Information Retrieval Methods

Helena Ahonen-Myka
Spring 2007, part 5
Matching methods (1/2)
Translation from Finnish: Greger Lindén

In this part

- Exact matching
  - Boolean search
- Partial matching
  - The vector model
  - Similarity measures

2

Exact matching: Boolean search

- Boolean query:
  - A list of terms that are combined with logical connectives AND, OR and NOT
  - The answer is the documents that satisfy the conditions of the query
  - text AND compression AND retrieval
    - The document is included in the answer if each of these three terms is found in the document (free order)

3

Exact matching: Boolean search

- ”...the **compression** and **retrieval** of large amounts of **text** is an interesting problem...”
- ”...this **text** describes the fractional distillation scavenging technique for **retrieval** of argon from air after **compression**”...

4

Processing a Boolean query

- query: ”text AND compression AND retrieval”
- The search engine finds each query term (possibly modified) in the dictionary file
  - The dictionary tells in how many documents the term occurs (df)
    - text: 8
    - compress: 4
    - retrieve: 6
- The terms are sorted in increasing order of their document frequency df: compress, retrieve, text

5

Processing a Boolean query

- The system reads the least frequent term’s inverted list from the inverted file
- The candidate list = a set of documents that have not yet been eliminated and that can be part of the answer
- The inverted lists of all remaining terms are merged in turn with the candidate list
  - Terms are processed in increasing order of their df

6

Example

- The inverted list of the term ’compress’:
  - <4; 2, 5, 12, 16>
- The inverted list of the term ‘retrieve’ :
  - <6; 2, 7, 12, 16, 20, 21>
- “compress AND retrieve”
  - <3; 2, 12, 16>
- The inverted list of the term ‘text’:
  - <8; 1, 4, 8, 12, 16, 20, 21, 30>
- “compress AND retrieve AND text”
  - <2; 12, 16>

Queries with AND

- In an AND query, a document cannot be part of the answer if it does not belong to all inverted lists
  - → The candidate list cannot get longer during the processing of a query
  - When processing term t, the system goes through the candidate list, and documents which are not in the inverted list of t are removed
  - The candidate list may become empty before all terms have been processed
- When all terms have been processed, the remaining documents in the candidate list are the answer

Queries with OR

- ”text OR data OR image”
- The terms can be processed simultaneously: when merging inverted lists, documents are included only once
  - text: <8; 1, 4, 8, 12, 16, 20, 21, 30>
  - data: <12; 2,4,7,8,10,12,13,15,19,20,21,28>
  - image: <5; 4,5,9,11,12>
- answer: <18;1,2,4,5,7,8,9,10,11,12,13,15,16,19,20,21,28,30>

A conjunction of disjunctions

- A conjunction of disjunctions is a typical type of queries
  - ”(text OR data OR image) AND (compression OR compaction) AND (retrieval OR indexing OR archiving)”
  - As a start value for the candidate list we choose the document set of the “smallest” disjunction; we estimate the size, e.g., by summing up the df values of the terms
    - This is a pessimistic estimate: we do not take any possible overlap into account
  - In the following phase, we merge the candidate list with the “second smallest” set, etc.

More general queries

- All Boolean queries can be transformed into a conjunction of disjunctions
- ”(information AND (retrieval OR indexing)) OR ((text OR data) AND (compression OR compaction))”
- → ”(information OR text OR data) AND (retrieval OR indexing OR text OR data) AND (information OR compression OR compaction) AND (retrieval OR indexing OR compression OR compaction)”

Transformation

- (A and B) or (C and D) =
  (A or C) and (B or C) and
  (A or D) and (B or D)

Queries with NOT

- NOT queries cannot be on their own, they are actually AND NOT queries
- ”text AND NOT data”
  - text: <8; 1, 4, 8, 12, 16, 20, 21, 30>
  - data: <12; 2,4,7,8,10,12,13,15,19,20,21,28>
- We first compute ”text AND data”
  - <4,8,12,20,21>
- We merge the inverted lists of the term ”text” and ”text AND data” in such a way that we remove documents that appear in both lists
  - <1,16,30>

13

Problems with exact matching

- We do not find documents that *almost* match the query
- The order of the answer set is random
- It is rather difficult to form Boolean queries
- It is hard to restrict the size of the answer

14

Problems with exact matching (more in detail)

- We do not find documents that *almost* match the query
  - It is hard to specify the information need unambiguously with search terms→ a very strict border between exact matching and partial matching is not motivated
- The order of the answer set is random
  - The order might be, e.g., the order in which the records have been stored
  - A better result would be the documents in the order of descending probable relevance

15

Problems with exact matching

- It is rather difficult to form Boolean queries
  - A user will easily make mistakes in forming queries
  - ”ski resorts in Sweden and Norway” → ”(Sweden OR Norway) and ski resort”
- It is hard to restrict the size of the answer
  - The result of AND queries is often too small
  - The result of OR queries is often too large

16

Quorum search

- We can try to solve the problems with exact matching by generalising the Boolean query into a Quorum search
- Idea: we automatically extend the query by stagewise simplifying the conditions
- E.g. the user gives the terms a, b, c and d; the system forms the Boolean queries
  - strict condition → looser conditions

17

Example

- a and b and c and d
- (a and b and c) or (a and b and d) or (a and c and d) or (b and c and d)
- (a and b) or (a and c) or (a and d) or (b and c) or (b and d) or (c and d)
- a or b or c or d

18

A Quorum search

- The answer set of retrieved documents will increase when we move from one level to the following looser level
  - On the first level, there are few documents, but relatively more relevant ones
  - On more general levels there are more documents, but relatively less relevant ones
- The user may pick the suitable level that returns a suitable number of documents and fair recall and precision

19

Partial matching

- With partial matching we try to solve the problems with exact matching
- We are able to find documents that only partially match the query
- The answer set is ordered according to how well the document matches with the query
  - The answer set is ordered in probable decreasing relevance order

20

Partial matching

- We do not necessarily need any operators in the query
  - Any text paragraph can be used as a query
- It is easy to restrict the size of the answer
  - The user specifies how many answer documents s/he wants

21

The vector model

- Matching based on the **vector (space) model** is the most common partial matching method
- Before we assumed that in the document collection there are t separate terms; each document is described with t terms (terms and their weights)
- In a Boolean search, we can say that a document is described with **a set** of t terms
- In the vector model each document (and the query) is described with **a vector** with t dimensions

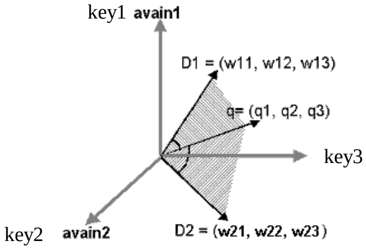
22

The vector model

- We make a simple assumption: the terms are independent of each other → the dimensions are orthogonal to each other
- We have to define a similarity function that describes the similarity between a document and a query (or between two documents)
- The answer documents are ordered according to the similarity value -> ranking of documents

23

The vector model



24

The vector model

- Most similarity functions used in the vector model are based on the inner product
- The inner product of document d_i and query q_j:

$$sim(d_i, q_j) = \sum_{k=1}^{t} d_{ik} \cdot q_{jk}$$

- where d_{ik} is the k^{th} term of document d_i and q_{jk} is the k^{th} term of query q_j

25

The inner product of vectors

- If the weights of the terms in a document vector are binary (0 or 1)
  – the inner product: number of shared terms (both the document and the query have 1)

$$sim(d_i, q_j) = \sum_{k=1}^{t} d_{ik} \cdot q_{jk}$$

  – document i: (1,0,1) and query j: (0,1,1)
    - Inner product: 0+0+1 = 1

26

The inner product of vectors

- x = (1,1,1,0,0,0,0,0)
- y = (1,1,1,0,0,0,0,0)

- x = (1,1,1,1,1,1,1,1)
- y = (1,1,1,0,0,0,0,0)

- The inner product x • y is in both cases 3

27

The inner product of vectors

- If the weights are non-binary
  – The inner product: the sum of the products of the corresponding pairs (term weights)

$$sim(d_i, q_j) = \sum_{k=1}^{t} d_{ik} \cdot q_{jk}$$

  – document i: (0.9, 0.1, 0.9) and query j: (0.1, 0.8, 0.9)
    - Inner product: (0.9 · 0.1) + (0.1 · 0.8) + (0.9 · 0.9) = 0.09 + 0.08 + 0.81 = 0.98
  – query j': (0.9, 0.2, 0.8)
    - Inner product: (0.9 · 0.9) + (0.1 · 0.2) + (0.9 · 0.8) = 0.81 + 0.02 + 0.72 = 1.55

28

The cosine function

- There is no upper limit on the inner product (i.e. maximum value for the similarity)
- Usually, the inner product is normalised with the lengths of the vectors, in which case the function denotes the cosine between the vectors
  – Two similar vectors → the angle is 0°, and cosine 1
  – Very different vectors → the angle is 90°, cosine 0
- Cosine function:

$$\cos(d_i, q_j) = \frac{\sum_{k=1}^{t} d_{ik} \cdot q_{jk}}{\sqrt{\sum_{k=1}^{t} (d_{ik})^2 \cdot \sum_{k=1}^{t} (q_{jk})^2}}$$

29

The cosine function

- The length of the query vector does not influence the ranking of answer documents (the query is the same for all documents)

$$\sqrt{\sum_{k=1}^{t} (q_{jk})^2}$$

- Still it can be useful: the similarity value is always in [0,1] -> the values of different queries are comparable
  – we could have a global similarity threshold to filter the answers

30

The overlap function

- If the documents are very long, the cosine function will give very small values
  - The length of the document affects the denominator directly
  - Queries are usually short, therefore the numerator will not grow in a similar manner
  - We can define a function that does not make longer documents less significant

$$overlap(d_i, q_j) = \frac{\sum_{k=1}^{t} \min(d_{ik}, q_{jk})}{\min(\sum_{k=1}^{t} d_{ik}, \sum_{k=1}^{t} q_{jk})}$$

31

The vector model

- Advantages with the vector model
  - Conceptually simple
  - The weights of the terms are included (in a natural way)
  - Order of similarity
  - It is easy to modify vectors during the retrieval process
- Problems with the vector model
  - We assume in the model that terms are independent even if they are not
  - The similarity measures are heuristic: there are no theoretical grounds for using some measure in a certain situation (or always)

32

In this part

- Exact matching
  - Boolean search
  - Quorum search
- Partial matching: the vector model
  - Similarity measures: inner product of vectors, cosine function, overlap function

33