

CAI: Cerca i Anàlisi de la Informació
Grau en Ciència i Enginyeria de Dades, UPC

Two Information Retrieval Models

September 21, 2021

Slides by Marta Arias, José Luis Balcázar, Ramon Ferrer-i-Cancho, Ricard Gavalda, Department of Computer Science, UPC

Contents

Two Information Retrieval Models

- The Boolean Model of Information Retrieval
- About Phrase Queries

- The Vector Space Model of Information Retrieval
- Evaluation

- Relevance Feedback

A couple of IR models

- ▶ **Boolean model**,
 - ▶ Boolean queries, exact answers;
 - ▶ extension: phrase queries.
- ▶ **Vector model**,
 - ▶ weights on terms and documents;
 - ▶ similarity queries, approximate answers, ranking.

A couple of IR models

Both are **Bag of Words** models:

- ▶ Order of the words in the document is forgotten
- ▶ So document = set of pairs (word, frequency)

Boolean model additionally forgets about frequency

Example

A very simple toy case

Consider 7 documents with a vocabulary of 6 terms:

d1 = one three

d2 = two two three

d3 = one three four five five five

d4 = one two two two two three six six

d5 = three four four four six

d6 = three three three six six

d7 = four five

Example

Our documents in the Boolean model

	<i>five</i>	<i>four</i>	<i>one</i>	<i>six</i>	<i>three</i>	<i>two</i>
$d1 =$	[0	0	1	0	1	0]
$d2 =$	[0	0	0	0	1	2]
$d3 =$	[3	1	1	0	1	0]
$d4 =$	[0	0	1	2	1	4]
$d5 =$	[0	3	0	1	1	0]
$d6 =$	[0	0	0	2	3	0]
$d7 =$	[1	1	0	0	0	0]

Note that we sort the terms alphabetically. See why later.

Example

... in sparse representation

(Omitting the quotes around strings in your programming language...)

d1 = [(one,1), (three,1)]

d2 = [(three,1), (two,2)]

d3 = [(five,3), (four,1), (one,1), (three,1)]

d4 = [(one,1), (six,2), (three,1), (two,4)]

d5 = [(four,3), (six,1), (three,1)]

d6 = [(six,2), (three,3)]

d7 = [(five,1), (four,1)]

Boolean Model of Information Retrieval

Documents:

A document is completely identified by the **set of terms** that it contains.

Thus, for a set of terms $\mathcal{T} = \{t_1, \dots, t_T\}$, a document is just a **subset** of \mathcal{T} .

Each document can be seen as a **bit vector** of length T , $d = (d_1, \dots, d_T)$, where

- ▶ $d_i = 1$ if and only if t_i appears in d , or, equivalently,
- ▶ $d_i = 0$ if and only if t_i does not appear in d .

Queries in the Boolean Model

Boolean queries, exact answers

Atomic query:

a **single term**.

The answer is the set of documents that contain it.

Combining queries:

- ▶ OR, AND: operate as union or intersection of answers;
- ▶ Set difference, t_1 BUTNOT t_2 ($\equiv t_1$ AND NOT t_2);
- ▶ motivation: avoid unmanageably large answer sets.

Relevance in the Boolean Model

The set of documents that satisfy a query is defined recursively:

- ▶ Single-term queries define the set of docs that contain it.
- ▶ View AND, OR, BUTNOT as set operations, proceed recursively.

So: Relevance is **binary**: A document is either totally relevant or totally irrelevant.

No **ranking**.

This may be good or bad depending of user intent.

Example

Our documents in a Bag-of-Words model

	<i>five</i>	<i>four</i>	<i>one</i>	<i>six</i>	<i>three</i>	<i>two</i>
$d1 =$	[0	0	1	0	1	0]
$d2 =$	[0	0	0	0	1	1]
$d3 =$	[1	1	1	0	1	0]
$d4 =$	[0	0	1	1	1	1]
$d5 =$	[0	1	0	1	1	0]
$d6 =$	[0	0	0	1	1	0]
$d7 =$	[1	1	0	0	0	0]

Exercise

Invent some queries and give their answers

Phrase Queries, I

Slightly beyond the Boolean model

Phrase queries: conjunction plus adjacency

Ability to answer with the set of documents that have the terms of the query consecutively.

- ▶ A user querying “Keith Richards” may not wish a document that mentions both Keith Emerson and Emil Richards.
- ▶ Requires extending the notion of “basic query” to include adjacency.

Phrase Queries, II

Options to “hack them in”

Options:

- ▶ Run as conjunctive query, then doublecheck the whole answer set to filter out nonadjacency cases.
Slow if many “false positives”.
- ▶ Keep in the index dedicated information about **adjacency** of any two terms in a document (e.g. positions).
- ▶ Keep in the index dedicated information about a choice of “interesting pairs” of words.

Vector Space Model of Information Retrieval

Basis of all successful approaches

- ▶ Order of words still irrelevant.
- ▶ Frequency is relevant.
- ▶ Not all words are equally important.
- ▶ For a set of terms $\mathcal{T} = \{t_1, \dots, t_T\}$, a document is a vector $d = (w_1, \dots, w_T)$ of **floats** instead of **bits**.
- ▶ w_i is the **weight** of t_i in d .

Vector Space Model of Information Retrieval

Moving to vector space

- ▶ A document is now a vector in \mathbb{R}^T .
- ▶ The document collection **conceptually** becomes a **matrix**
terms \times documents.
but we never compute the matrix explicitly.
- ▶ Queries may also be seen as vectors in \mathbb{R}^T .

The tf-idf scheme

A way to assign weight vectors to documents

Two principles:

- ▶ The more frequent t is in d , the higher weight it should have.
- ▶ The more frequent t is in **the whole collection**, the less it discriminates among documents, so the lower its weight should be in all documents.

The tf-idf scheme, II

The formula

A document is a vector of weights

$$d = [w_{d,1}, \dots, w_{d,i}, \dots, w_{d,T}].$$

Each weight is a product of two terms

$$w_{d,i} = tf_{d,i} \cdot idf_i.$$

The term frequency term tf is

$$tf_{d,i} = \frac{f_{d,i}}{\max_j f_{d,j}}, \quad \text{where } f_{d,j} \text{ is the frequency of } t_j \text{ in } d.$$

And the inverse document frequency idf is

$$idf_i = \log_2 \frac{D}{df_i}, \quad \text{where } D = \text{number of documents}$$

and df_i = number of documents that contain term t_i .

Example, I

	<i>five</i>	<i>four</i>	<i>one</i>	<i>six</i>	<i>three</i>	<i>two</i>	maxf
$d1 =$	[0	0	1	0	1	0] 1
$d2 =$	[0	0	0	0	1	2] 2
$d3 =$	[3	1	1	0	1	0] 3
$d4 =$	[0	0	1	2	1	4] 4
$d5 =$	[0	3	0	1	1	0] 3
$d6 =$	[0	0	0	2	3	0] 3
$d7 =$	[1	1	0	0	0	0] 1
df =	2	3	3	3	6	2	

Example, II

$$\begin{aligned} \mathbf{df} &= \begin{bmatrix} 2 & 3 & 3 & 3 & 6 & 2 \end{bmatrix} \\ d3 &= \begin{bmatrix} 3 & 1 & 1 & 0 & 1 & 0 \end{bmatrix} \\ &\rightarrow \\ d3 &= \begin{bmatrix} \frac{3}{3} \log_2 \frac{7}{2} & \frac{1}{3} \log_2 \frac{7}{3} & \frac{1}{3} \log_2 \frac{7}{3} & \frac{0}{3} \log_2 \frac{7}{3} & \frac{1}{3} \log_2 \frac{7}{6} & \frac{0}{3} \log_2 \frac{7}{2} \end{bmatrix} \\ &= \begin{bmatrix} 1.81 & 0.41 & 0.41 & 0 & 0.07 & 0 \end{bmatrix} \\ d4 &= \begin{bmatrix} 0 & 0 & 1 & 2 & 1 & 4 \end{bmatrix} \\ &\rightarrow \\ d4 &= \begin{bmatrix} \frac{0}{4} \log_2 \frac{7}{2} & \frac{0}{4} \log_2 \frac{7}{3} & \frac{1}{4} \log_2 \frac{7}{3} & \frac{2}{4} \log_2 \frac{7}{3} & \frac{1}{4} \log_2 \frac{7}{6} & \frac{4}{4} \log_2 \frac{7}{2} \end{bmatrix} \\ &= \begin{bmatrix} 0 & 0 & 0.31 & 0.61 & 0.06 & 1.81 \end{bmatrix} \end{aligned}$$

Tweaking tf-idf

Many variations possible. Just two:

- ▶ Boost the $tf_{d,i}$ term if i appears in the title, or in boldface, or in the metadata of d .
- ▶ Laplace correction: Define $idf_i = \log \frac{D+1}{df_i+1}$. Allows to define it for terms i not in the index.

With a generating mindframe, accounts for terms so infrequent in the document source that did not appear in any of the D sampled documents.

Similarity of Documents in the Vector Space Model

The cosine similarity measure

- ▶ “Similar vectors” may happen to have very different sizes.
- ▶ We better compare only **their directions**.
- ▶ Equivalently, we **normalize** them before comparing them to have the same Euclidean length.

$$\text{sim}(d1, d2) = \frac{d1 \cdot d2}{|d1| |d2|} = \frac{d1}{|d1|} \cdot \frac{d2}{|d2|}$$

where

$$v \cdot w = \sum_i v_i \cdot w_i, \text{ and } |v| = \sqrt{v \cdot v} = \sqrt{\sum_i v_i^2}.$$

- ▶ Our weights are all nonnegative.
- ▶ Therefore, all cosines / similarities are between 0 and 1.

Cosine similarity, Example

$$\begin{aligned}d3 &= \begin{bmatrix} 1.81 & 0.41 & 0.41 & 0 & 0.07 & 0 \end{bmatrix} \\d4 &= \begin{bmatrix} 0 & 0 & 0.31 & 0.61 & 0.06 & 1.81 \end{bmatrix}\end{aligned}$$

Then

$$|d3| = 1.898, \quad |d4| = 1.933, \quad d3 \cdot d4 = 0.13$$

and $\text{sim}(d3, d4) = 0.035$ (i.e., small similarity).

Query Answering

- ▶ Queries can be transformed to vectors too.
- ▶ Sometimes, tf-idf weights; often, binary weights.
- ▶ $\text{sim}(\text{doc}, \text{query}) \in [0, 1]$.
- ▶ Answer: List of documents sorted by decreasing similarity.
- ▶ Note that $\text{sim}(d1, d2)$ make sense too.

Evaluation of Information Retrieval Usage

Start in the Boolean model. Notation:

\mathcal{D} : set of **all our documents**;

\mathcal{A} : **answer set**: documents that the system retrieves as answer;

\mathcal{R} : **relevant documents**: those that the user actually wishes to see as answer.

(But **no one** knows this set, not even the user!)

The Recall and Precision measures

Let's settle for:

- ▶ **recall**, $\frac{|\mathcal{R} \cap \mathcal{A}|}{|\mathcal{R}|}$:

$$Pr(d \in \mathcal{A} | d \in \mathcal{R})$$

- ▶ **precision**, $\frac{|\mathcal{R} \cap \mathcal{A}|}{|\mathcal{A}|}$:

$$Pr(d \in \mathcal{R} | d \in \mathcal{A})$$

Difficult to get both high recall and high precision.

Recall and Precision, confusion matrix

Equivalent definition

Confusion matrix

		<i>Answered</i>	
		relevant	not relevant
<i>Reality</i>	relevant	tp	fn
	not relevant	fp	tn

- ▶ $|\mathcal{R}| = tp + fn$
- ▶ $|\mathcal{A}| = tp + fp$
- ▶ $|\mathcal{R} \cap \mathcal{A}| = tp$
- ▶ $\text{Recall} = \frac{|\mathcal{R} \cap \mathcal{A}|}{|\mathcal{R}|} = \frac{tp}{tp + fn}$
- ▶ $\text{Precision} = \frac{|\mathcal{R} \cap \mathcal{A}|}{|\mathcal{A}|} = \frac{tp}{tp + fp}$

In information retrieval, usually tn is huge.

Accuracy in the Machine Learning sense is not interesting (99.9%).

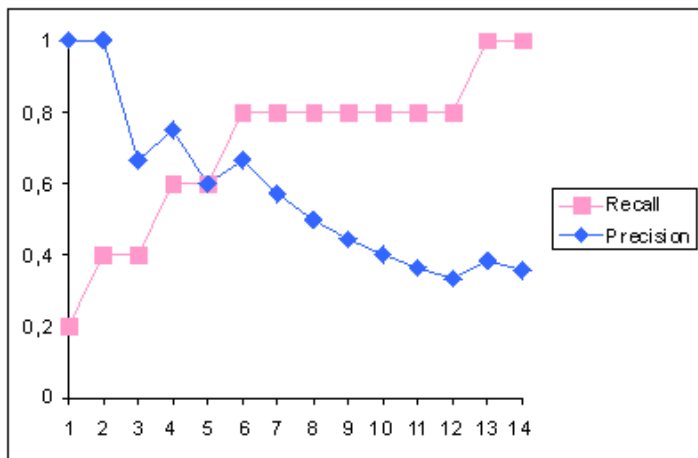
How many documents to show?

Now think of the Vector Model. Relevance is a real number.

- ▶ Users won't read too large answers.
- ▶ Long answers are likely to exhibit **low precision**.
- ▶ Short answers are likely to exhibit **low recall**.

We analyze precision and recall as functions of the number of documents k provided as answer.

Rank-recall and rank-precision plots

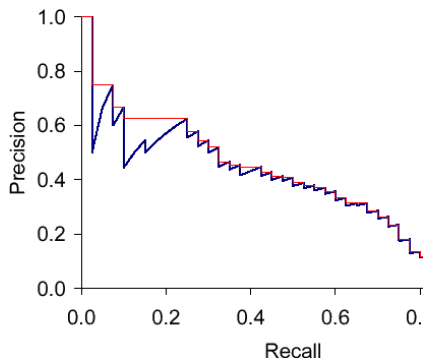


(Source: Prof. J. J. Paijmans, Tilburg)

A single “precision and recall” curve

x -axis for recall, and y -axis for precision.

(Similar to, and related to, the ROC curve in predictive models.)



(Source: Stanford NLP group)

Often: Plot 11 points of interpolated precision, at 0%, 10%, 20%, ..., 100% recall

Other measures of effectiveness

- ▶ AUC: Area under the curve of the plots above, relative to best possible

- ▶ F-measure:
$$\frac{2}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}}$$

- ▶ Harmonic mean. Closer to $\min(.,.)$ than arithmetic mean

- ▶ α -F-measure:
$$\frac{1}{\frac{\alpha}{\text{recall}} + \frac{1-\alpha}{\text{precision}}}$$

(F-measure is α -F-measure for $\alpha = 0.5$. Check!)

(other α 's give different importance to recall and precision)

Other measures of effectiveness, II

Take into account *the documents previously known to the user*.

- ▶ Coverage:

$$|\text{relevant \& known \& retrieved}| / |\text{relevant \& known}|$$

- ▶ Novelty:

$$|\text{relevant \& retrieved \& UNknown}| / |\text{relevant \& retrieved}|$$

Relevance Feedback

Going beyond what the user asked for

The user relevance cycle:

1. Get a query q
2. Retrieve relevant documents for q
3. Show top k to user
4. Ask user to mark them as relevant / irrelevant
5. Use answers to **refine** q
6. If desired, go to 2

Rocchio's rule

One way to create the new query

Given a query q , and a set of documents, **split** into relevant R and nonrelevant NR sets, build a new query q' :

$$q' = \alpha \cdot q + \beta \cdot \frac{1}{|R|} \cdot \sum_{d \in R} d - \gamma \cdot \frac{1}{|NR|} \cdot \sum_{d \in NR} d$$

- ▶ All vectors q and d 's must be **normalized** (e.g., unit length).
- ▶ Weights α, β, γ , scalars, with $\alpha > \beta > \gamma \geq 0$; often $\gamma = 0$.
 - α : degree of trust on the original user's query,
 - β : weight of positive information (terms that do not appear on the query but do appear in relevant documents),
 - γ : weight of negative information.

Relevance Feedback, III

In practice, often:

- ▶ good improvement of the **recall** for first round,
- ▶ marginal for second round,
- ▶ almost none beyond.

In web search, **precision** matters much more than **recall**, so the extra computation time and user patience may not be productive.

Relevance Feedback, IV

... as Query Expansion

It is a form of **Query Expansion**:

The new query has non-zero weights on words
that were not in the original query

Pseudorelevance feedback

Do not ask anything from the user!

- ▶ User patience is **precious** resource. They'll just walk away.
- ▶ Assume you did great in answering the query!
- ▶ That is, top- k documents in the answer are all relevant
- ▶ No interaction with user
- ▶ But don't forget that the search will feel slower.
- ▶ Stop, at the latest, when you get the same top k documents.

Pseudorelevance feedback, II

Alternative sources of feedback / query refinement:

- ▶ Links clicked / not clicked on.
- ▶ Think time / time spent looking at item.
- ▶ User's previous history.
- ▶ Other users' preferences!
- ▶ Co-occurring words: Add words that often occur with words in the query - for query expansion.