IR: static (fixed-~~data~~ information range)

IF: on-going (stream). (can add ~~some~~ some words)

pre-processing: (NLP).

(P) set (underlying docu). use same pre-process Q (set).

(key words, phrase ...)

comparison

result.

paragram: bag of words. represent the to.

Stemming: dis: over stemming {skiing, skies.

{obsorb, obsorp

partial string match {abcdef, fabcdcfg.

stop-words: dis: someone has meaning in phrase.

"The the" someone has certain meaning.

Thesaurus construction: thesis ↔ diser-ation. or thesis

related words: messi {soccer, barcelona, psg.

IR.

D. (pseudo-sense).

(could be a bag of words)

(set of terms).

stems

...

Q.

framework. (mathematical).

Ranking ← sim(query, doc).

indexing:

t_1 t_2 ... t_n.

d_1 w_{11} w_{21}

doc 2

d_3

$w_{i,j}$: how well the term discribe this doc. / relate to.

how to assign weights.

this matrix could be huge / sparse. => it could be inefficient.

(weight can be score in vectors, some of them could be 0).

how to ~~see a~~ find a good weight for each term?

Boolean Model.

to binary matrix

	t_1	t_2	t_3	t_n
d_1	0	0	1	0
d_2				
d_3				
⋮				

t set.

$t_1 \wedge t_2$

$t_1 \vee t_2$

disadv: no difference in weight.

coffee

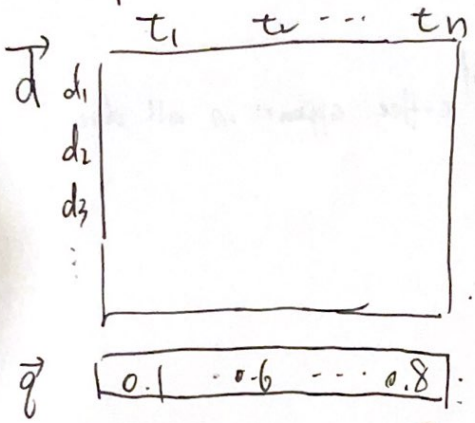
d_1

. take no account to d_2.

frequency

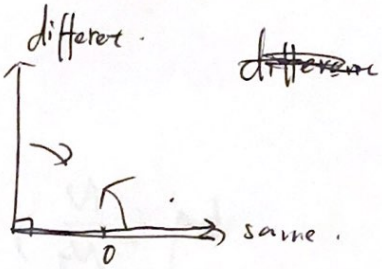
. each term vectors are independent

Vector space Model.



$0 \le w_{ij} \le 1$. (default).

Similarity



$$sim(q,d) = \frac{\sum}{\sum \quad \sum}$$

$\sum_{t \in q \wedge d}$

calculate instance

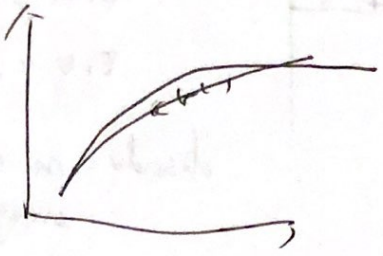
already calculated.

return the short documents which contain my query.

normalise based on the length of query and documents.

tf : local

idf : global.



function of frequency.

1 → 2 difference is huge

50 → 51 less.

f(freq)

tf × idf

(local) (global).

→. e.g. if coffee appears in all doc.

normalisation

t_{ij} $\log\left(\frac{N}{N_j}\right)$

↓ ↓

million of terms frequency (millions of documents)

↓

idf