,k} \times w_{Q,k}}{\sqrt{\sum_{k=1}^{n} (w_{i,k})^2 \times \sum_{k=1}^{n} (w_{Q,k})^2}}$$

matching: set interpretation

When binary discrete values, on have a set interpretations $D_i^{\{\}}$ and $Q^{\{\}}$:

$$SIM_{cos}(\vec{D_i}, \vec{Q}) = \frac{|D_i^{\{\}} \cap Q^{\{\}}|}{\sqrt{|D_i^{\{\}}| \times |Q^{\{\}}|}}$$

$SIM_{cos}(\vec{D_i}, \vec{Q}) = 0 : D_i^{\{\}}$ and $Q^{\{\}}$ are disjoined

$SIM_{cos}(\vec{D_i}, \vec{Q}) = 1 : D_i^{\{\}}$ and $Q^{\{\}}$ are equal

Other matching functions

Dice coefficient

$$SIM_{dice}(\vec{D_i}, \vec{Q}) = \frac{2\sum_{k=1}^{N} w_{i,k} \times w_{Q,k}}{\sum_{k=1}^{N} w_{i,k} + w_{Q,k}}$$

Discrete Dice coefficient

$$SIM_{dice}(\vec{D_i}, \vec{Q}) = \frac{2|D_i^{\{\}} \cap Q^{\{\}}|}{|D_i^{\{\}}| + |Q^{\{\}}|}$$

Outline

Matching cosine simplification

Question

Transform the cosine formula so that to simplify the matching formula.

Cosine

$$SIM_{cos}(\vec{D_i}, \vec{Q}) = \frac{\sum_{k=1}^{n} w_{i,k} \times w_{Q,k}}{\sqrt{\sum_{k=1}^{n} (w_{i,k})^2 \times \sum_{k=1}^{n} (w_{Q,k})^2}}$$

Exercice: Similarity et dissimilarity and distance

The Euclidian distance between point x and y is expressed by :

$$L_2(x, y) = \sum_i (x_i - y_i)^2$$

Question

Show the potential link between L_2 and the cosine.

Tips:

- consider points as vectors
- consider document vectors has normalized, i.e. with $\sum_i y_i^2$ as constant.

Exercice: dot product and list intersection

The index is usually very sparse: a (usefull) term appears in less than 1% of document.
A term t has a non null weight in $\vec{D}$ *iff* t appears in document D

Question

Show that the algorithm for computing the dot product is equivalent to list intersections.

Exercice: efficiency of list intersection

Let the two lists X and Y of size $|X|$ and $|Y|$:

Question

Compare the efficiency of a non sorted list intersection algorithm and a sorted list intesection.

Outline

Link between dot product and inverted files

If $RSV(x, y) \propto \sum_i x_i y_i$ after some normalizations of y, then :

$$RSV(x, y) \propto \sum_{i, x_i \neq 0, y_i \neq 0} x_i y_i$$

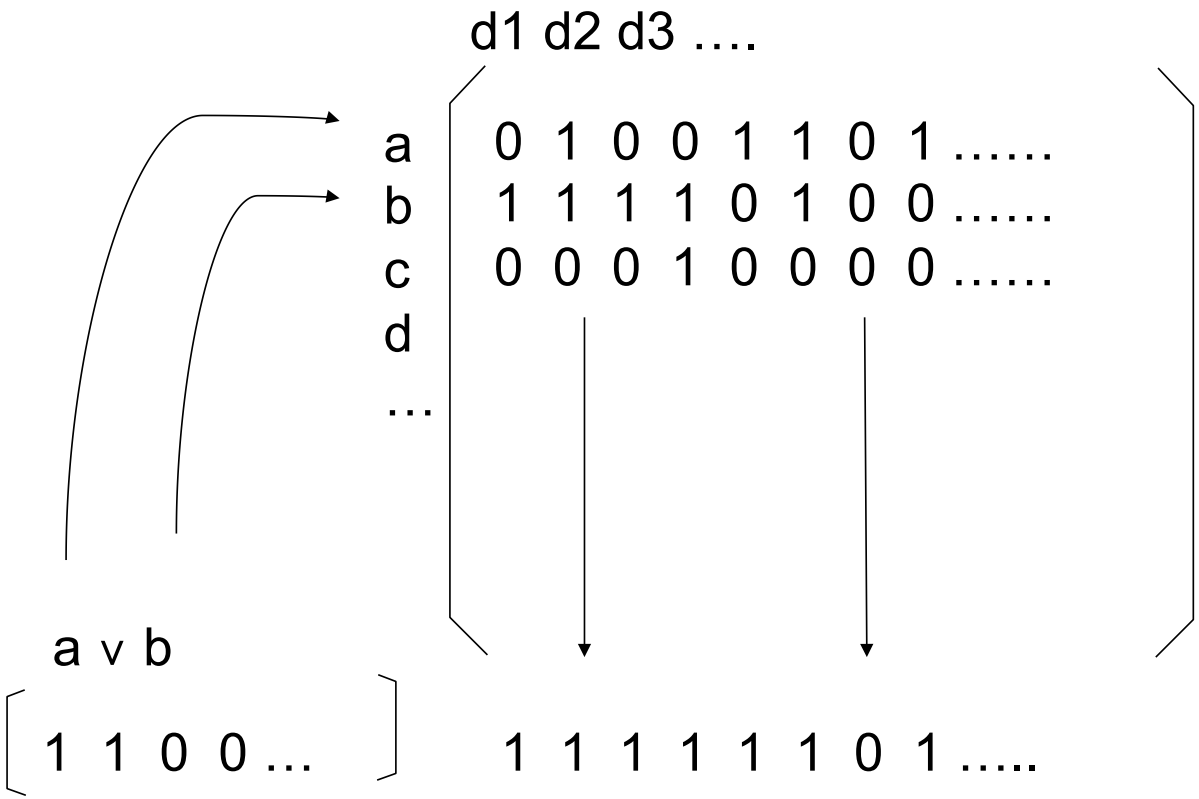
- Query: just to proceed with terms that are in the query, i.e. whose weight are not null.
- Documents: store only non null terms.
- Inverted file: access to non null document weight for for each term id.

The index

- Use a Inverted file like Boolean model
- Store term frequency in the posting list
- *Do not pre-compute* the weight, but keep raw integer values in the index
- Compute the matching value on the fly, i.e. during posting list intersection

Matching: matrix product

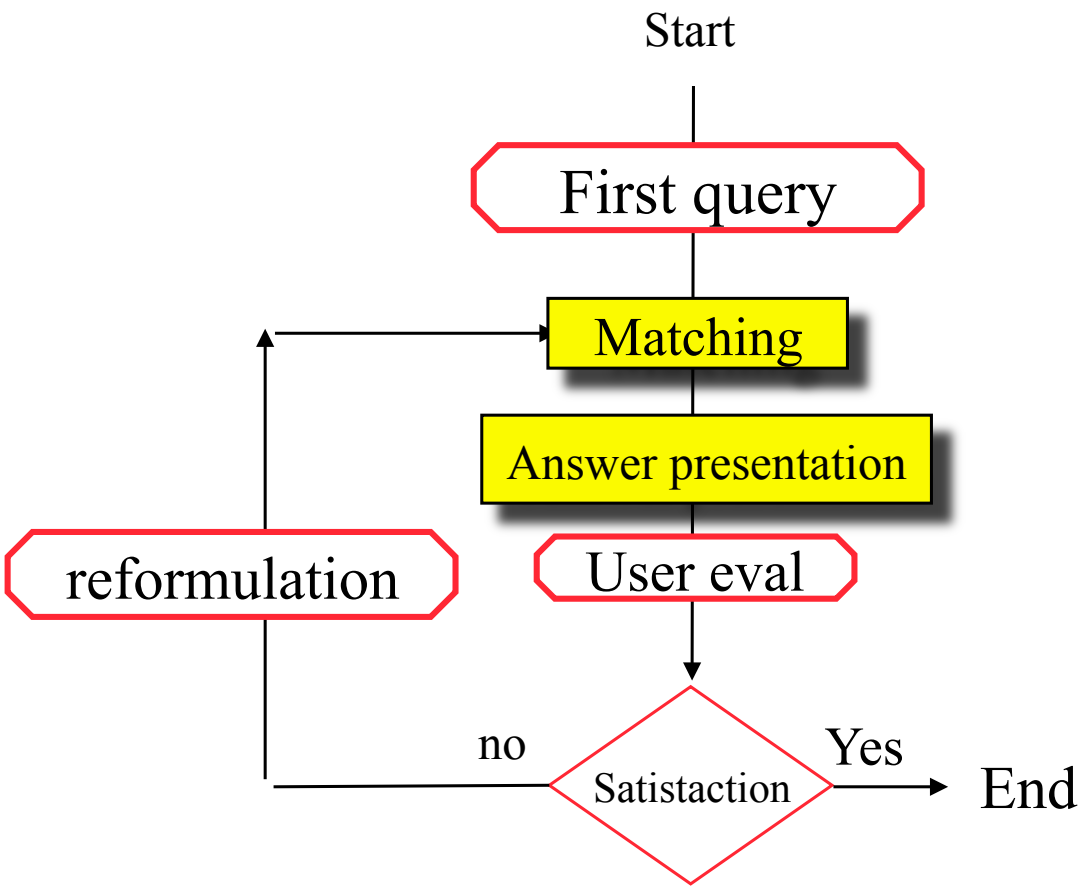
With an inverted file, in practice, matching computation is a matrix product:



Outline

Relevance feedback

To learn system relevance from user relevance:



Rocchio Formula

Rocchio Formula

$$\vec{Q}_{i+1} = \alpha \vec{Q}_i + \beta \vec{Rel}_i - \gamma n\vec{Rel}_i$$

With:

- $\vec{Rel}_i$: the cluster center of relevant documents, i.e., the positive feedback
- $n\vec{Rel}_i$: the cluster center of non relevant documents, i.e., the negative feedback

Note : when using this formula, the generated query vector may contain negative values.

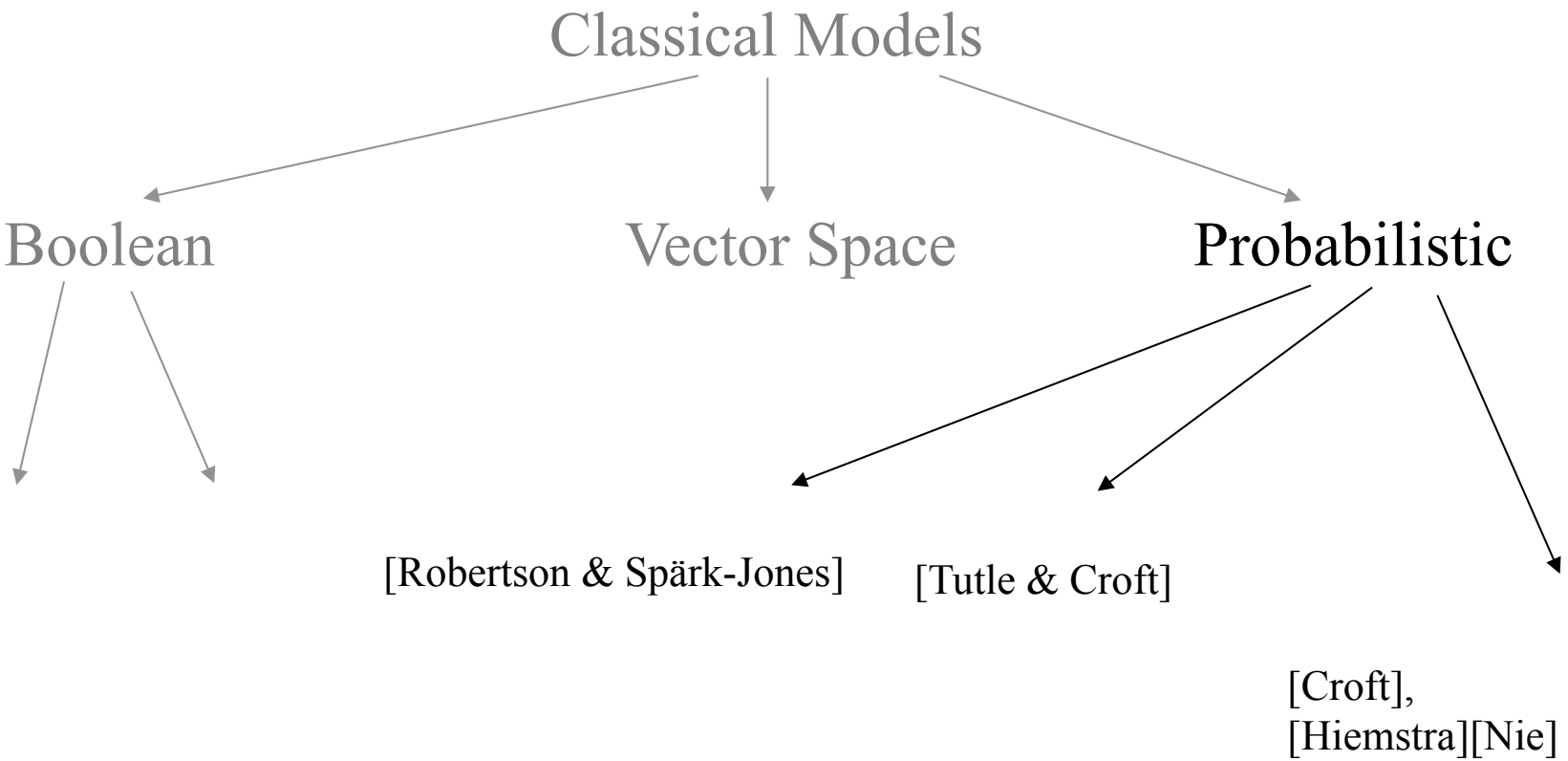
Outline

Probabilistic Models

Capture the IR problem in a probabilistic framework.

- first probabilistic model (Binary Independent Retrieval Model) by Robertson and Spark-Jones in 1976...
- late 90s, emergence of language models, still hot topic in IR
- Overall question: "what is the probability for a document to be relevant to a query ?" several interpretation of this sentence

Models


Classical Models
Boolean
Vector Space
Probabilistic
[Robertson & Spärk-Jones]
[Tutle & Croft]
[Croft],
[Hiemstra][Nie]

Probabilistic Model of IR

Different approaches of seeing a probabilistic approach for information retrieval

- Classical approach: probability to have the event Relevant knowing one document and one query.
- Inference Networks approach: probability that the query is true after inference from the content of a document.
- Language Models approach: probability that a query is generated from a document.

Binary Independent Retrieval Model

Robertson & Spark-Jones 1976

- computes the relevance of a document from the relevance known a priori from other documents.
- achieved by estimating of each indexing term a the document, and by using the Bayes Theorem and a decision rule.

Binary Independent Retrieval Model

R: binary random variable

- $R = r$: relevant; $R = \bar{r}$: non relevant
- $P(R = r|d, q)$: probability that R is relavant for the document and the query considered (noted $P(r|d, q)$)

Probability of relevance depends only from document and query

- term weights are binary: $d = (11\ldots100\ldots), w_t^d = 0$ or 1

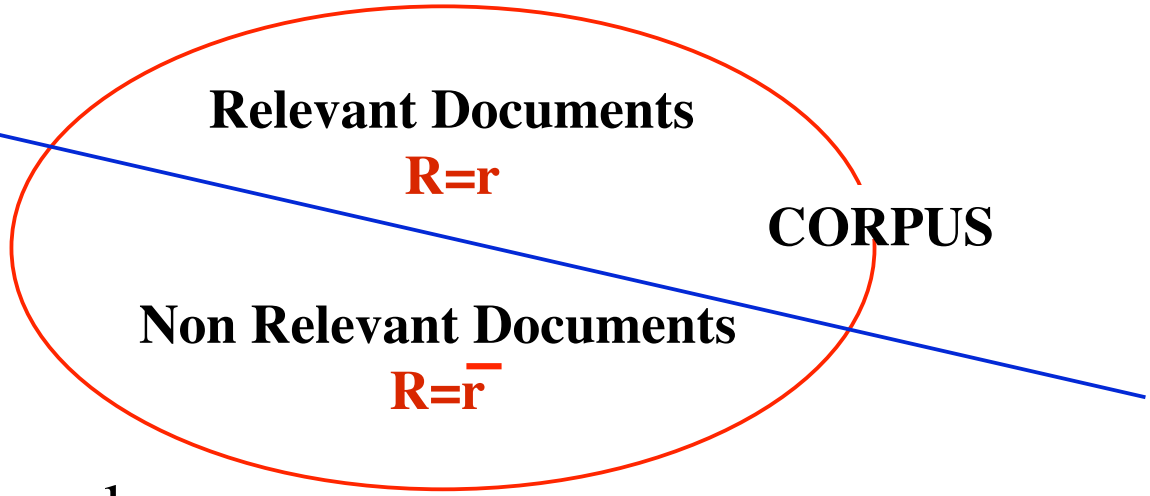
Each term t is characterized by a binary variable w_t, indicating the probability that the term occurs.

- $P(w_t = 1|q, r)$: probability that t occurs in a relevant document.
- $P(wt = 0|q, r) = 1–P(wt = 1|q, r)$

The terms are conditionnaly independant to R

Binary Independent Retrieval Model

For a query Q



with

$$\text{Corpus} = \text{rel} \cup \text{nonrel}$$

$$\text{Relevant Documents} \cap \text{Non Relevant Documents} = \varnothing$$

Binary Independent Retrieval Model

Matching function : use of Bayes theorem

Probability to obtain the description d from observed relevances

P(r,q) is the relevance probability. It is the chance of randomly taking one document from the corpus which is relevant for the query q

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

Probability that the document d belongs to the set of relevant documents of the query q.

probability that the document d is picked for q

BIR: Matching function

Decision: document retrieved if:

$$\frac{P(r|d,q)}{P(\bar{r}|d,q)} = \frac{P(d|r,q).P(r,q)}{P(d|\bar{r},q).P(\bar{r},q)} > 1$$

IR looks for a ranking, so we eliminate $\frac{P(r,q)}{P(\bar{r},q)}$ because it is constant for a given query

BIR: Matching function

In IR, it is more simple to use logs:

$$rsv(d) =_{rank} log(\frac{P(d|r,q)}{P(d|\bar{r},q)})$$

BIR: Matching function

Simplification using hypothesis of independence between terms (Binary Independence) with weight w for term t in d

$$P(d|r,q) = P(d = (10...110...)|r,q)$$
$$= \prod_{w_t^d=1} P(w_t^d = 1|r,q) \prod_{w_t^d=0} P(w_t^d = 0|r,q)$$

$$P(d|\bar{r},q) = P(d = (10...110...)|\bar{r},q)$$
$$= \prod_{w_t^d=1} P(w_t^d = 1|\bar{r},q) \prod_{w_t^d=0} P(w_t^d = 0|\bar{r},q)$$

BIR: Matching function

Let's simplify the notations:

$$P(w_t^d = 1|r, q) = p_t$$
$$P(w_t^d = 0|r, q) = 1 - p_t$$
$$P(w_t^d = 1|\bar{r}, q) = u_t$$
$$P(w_t^d = 0|\bar{r}, q) = 1 - u_t$$

Hence :

$$P(d|r, q) = \prod_{w_t^d=1} p_t \prod_{w_t^d=0} 1 - p_t$$
$$P(d|\bar{r}, q) = \prod_{w_t^d=1} u_t \prod_{w_t^d=0} 1 - u_t$$

BIR: Matching function

Now let's compute the Relevance Status Value:

$$rsv(d) =_{rank} log(\frac{P(d|r,q)}{P(d|\bar{r},q)}) = log(\frac{\prod_{w_t^d=1} p_t \prod_{w_t^d=0} 1-p_t}{\prod_{w_t^d=1} u_t \prod_{w_t^d=0} 1-u_t})$$

Conclusion

Common step for indexing:

- Define index set
- Automatize index construction from documents
- Select a model for index representation and weighting
- Define a matching process and an associated ranking

This is used for textual processing but also for other media.

Outline

Solution: matching cosine simplification

Because Q is constant, norm of vector Q is a constant that can be removed.

$$K = \frac{1}{\sum_{k=1}^{n} (w_{Q,k})^2}$$

$$SIM_{cos}(\vec{D_i}, \vec{Q}) = K \frac{\sum_{k=1}^{n} w_{i,k} \times w_{Q,k}}{\sqrt{\sum_{k=1}^{n} (w_{i,k})^2}}$$

Solution: matching cosine simplification

The division of the document vector by its norm can be precomputed:

$$w'_{i,k} = \frac{w_{i,k}}{\sqrt{\sum_{k=1}^{n} (w_{i,k})^2}}$$

Solution: matching cosine simplification

Hence, one juste has to compute a vector dot product:

$$SIM_{cos}(\vec{D_i}, \vec{Q}) = K \sum_{k=1}^{n} w'_{i,k} \times w_{Q,k}$$

The constant does not influence the order:

$$RSV_{cos}(\vec{D_i}, \vec{Q}) \propto \sum_{k=1}^{n} w'_{i,k} \times w_{Q,k}$$

Solution: Similarity et dissimilarity and distance

To transform a distance or dissimilarity into a similarity, simply negate the value. Ex. with the Squared Euclidian distance :

$$RSV_{D_2}(x,y) = -D_2(x,y)^2 = -\sum_i (x_i - y_i)^2$$

$$= -\sum_i (x_i^2 + y_i^2 - 2x_i y_i) = -\sum_i x_i^2 - \sum_i y_i^2 + 2\sum_i x_i y_i$$

if x is the query then $\sum_i x_i^2$ is constant for matching with a document set.

$$RSV(x,y)_{D_2} \propto -\sum_i y_i^2 + 2\sum_i x_i y_i$$

Solution: Similarity et dissimilarity and distance

$$RSV(x,y)_{D_2} \propto -\sum_i y_i^2 + 2\sum_i x_i y_i$$

If $\sum_i y_i^2$ is constant over the corpus then :

$$RSV(x,y)_{D_2} \propto 2\sum_i x_i y_i$$

$$RSV(x,y)_{D_2} \propto \sum_i x_i y_i$$

Hence, if we normalize the corpus so each document vector length is a constant, then using the Euclidean distance as a similarity, provides the same results than the cosine of the vector space model !

Solution: dot product and list intersection

In the dot product $\sum_i x_i y_i$, if the term at rank i is not in x or y then the term $x_i y_i$ is null and does not participate to the value.
Hence,
If we compute directly $\sum_i x_i y_i$ using a list of terms for each document, then we sum the $x_i y_i$ only for terms that are in the intersection.
Hence, it is equivalent to an intersection list.

Solution: efficiency of list intersection

If X and Y are not sorted, for each x_i one must find the term in Y by sequential search. So complexity is $\mathcal{O}(|X| \times |Y|)$
If X and Y are sorted, the algorithm used two pointer:

- Before the two pointer, lists are merged
- Lets compare the pointers :
  - If terms are equal : we can compute the merge (or the product), then move the tow pointers
  - It terms are different : because of the order, one are sure the smaller term to net be present in the other list. On ca move that pointeur.

Hence one move at least one pointer on each list, the the complexity is : $\mathcal{O}(|X| + |Y|)$