Modern Information Retrieval

Vector space model[1]

Hamid Beigy

Sharif university of technology

October 30, 2022

[1] Some slides have been adapted from slides of Manning, Yannakoudakis, and Schütze.

Table of contents

Introduction

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

Parametric index

1. Digital documents generally encode, in machine-recognizable form, certain metadata, such author(s), title, and date of publication of a document.
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

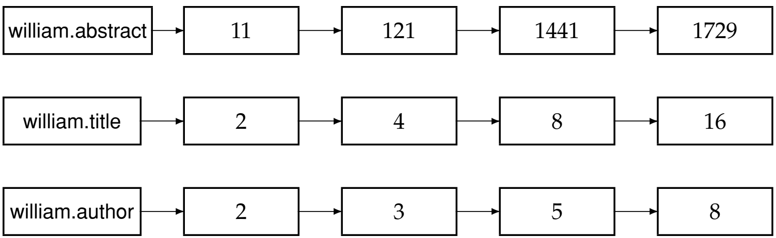
Bibliographic Search

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

Find document via ID

Zone index

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



Zone index

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.

   william → 2.author,2.title → 3.author → 4.title → 5.author

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

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 ?

1. Terms occurring very often in the collection are not relevant for distinguishing among the documents
2. A relevance measure cannot only take term frequency into account
3. 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).
4. *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:

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

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*:

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

2. $w(t, d)$ is:
   2.1 high when t occurs many times in a small set of documents
   2.2 low when t occurs fewer times in a document, or when it occurs in many documents
   2.3 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 $V(\vec{d_1}).V(\vec{d_2})$
5. What about the length of a vector ? Longer documents will be represented with longer vectors, but that does not mean they are more important

Vector normalization and similarity

1. Euclidian normalization (vector length normalization):

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

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

$$sim(d_1, d_2) = v(\vec{d_1}).v(\vec{d_2})$$

3. This similarity measure can be applied on a $M \times N$ *term-document* matrix, where M is the size of the dictionary and N that of the collection:

$$m[t, d] = v(\vec{d}).v(\vec{t})$$

Example (Manning et al, 07)

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:

$$score(q,d) \;=\; v(\vec{q}).v(\vec{d})$$

3. In the previous example, with $q :=$ *jealous gossip*, we obtain:

$$\begin{aligned} v(\vec{q}).v(\vec{d_1}) &= 0.074 \\ v(\vec{q}).v(\vec{d_2}) &= 0.085 \\ v(\vec{q}).v(\vec{d_3}) &= 0.509 \end{aligned}$$

Retrieving documents

1. Basic idea: similarity cosines between the query vector and each document vector, finally selection of the top K scores
2. Provided we use the $tf - idf_{t,d}$ measure as a weight, which information do we store in the index ?
    2.1 The size of the collection divided by the document frequency, $\frac{N}{df_t}$, (stored with the pointer to the postings list)
    2.2 The term frequency $tf_{t,d}$ (stored in each posting)

Cosine similarity

CosineScore(q)

$float\ Scores[N] = 0$
$float\ Length[N]$
for each query term t
do calculate $\mathrm{w}_{t,q}$ and fetch postings list for t
    for each pair$(d, \mathrm{tf}_{t,d})$ in postings list
    do $Scores[d] += \mathrm{w}_{t,d} \times \mathrm{w}_{t,q}$
Read the array *Length*
for each d
do $Scores[d] = Scores[d]/Length[d]$
return Top K components of $Scores[]$

Variant tf–idf functions

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

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

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

Maximum term frequency normalization

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

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

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

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

Limitations of maximum tf normalization

1. lack of stability with respect to the stop-list
2. what if the document contains a high-occurrence term that is not relevant with respect to the document's topic ? (inter versus intra-document frequencies)
3. No distinction of the case when the most frequent term has the same number of occurrences of others

SMART weightings

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

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

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

SMART weightings

- Frequency weighting, discrimination and normalisation:

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

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

Conclusion

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

Reading

1. Chapters 6 of Information Retrieval Book[2]

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

References

Questions?