

Modern Information Retrieval

Vector space model

Hamid Beigy

Sharif university of technology

October 31, 2025

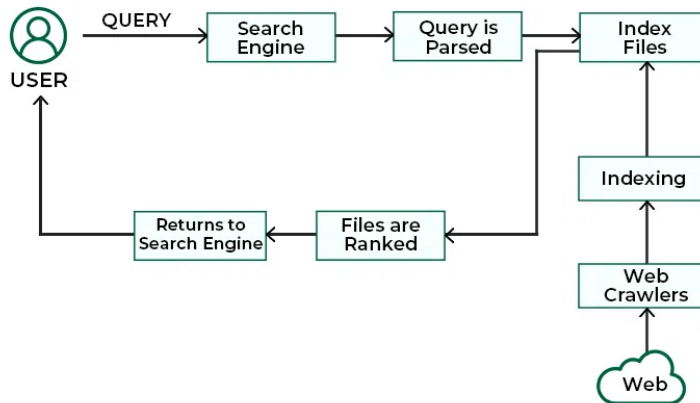




1. Introduction
2. Parametric and zone indexes
3. Term weighting
4. Vector space model
5. Variant tf-idf functions
6. Conclusion
7. References

Introduction

1. Boolean model: all documents *matching* the query are retrieved
2. The matching is *binary*: *yes* or *no*
3. In extreme cases, the list of retrieved documents can be empty or huge
4. A *ranking* of the documents matching a query is needed
5. A *score* is computed for each pair of (*query*, *document*)



Parametric and zone indexes



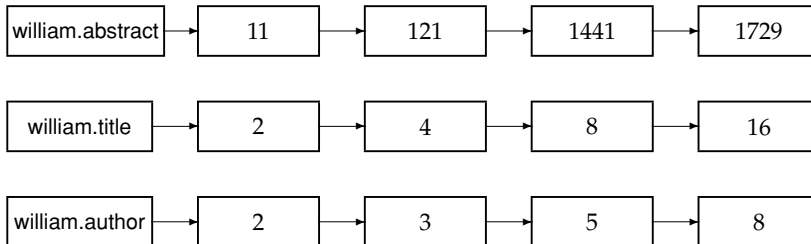
1. Digital documents generally encode, in machine-recognizable form, certain **metadata**, such
 - author(s),
 - title,
 - publication date
2. These metadata would generally include fields, such as the creation data and the format of the document, author and the title of the document.
3. Consider query **find documents authored by William Shakespeare in 1601, containing the phrase alas poor Yorick.**
4. Query processing then consists as usual of postings intersections, except that we may merge postings from standard inverted as well as parametric indexes.
5. There is one parametric index for each field (say, date of creation); it allows us to select only the documents matching a date specified in the query.

1. Parametric search

Bibliographic Search

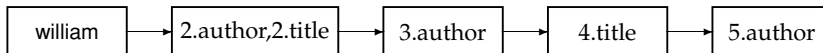
Search category	Value
Author	<i>Example: Widom, J or Garcia-Molina</i>
Title	<i>Also a part of the title possible</i>
Date of publication	<i>Example: 1997 or <1997 or >1997 limits the search to the documents appeared in, before and after 1997 respectively</i>
Language	Language the document was written in English v
Project	ANY v
Type	ANY v
Subject group	ANY v
Sorted by	Date of publication v
Start bibliographic search	

1. Zones are similar to fields, except **the contents of a zone can be arbitrary free text**.
2. A field may take on a relatively small set of values, a zone can be thought of as an arbitrary, unbounded amount of text.
3. We may build a separate inverted index for each zone of a document.
4. Consider query **find documents with william in the title and william in the author list and the phrase gentle rain in the body**





1. The dictionary for a parametric index comes from a fixed vocabulary (the set of languages, or the set of dates), the dictionary for a zone index must structure whatever vocabulary stems from the text of that zone.
2. We can reduce the size of the dictionary by encoding the zone in which a term occurs in the postings.



3. How do you compute the score of a document for a given query?



1. Zones (or fields) can be weighted differently to compute each document's relevance.
2. Scoring is the basis for ranking or sorting document that are returned from a query.
3. Ideally the score is high when the document is relevant.
4. Let $g = (0.6, 0.3, 0.1)$ be the weights, where g_k is weight of zone k .
5. Consider the weighted sum of weights as

$$\text{Score}(d) = 0.6(\text{William} \in \text{Title}) + 0.3(\text{William} \in \text{Abstract}) + 0.1(\text{William} \in \text{Body})$$

6. Weights can be determined using
 - the experts,
 - learning from data $\{(q, d, \text{relevance}/\text{nonrelevance})\}$

Term weighting



1. Evaluation of how important a term is with respect to a document
2. First idea: the more important a term is, the more often it appears: *term frequency*

$$tf_{t,d} = \sum_{x \in d} f_t(x) \text{ where } f_t(x) = \begin{cases} 1 & \text{if } x = t \\ 0 & \text{otherwise} \end{cases}$$

3. The *order of terms* within a doc is ignored
4. Are all words *equally important* ? What about stop-lists ?
5. Terms *occurring very often* in collection are *irrelevant* for distinguishing among documents
6. A relevance measure cannot only take term frequency into account
7. Idea: reducing the relevance (weight) of a term using a factor growing with the *collection frequency* (*the total number of occurrences of a term in the collection*).
8. *Collection frequency versus document frequency* ?

Term t	cf_t	df_t
try	10422	8760
insurance	10440	3997



1. *Inverse document frequency* of a term t :

$$idf_t = \log \frac{N}{df_t} \quad \text{with } N = \text{collection size}$$

2. Rare terms have high idf , contrary to frequent terms
3. Example (Reuters collection):

Term t	df_t	idf_t
car	18165	1.65
auto	6723	2.08
insurance	19241	1.62
best	25235	1.5



1. The weight of a term is computed using both *tf* and *idf*:

$$w(t, d) = tf_{t,d} \times idf_t \quad \text{called } tf - idf_{t,d}$$

2. $w(t, d)$ is:

- high when t occurs many times in a small set of documents
- low when t occurs fewer times in a document, or when it occurs in many documents
- very low when t occurs in almost every document

3. **Score** of a document with respect to a query:

$$score(q, d) = \sum_{t \in q} w(t, d)$$

Vector space model



1. Each term t of the dictionary is considered as a *dimension*
2. A document d can be represented by the weight of each dictionary term:

$$V(\vec{d}) = (w(t_1, d), w(t_2, d), \dots, w(t_n, d))$$

3. Question: does this representation allow to compute the similarity between documents?
4. Similarity between vectors? *inner product* $V(\vec{d}_1) \cdot V(\vec{d}_2)$
5. What about the length of a vector ?



1. Euclidian normalization (vector length normalization):

$$v(\vec{d}) = \frac{V(\vec{d})}{\|V(\vec{d})\|} \quad \text{where } \|V(\vec{d})\| = \sqrt{\sum_{i=1}^n x_i^2}$$

2. Similarity given by the *cosine* measure between normalized vectors:

$$sim(d_1, d_2) = v(\vec{d}_1) \cdot v(\vec{d}_2)$$

3. Consider the following example

Dictionary	$v(\vec{d}_1)$	$v(\vec{d}_2)$	$v(\vec{d}_3)$
affection	0.996	0.993	0.847
jealous	0.087	0.120	0.466
gossip	0.017	0	0.254

$$sim(d_1, d_2) = 0.999$$

$$sim(d_1, d_3) = 0.888$$



1. Queries are represented using vectors in the same way as documents
2. In this context:

$$\text{score}(q, d) = v(\vec{q}) \cdot v(\vec{d})$$

3. In the previous example, with $q := \text{jealous gossip}$, we obtain:

$$v(\vec{q}) \cdot v(\vec{d}_1) = 0.074$$

$$v(\vec{q}) \cdot v(\vec{d}_2) = 0.085$$

$$v(\vec{q}) \cdot v(\vec{d}_3) = 0.509$$



1. Basic idea: Compute cosine similarity between query and each document, finally select the K -documents with top scores.
2. We use the $tf - idf_{t,d}$ measure as a weight, which information do we store in the index ?
 - The size of the collection divided by the document frequency, $\frac{N}{df_t}$, (stored with the pointer to the postings list)
 - The term frequency $tf_{t,d}$ (stored in each posting)
3. We can compute weights as we retrieve postings

```
COSINESCORE( $q$ )
1  float Scores[ $N$ ] = 0
2  float Length[ $N$ ]
3  for each query term  $t$ 
4  do calculate  $w_{t,q}$  and fetch postings list for  $t$ 
5    for each pair( $d, tf_{t,d}$ ) in postings list
6    do Scores[ $d$ ] +=  $w_{t,d} \times w_{t,q}$ 
7  Read the array Length
8  for each  $d$ 
9  do Scores[ $d$ ] = Scores[ $d$ ] / Length[ $d$ ]
10 return Top  $K$  components of Scores[]
```

Variant tf-idf functions



1. Idea: balancing the number of occurrences of a term, using a logarithm

$$w_{t,d} = \begin{cases} 1 + \log(tf_{t,d}) & \text{if } tf_{t,d} \geq 1 \\ 0 & \text{otherwise} \end{cases}$$

2. The relevance of a term is not directly proportional to its frequency



1. Idea: normalizing $tf_{t,d}$ with the maximum term frequency of the document d

$$tf_{max}(d) = \max_{\tau \in d} tf_{\tau,d}$$

$$ntf_{t,d} = a + (1 - a) \frac{tf_{t,d}}{tf_{max}(d)}$$

2. $0 \leq a \leq 1$ is a smoothing coefficient (generally set to 0.4)
3. a allows to avoid having big changes of $ntf_{t,d}$ while $tf_{t,d}$ slightly changes



1. Named after a widely used IR system whose development started at Cornell University (US)
2. Library of weightings schemes fitting the Vector Space Model (cosine similarity)
3. Based on the following weighting:

$$w(t, d) = \frac{tf'_{t,d} \times idf'_t}{norm'_d}$$

4. where (i) $tf'_{t,d}$, (ii) idf'_t , and (iii) $norm'_d$ are parameter of the system

1. Frequency weighting, discrimination and normalization:

$tf'_{t,d}$	idf'_t	$norm'_d$
$b \quad \{0, 1\}$	$n \quad 1$	$n \quad 1$
$n \quad tf_{t,d}$	$t \quad idf_t = \log\left(\frac{N}{df_t}\right)$	$c \quad \frac{1}{\sqrt{w_1^2 + \dots + w_n^2}}$
$l \quad 1 + \log(tf_{t,d})$	$p \quad \max(0, \log\left(\frac{N - df_t}{df_t}\right))$	$p \quad K(cf \text{ supra})$
$m \quad ntf_{t,d}$		
$a \quad 0.5 + \frac{0.5 \times tf_{t,d}}{\max_t(tf_{t,d})}$		

- The mnemonic *ddd.qqq* is used (term/document/normalization).
- $tf - idf_{t,d} := ntc$
- doc and query can use different parameters

Conclusion



1. What we have seen today ?
 - Term weighting using $tf - idf_{t,d}$
 - Vector space model (cosine similarity)
 - Optimizations for document ranking
2. Next lecture ?
 - Other weighting schemes

References



1. Chapters 6 of [Information Retrieval Book](#)¹
2. Section 7.1 of [Search Engines - Information Retrieval in Practice Book](#)²

¹Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze (2008). *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press.

²W. Bruce Croft, Donald Metzler, and Trevor Strohman (2009). *Search Engines - Information Retrieval in Practice*. Pearson Education.



-  Croft, W. Bruce, Donald Metzler, and Trevor Strohman (2009). *Search Engines - Information Retrieval in Practice*. Pearson Education.
-  Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze (2008). *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press.

Questions?