CAIM: Cerca i Anàlisi d'Informació Massiva

FIB, Grau en Enginyeria Informàtica

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

Fall 2018

`http://www.cs.upc.edu/~caim`

2. Information Retrieval Models

Information Retrieval Models, I

Setting the stage to think about IR

What is an Information Retrieval Model?

We need to clarify:

- A proposal for a logical view of documents
  (what info is stored/indexed about each document?),
- a query language
  (what kinds of queries will be allowed?),
- and a notion of relevance
  (how to handle each document, given a query?).

Information Retrieval Models, II

A couple of IR models

Focus for this course:

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

Boolean Model of Information Retrieval
Relevance assumed binary

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

- Order of occurrence considered irrelevant,
- number of occurrences considered irrelevant

  (but a closely related model, called bag-of-words or BoW, does consider relevant the number of occurrences).

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

Each document can be seen as a bit vector of length T, $d = (d_1, \ldots, d_T)$, 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, I
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.

In Lucene: $+/-$ signs on query terms, Boolean operators.

Queries in the Boolean Model, II
A close relative to propositional logic

Analogy:

- Terms act as propositional variables;
- documents act as propositional models;
- a document is relevant for a term if it contains the term,

  that is, if, as a propositional model, satisfies the variable;
- queries are propositional formulas

  (with a syntactic condition of avoiding global negation);
- a document is relevant for a query if, as a propositional model, it satisfies the propositional formula.

Example, I
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, II

Our documents in the Boolean model

$$
\begin{array}{rcccccc}
 & five & four & one & six & three & two \\
d1 = & [\ 0 & 0 & 1 & 0 & 1 & 0\ ] \\
d2 = & [\ 0 & 0 & 0 & 0 & 1 & 1\ ] \\
d3 = & [\ 1 & 1 & 1 & 0 & 1 & 0\ ] \\
d4 = & [\ 0 & 0 & 1 & 1 & 1 & 1\ ] \\
d5 = & [\ 0 & 1 & 0 & 1 & 1 & 0\ ] \\
d6 = & [\ 0 & 0 & 0 & 1 & 1 & 0\ ] \\
d7 = & [\ 1 & 1 & 0 & 0 & 0 & 0\ ]
\end{array}
$$

(Invent some queries and compute their answers!)

Queries in the Boolean Model, III

No ranking of answers

Answers are not quantified:

A document either

- matches the query (is fully relevant),
- or does not match the query (is fully irrelevant).

Depending on user needs and application, this feature may be good or may be bad.

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.

  This option may be very slow in cases of large amounts of “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, I

Basis of all successful approaches

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

Vector Space Model of Information Retrieval, II

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 vector 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

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

Each weight is a product of two terms

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

The term frequency term tf is

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

And the inverse document frequency idf is

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

and df_i = number of documents that contain term t_i.

Example, I

$$
\begin{array}{lcccccccc}
 & & five & four & one & six & three & two & & \mathbf{maxf} \\
d1 = & [& 0 & 0 & 1 & 0 & 1 & 0 &] & 1 \\
d2 = & [& 0 & 0 & 0 & 0 & 1 & 2 &] & 2 \\
d3 = & [& 3 & 1 & 1 & 0 & 1 & 0 &] & 3 \\
d4 = & [& 0 & 0 & 1 & 2 & 1 & 4 &] & 4 \\
d5 = & [& 0 & 3 & 0 & 1 & 1 & 0 &] & 3 \\
d6 = & [& 0 & 0 & 0 & 2 & 3 & 0 &] & 3 \\
d7 = & [& 1 & 1 & 0 & 0 & 0 & 0 &] & 1 \\
 & & & & & & & & & \\
\mathbf{df} = & & 2 & 3 & 3 & 3 & 6 & 2 & &
\end{array}
$$

Example, II

$$
\begin{array}{lccccccccc}
\mathbf{df} = & & & 2 & 3 & 3 & 3 & 6 & 2 & \\
d3 = & & [& 3 & 1 & 1 & 0 & 1 & 0 &] \\
\rightarrow & & & & & & & & & \\
d3 = & & [& \frac{3}{3}\log_2\frac{7}{2} & \frac{1}{3}\log_2\frac{7}{3} & \frac{1}{3}\log_2\frac{7}{3} & \frac{0}{3}\log_2\frac{7}{3} & \frac{1}{3}\log_2\frac{7}{6} & \frac{0}{3}\log_2\frac{7}{2} &] \\
 & = & [& 1.81 & 0.41 & 0.41 & 0 & 0.07 & 0 &] \\
d4 = & & [& 0 & 0 & 1 & 2 & 1 & 4 &] \\
\rightarrow & & & & & & & & & \\
d4 = & & [& \frac{0}{4}\log_2\frac{7}{2} & \frac{0}{4}\log_2\frac{7}{3} & \frac{1}{4}\log_2\frac{7}{3} & \frac{2}{4}\log_2\frac{7}{3} & \frac{1}{4}\log_2\frac{7}{6} & \frac{4}{4}\log_2\frac{7}{2} &] \\
 & = & [& 0 & 0 & 0.61 & 1.22 & 0.11 & 3.61 &]
\end{array}
$$

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.

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

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

$$
\begin{array}{lcccccccc}
d3 & = & [& 1.81 & 0.41 & 0.41 & 0 & 0.07 & 0 &] \\
d4 & = & [& 0 & 0 & 0.61 & 1.22 & 0.11 & 3.61 &]
\end{array}
$$

Then

$$|d3| = 1.898, \quad |d4| = 3.866, \quad d3 \cdot d4 = 0.26$$

and $sim(d3, d4) = 0.035$ (i.e., small similarity).

Query Answering

- Queries can be transformed to vectors too.
- Sometimes, tf-idf weights; often, binary weights.
- $sim(doc, query) \in [0, 1]$.
- Answer: List of documents sorted by decreasing similarity.

- We will find uses for comparing $sim(d1, d2)$ too.