Information Retrieval Models

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

Grau en Enginyeria Informàtica, UPC

September 14, 2025

Why IR Models?

- Information Retrieval (IR) underpins search engines, digital libraries, recommender systems, etc.
- Goal: **help users find relevant information** within massive document collections.
- Challenges:
  - How to represent documents and queries?
  - How to measure relevance effectively?
  - How to evaluate performance and improve results?

Contents

What is an Information Retrieval Model?

- A proposal for a logical view of **documents**:
  - What information is stored or indexed about each document?
- A **query language**:
  - What kinds of queries are allowed?
- A notion of **relevance**:
  - How to decide whether each document matches a query?

Two IR models

- **Boolean model**
  - Boolean queries, exact answers
  - extension: phrase queries
- **Vector model**
  - Weights on terms and documents
  - similarity queries, aproximate answers, ranking

Both are *bag-of-words* approaches: order ignored

Vector model keeps term frequencies; Boolean model forgets frequency (only presence/absence)

Example: Simple toy collection

Consider 7 documents with 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
```

We'll reuse this example for Boolean and Vector slides.

Boolean Model — representation

- A document is identified by the **set of terms it contains**.
- For vocabulary $T = \{t_1, \ldots, t_T\}$, a document is a subset of T.
- Equivalently: a bit vector $d = (d_1, \ldots, d_T)$ where
  - $d_i = 1$ iff t_i appears in d.

Boolean queries — operators

- Atomic query: a single term $\rightarrow$ answer $=$ documents that contain it.
- Combine with Boolean operators:
  - `OR`, `AND` $\rightarrow$ union / intersection of answer sets.
  - `BUTNOT` / `AND NOT` $\rightarrow$ set difference.
- No ranking: relevance is binary (match / no match).

Boolean model — logic analogy

- Terms act as propositional variables.
- Documents act as propositional models (truth assignments).
- A document satisfies a query if, as a model, it satisfies the propositional formula.
- Note: global negation is typically avoided in practical IR query syntaxes.

Toy example - binary matrix

	five	four	one	six	three	two
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 queries and compute answers.)

Phrase Queries — motivation

- Phrase queries require **consecutive** occurrence of terms (adjacency), e.g. `"Keith Richards"`.
- Simple conjunction would return any doc containing both `Keith` and `Richards`, possibly far apart — not always desirable.

Phrase Queries — implementation options

1. Run conjunctive query, then filter by checking positions (slow if many false positives).
2. Store positional information in the index (common approach).
3. Store selected “interesting pairs” or n-grams in the index.

Vector Space Model — overview

- A document is a vector of weights $d = (w_1, \ldots, w_T)$ in $\mathbb{R}^T$.
- Weight w_i represents importance of term t_i in the document.
- Order ignored; frequency matters; not all words equally important.

Conceptual matrix view

- The collection is a (conceptual) terms $\times$ documents matrix.
- We typically use sparse representations and inverted indexes — we do **not** compute the dense matrix explicitly.
- Queries can also be represented as vectors.

tf–idf: principles

Two guiding principles:

1. The more frequent a term t is in document d, the higher its weight in d.
2. The more frequent t is across the whole collection, the less discriminative it is $\rightarrow$ lower weight.

Combine term-frequency (tf) and inverse document frequency (idf).

tf–idf: formula

For document d and term t_i:

$$w_{d,i} = \mathrm{tf}_{d,i} \times \mathrm{idf}_i$$

Term frequency (normalized):

$$\mathrm{tf}_{d,i} = \frac{f_{d,i}}{\max_j f_{d,j}}$$

Inverse document frequency:

$$\mathrm{idf}_i = \log_2\left(\frac{D}{\mathrm{df}_i}\right)$$

where D = total number of documents, and df_i = number of documents containing t_i.

Toy example — frequency matrix

Columns: five, four, one, six, three, two, and maxf

```
d1 = [ 0 0 1 0 1 0 ]   maxf=1
d2 = [ 0 0 0 0 1 2 ]   maxf=2
d3 = [ 3 1 1 0 1 0 ]   maxf=3
d4 = [ 0 0 1 2 1 4 ]   maxf=4
d5 = [ 0 3 0 1 1 0 ]   maxf=3
d6 = [ 0 0 0 2 3 0 ]   maxf=3
d7 = [ 1 1 0 0 0 0 ]   maxf=1
```

Document frequencies: `df = [ 2 3 3 3 6 2 ]`.

tf–idf: worked example (d3)

For $d_3 = [3, 1, 1, 0, 1, 0]$ with $\max f = 3$ and $D = 7$:

$$\text{tf}_3 = \left[\tfrac{3}{3}, \tfrac{1}{3}, \tfrac{1}{3}, 0, \tfrac{1}{3}, 0\right]$$

$$\text{idf} = \left[\log_2 \frac{7}{2}, \log_2 \frac{7}{3}, \log_2 \frac{7}{3}, \log_2 \frac{7}{3}, \log_2 \frac{7}{6}, \log_2 \frac{7}{2}\right]$$

Multiplying gives (approx):

$$d_3 \approx [1.81,\ 0.41,\ 0.41,\ 0,\ 0.07,\ 0]$$

tf-idf extensions in Information Retrieval

Extension	Formula / Concept	Key Idea / Intuition
Raw tf-idf	$w_{d,i} = tf_{d,i} \cdot \log \frac{D}{df_i}$	Basic weighting: importance proportional to term frequency and inverse document frequency.
Log-scaled tf	$tf_{d,i} = 1 + \log(f_{d,i})$	Reduces effect of very frequent terms.
Smooth idf	$idf_i = \log \frac{D}{1+df_i} + 1$	Avoids division by zero; moderates weight of rare terms.
Probabilistic idf	$idf_i = \log \frac{D-df_i}{df_i}$	Reflects odds of term appearing in relevant vs. non-relevant docs.

tf-idf extensions in Information Retrieval, cont.

Extension	Formula / Concept	Key Idea / Intuition
Query Expansion	Add related terms or adjust weights based on relevance feedback	Improves matching by considering synonyms or user context.
Document-level Weighting	Boost terms from titles, headings, abstracts	Prioritizes more informative sections of a document.

Your task: look up BM25 ranking function

Document similarity — cosine

- To compare documents we look at cosine similarity of vectors:

$$\mathrm{sim}(d_1, d_2) = \frac{d_1 \cdot d_2}{\|d_1\|\|d_2\|}$$

- **Normalize vectors** to unit length to compare direction only.
  - $\|\mathbf{v}\| = \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2}$
- With nonnegative weights, similarities are in $[0, 1]$.

Cosine example

```
d3 = [1.81, 0.41, 0.41, 0, 0.07, 0]
d4 = [0, 0, 0.31, 0.61, 0.06, 1.81]
```

$\|d_3\| \approx 1.898, \|d_4\| \approx 1.933, d_3 \cdot d_4 = 0.13$

$$\text{sim}(d_3, d_4) \approx 0.035$$

Boolean vs Vector: Pros & Cons

Boolean model

- Pros: simple, exact, predictable answers.
- Cons: no ranking, brittle queries, low recall if query is too strict.

Vector model

- Pros: ranked results, partial matches allowed, robust.
- Cons: weights heuristic, less interpretable, ignores word order/semantics.

Query answering in vector model

- Queries as vectors (binary tf or tf–idf).
- Compute $\mathrm{sim}(\mathrm{doc}, \mathrm{query}) \in [0, 1]$.
- Return documents sorted by decreasing similarity (ranked list).

Evaluation — basic definitions

- D: set of all documents.
- A: answer set (retrieved documents).
- R: relevant documents (ground truth, usually unknown).

Key measures:

$$\text{Recall} = \frac{|R \cap A|}{|R|}, \quad \text{Precision} = \frac{|R \cap A|}{|A|}$$

Intuitively,

- *precision*: how many selected items are relevant?
- *recall*: how many relevant items are selected?

Confusion matrix (IR view)

	Answered relevant	Answered not relevant
Reality relevant	TP	FN
Reality not relevant	FP	TN

- In IR, TN is typically huge $\rightarrow$ accuracy not useful.

How many documents to show?

- Ranked retrieval produces a list. User reads only first k.
- Short lists $\rightarrow$ higher precision, lower recall.
- Long lists $\rightarrow$ higher recall, lower precision.
- Analyze `Precision@k` and `Precision@k` as functions of k.

Precision–Recall plots

- Precision-recall plots show how precision drops as recall increases.
- Plot **Precision (y-axis)** vs **Recall (x-axis)** for various thresholds.
- Commonly, plot interpolated precision at 11 points (0%,10%,. . . ,100%).

Precision–Recall plots, example

rank	doc id	relevant?	recall	precision
1	10	no	0	0/1
2	3	yes	1/4	1/2
3	1	yes	2/4	2/3
4	7	no		2/4
5	8	yes	3/4	
6	2	no		
7	4	no		
8	9	no		
9	5	yes	4/4	
10	6	no		

Other evaluation measures

- AUC (area under precision–recall curve, or ROC-related).
- F-measure (harmonic mean):

$$F = \frac{2}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}}$$

- α-F-measure to weight recall vs precision.

Coverage & Novelty

- Coverage: proportion of relevant & known documents retrieved.
- Novelty: proportion of retrieved relevant documents that were unknown to the user.

Relevance feedback — overview

User relevance cycle:

1. User issues query q.
2. System retrieves top k documents.
3. User marks some as relevant / nonrelevant.
4. System refines the query (or model).
5. Iterate if desired.

Rocchio's algorithm (query refinement)

Given original query vector q, relevant set R, nonrelevant NR:

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

- Parameters $\alpha > \beta > \gamma \geq 0$. Often $\gamma = 0$.

Relevance feedback — practical notes

- First round: often substantial recall improvement.
- Subsequent rounds: diminishing returns.
- In web search, precision is prioritized — interactive feedback often not used due to latency and user patience.

Pseudorelevance feedback

- Assume top-k retrieved documents are relevant (no user input).
- Use them to expand/refine the query automatically.
- Pros: no user interaction. Cons: risk of query drift.

References & Further Reading

- C. Manning, P. Raghavan, H. Schütze: *Introduction to Information Retrieval*. Cambridge University Press, 2008.
- R. Baeza-Yates, B. Ribeiro-Neto: *Modern Information Retrieval*. Addison Wesley, 2nd ed., 2011.
- Online resources: Stanford NLP IR book, Wikipedia pages on IR, BM25, tf–idf.