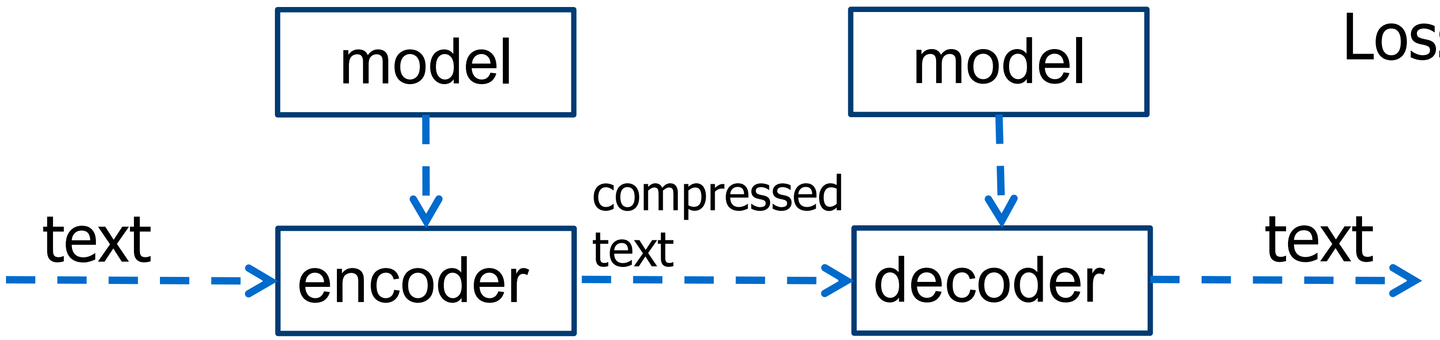
Text compression: dictionary models

IN4325 – Information Retrieval

Text compression



Lossless!

- 2 classes: **symbolwise** and **dictionary** methods

Symbolwise compression

- **Modeling**: estimation of symbol probabilities (➔statistical methods)
  - Frequently occurring symbols are assigned shorter codewords
  - E.g. in English ‘e’ is a very common character, ‘the’ is a common term in most texts, etc.
- **Coding**: conversion of probabilities into a bitstream
  - Usually based on either Huffman coding or arithmetic coding

Dictionary models

- Replace substrings in a text with a codeword that identifies the substring in a dictionary (codebook)
  - December→12, "the chord of B minor" →Bm,..
- Fixed codewords instead of probability distributions (coding component is not that important)
- Digram coding
  - Selected pairs of letters are replaced with codewords
  - A codebook for the ASCII set might contain 128 ASCII characters and 128 common letter pairs
  - Static codebooks are not suitable for all texts

Dictionary models

Ziv and Lempel, 1977

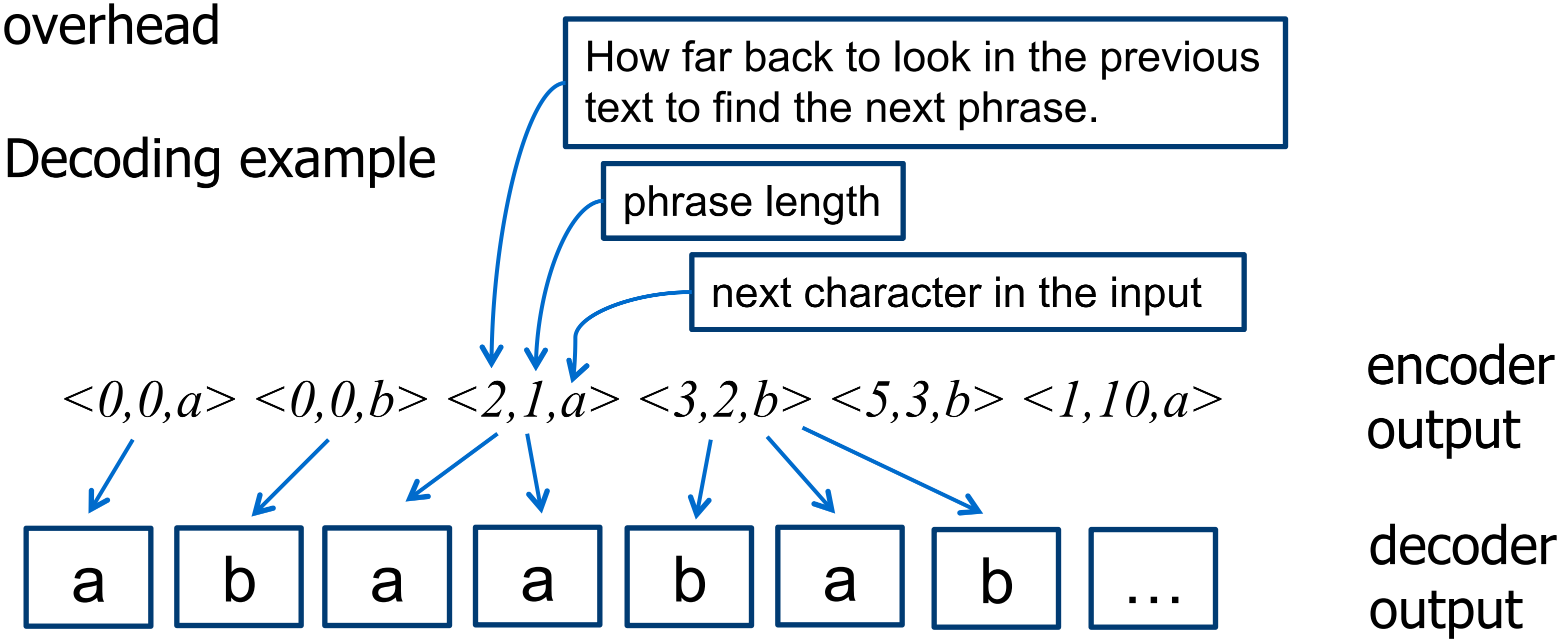
- Semi-static dictionary: construct a codebook for each text to be compressed
  - Overhead of storing/transmitting the dictionary

- Adaptive dictionary: all methods are based on two methods developed by Jacob Ziv and Abraham Lempel (LZ77, LZ78)
  - A substring of text is replaced with a pointer to where it occurred previously
  - Codebook is thus all the text prior to the current position and codewords are represented by pointers
  - No explicit dictionary transmission (the text IS the dictionary)

LZ77 family of adaptive dictionary coders

Ziv and Lempel, 1977

- Easy to implement
- Very fast decoding with only a small amount of memory overhead
- Decoding example



Source: [5], Figure 2.32 (page 76)

LZ77 family of adaptive dictionary coders

Ziv and Lempel, 1977

- Encode text $S[1..N]$ with sliding window W

  ① Set $p=1$ (next character of S to be coded)
  ② While there is more text
    ① Search for the longest match for $S[p...]$ in $S[p-W..p-1]$; suppose the match occurs at position m, with length l
    ② Output the triple $(p-m,l,S[p+1])$
    ③ Set $p=p+l+1$

- Further compression by using different pointer representations; compession can be accelerated by indexing the prior text in the window

Source: [5]

Retrieval models I: Vector Space Model

IN4325 – Information Retrieval

Relevance

Saracevic, 2007 [6]

- The **key notion** in information retrieval
- A good retrieval system retrieves all the relevant documents but as few non-relevant documents as possible
- Relevance is an intuitive notion for humans
- Retrieval systems *create* relevance, and users *derive* relevance

Relevance

Saracevic, 2007 [6]

"relevance is a tangled affair"

interacting layers

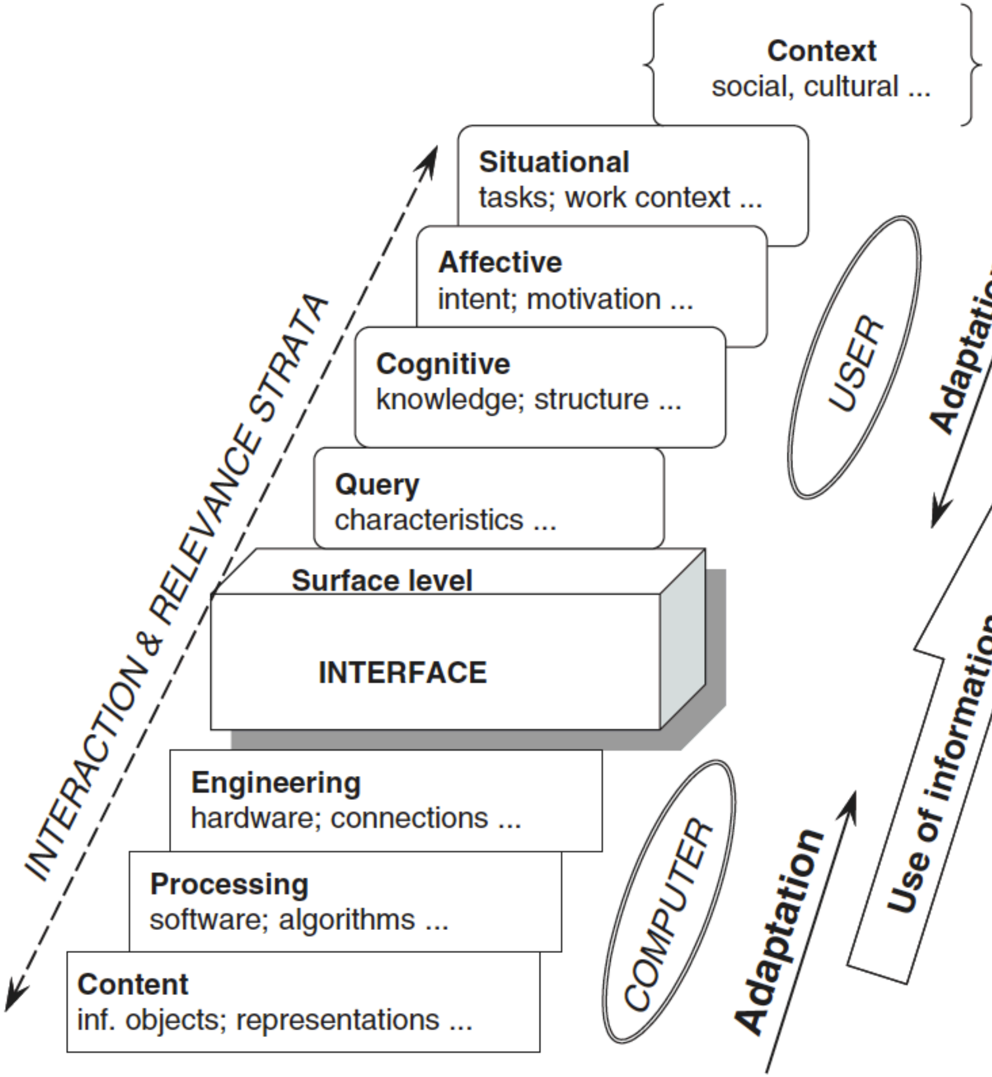


FIG. 1. Stratified model of relevance interactions.

Manifestations of relevance

Saracevic, 2007 [6]

- **System relevance**: relation between query and information objects (documents)
- **Topical relevance**: relation between the subject of the topic and the subject of the information objects
- **Cognitive relevance** (pertinence): relation between the cognitive state of the user and the information objects
  - Cognitive correspondence, novelty, information quality, etc.
- **Situational relevance** (utility): relation between the situation and the information objects
  - Appropriateness of information, reduction of uncertainty
- **Affective relevance**: relation between the intent, goals, emotions of the user and information
  - Success, accomplishment

Terminology

$$T \text{ is the set of all terms}$$

$$query\ Q = \{q_1, q_2, \ldots, q_m\},\ q_i \in T$$

$$document\ D = \{t_1, t_2, \ldots, t_n\},\ t_i \in T$$

$$q_i = q_j \text{ and } t_i = t_j \text{ possible, even if } i \neq j$$

$$scoring\ function\ S(Q, D) \in \Re$$

Bag of words

the dog eats the cat
=
the cat eats the dog
=
eats cat dog the the
…..

But

the dog eats the cat
!=
the dog eats cat

Retrieval in short

① Given a query, calculate the score of each document in the collection $S(Q,D)$
   - Score is a measure of a document's match to the query

② Rank the documents wrt. Q

③ Present the top-k ranked documents to the user

- Questions
  - How are documents scored?
  - What are the assumptions behind the scoring functions?

Term frequency & weighting

- Assumption: a document in which more query terms occur (more often) has more to do with the query and thus should receive a higher score
  - Compute $S(Q,D)$ as the sum over all query terms
- Terms in a document are assigned *weights*
  - Score is calculated based on this weight
  - Three components: term frequency, inverse document frequency and document length normalization
- Simplest approach: weight as the frequency of the term in the document $tf_{t,d}$

Term frequency & weighting

- Assumption: a document in which more query terms occur (more often) has more to do with the query and thus should receive a higher score
  - Compute $S(Q,D)$ as the sum over all query terms
- Terms in a document are assigned *weights*
  - Score is calculated based on this weight
  - Three components: term frequency, inverse document frequency and document length normalization
- Simplest approach: weight as the frequency of the term in the document $tf_{t,d}$

Term frequency & weighting

- Assu
(mor
recei
  - C
- Term
  - S
  - T
fr
- Simp
the c

$$D_1 = \{the, dog, eats, the, cat\}$$

$$D_2 = \{my, dog, is, in, the, house\}$$

$$D_3 = \{my, house, in, the, prairie\}$$

$$D_4 = \{waiting, at, work, for, my, dog\}$$

$$w(the, D_1) = tf_{the,D_1} = 2$$

$$w(prairie, D_3) = tf_{prairie,D_3} = 1$$

$$w(the, D_3) = tf_{the,D_3} = 1$$

$$w(prairie, D_4) = tf_{prairie,D_4} = 0$$

equally important?

Inverse collection frequency

- Raw term frequencies consider all terms equally important for $S(Q,D)$
- High frequency terms (stopwords) should be given little power in $S(Q,D)$
- Idea 1: reduce the weight of terms with a high *collection frequency* (total number of occurrences of a term)
  - The higher cf_t, the lower $w(t,D)$

Inverse document frequency

- Idea 2: reduce the weight of terms with a high *document frequency* df_t (number of documents containing t)
  - The higher df_t, the lower $w(t,D)$

commonly, df is used in score functions

- Examples of the Wikipedia corpus (~7Mio docs)

	df	*cf*	*df/cf*
netherlands	63,214	157,659	0.40
the	3,585,710	91,024,521	0.04
physics	123,068	248,338	0.50
actor	147,473	477,476	0.31
chess	14,690	83,641	0.17
indeed	55,735	80,597	0.69

Inverse document frequency

What is the idf_t of a term occurring in every document?

- Let N be the total number of documents

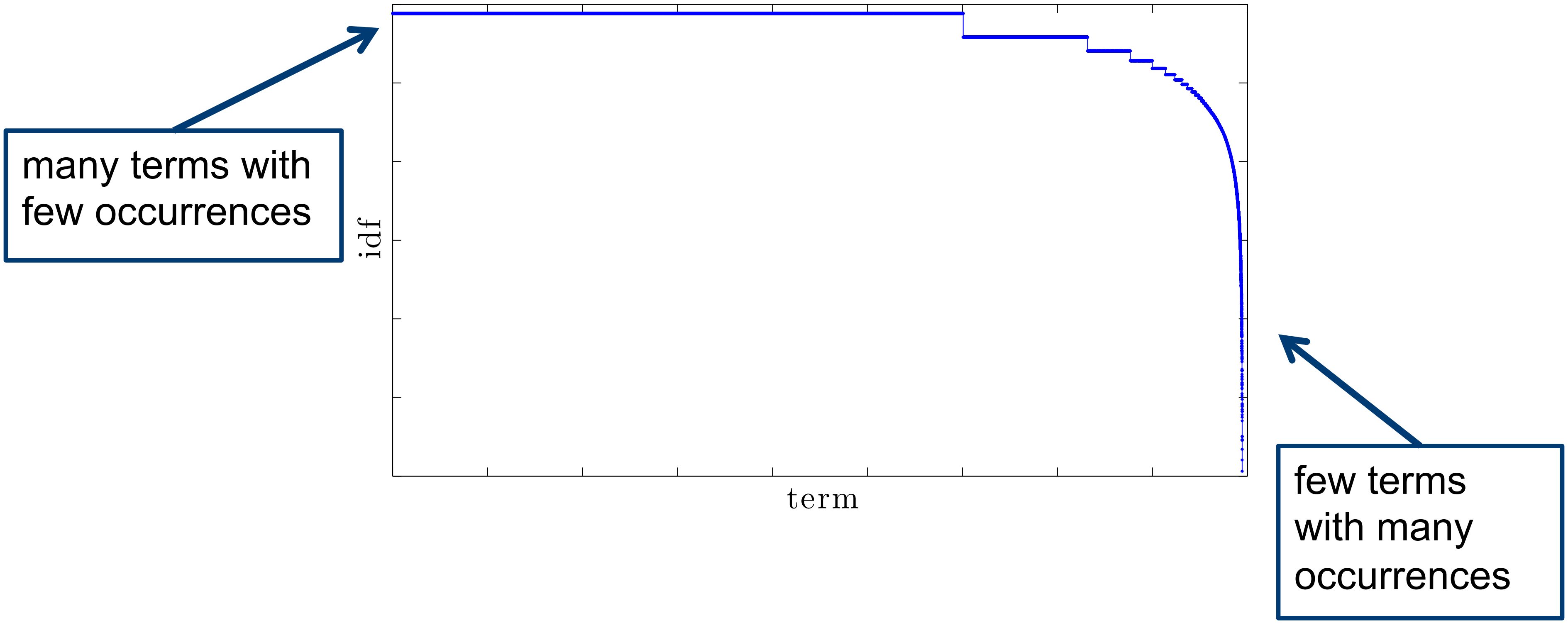
$$idf_t = \log_{10} \frac{N}{df_t}$$

What is the relationship between idf_t and stopword lists?

	df	*cf*	*df/cf*	
netherlands	63,214	157,659	0.40	
the	3,585,710	91,024,521	0.04	0.33
physics	123,068	248,338	0.50	1.79
actor	147,473	477,476	0.31	1.71
chess	14,690	83,641	0.17	2.72
indeed	55,735	80,597	0.69	2.14

Inverse document frequency

- Wikipedia corpus with M=17,711,785



TF-IDF

- Combining term and inverse document frequency results in the *tf-idf* weight of a term:

$$tf - idf_{t,D} = tf_{t,D} \times idf_t$$

- *tf-idf* is highest, when t occurs many times within a small number of documents (cmp. *chess* and *indeed*)
- *tf-idf* is lower when the term occurs fewer times in a document, or occurs in many documents
- *tf-idf* is lowest when the term occurs in all documents.

Vector space model

A vector space model for automatic indexing.
A classic paper by G. Salton et al. from 1975.

- A classic information retrieval model
- Still in use today (e.g. Apache Lucene)

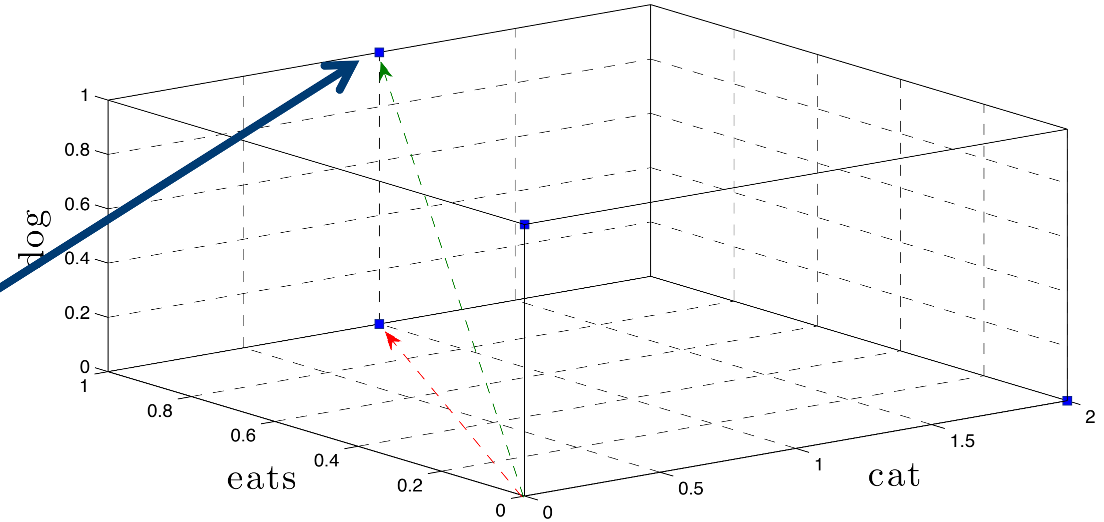
- Corpus: a set of vectors in a vector space with one dimension for each term

$$D_1 = \{cat, eats\}$$

$$D_2 = \{dog\}$$

$$D_3 = \{dog, eats, cat\}$$

$$D_4 = \{cat, cat\}$$



Vector representation

$$D_3 = \{my, house, in, the, prairie\}$$

$$Q = \{dog, prairie\}$$

- Documents and queries can be represented as vectors

$$vocabulary = \begin{pmatrix} the \\ dog \\ eats \\ cat \\ my \\ is \\ in \\ house \\ prairie \\ waiting \\ at \\ work \\ for \end{pmatrix} \quad D_3 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad D_3 = \begin{pmatrix} 1\times 0.1249 \\ 0 \\ 0 \\ 0 \\ 1\times 0.1249 \\ 0 \\ 1\times 0.6021 \\ 1\times 0.3010 \\ 1\times 0.6021 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad Q = \begin{pmatrix} 0 \\ 1\times 0.1249 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1\times 0.6021 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$tf-idf_{t,D} = tf_{t,D} \times idf_t$$

Vector representation

$$D_3 = \{my, house, in, the, prairie\}$$

$$Q = \{dog, prairie\}$$

- Documents and queries can be represented as vectors

$$\begin{pmatrix} the \\ dog \\ eats \\ cat \\ \vdots \\ at \\ work \\ for \end{pmatrix} \quad D_3 = \begin{pmatrix} 1 \\ \vdots \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 1 \times 0.1249 \\ \vdots \\ 1 \times 0.6021 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad Q = \begin{pmatrix} 0 \\ 1 \times 0.1249 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \times 0.6021 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

The representation does not take term sequences into account.

Queries are transferred into the same vector space as the corpus documents

$$tf - idf_{t,D} = tf_{t,D} \times idf_t$$

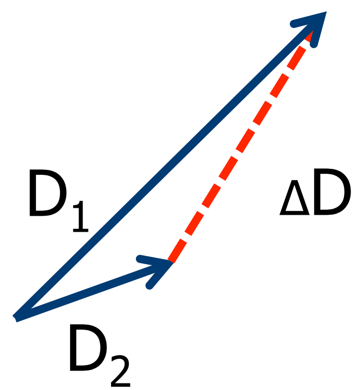
Vector space model

D_1 is much longer than D_2

- Given two vectors, how can we score their similarity?

  ① Score vectors by vector difference

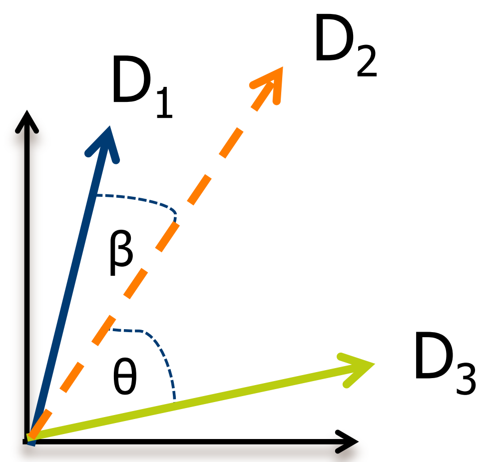
  ② Score vectors according to their cosine similarity



dot product

$$\cos\theta = \frac{D_1 \bullet D_2}{\|D_1\| \times \|D_2\|} = \frac{\sum_{i=1}^{m} w_{i,1} w_{i,2}}{\sqrt{\sum_{i=1}^{m} w_{i,1}^2} \times \sqrt{\sum_{i=1}^{m} w_{i,2}^2}}$$

Euclidean length (length normalization)



Vector space model

The term weights can be binary, *tf*, *idf*, *tf-idf* based or ….

- Example

A document does not need to contain all query terms!

$$Q = \{cat, chaos\}$$

tf $\quad \frac{w_i}{\sqrt{\sum_{i=1}^{m} w_i^2}}, \forall i$

	D_1	Norm.	D_2	Norm.	D_3	Norm.	Q	Norm.
cat	33	0.93	3	0.08	18	0.39	1	0.71
dog	5	0.14	26	0.72	0	0.00	0	0.00
dislike	1	0.03	25	0.69	40	0.86	0	0.00
chaos	12	0.34	0	0.00	15	0.32	1	0.71

$$S_{cosine}(D_1, D_2) = 0.1978 \qquad S_{cosine}(Q, D_1) = 0.8968$$
$$S_{cosine}(D_1, D_3) = 0.4949 \qquad S_{cosine}(Q, D_2) = 0.0586$$
$$S_{cosine}(D_2, D_3) = 0.6282 \qquad S_{cosine}(Q, D_3) = 0.5034$$

$\Rightarrow$

1. D_1
2. D_3
3. D_2

Vector space model

search engines often use $k=10$

COSINESCORE(Q,k)

$$< t; df_t; (D_1, f_{t1}), (D_2, f_{t2}), \ldots, (D_{f_t}, f_{df_t}) >, d_i < d_j \; \forall i < j$$

```
float scores[N]=0
initialize length[N]
for each query term q_i
do calculate w(q_i,Q) and fetch postings list for q_i
      for each pair (D,f_qi) in postings list
      do scores[D]+=w(q_i,D)*w(q_i,Q)
for each D
do scores[D]=scores[D]/length[D]
return top k components of scores[]
```

accumulator

Source: [1] (Figure 6.14)

TF-IDF variants

- Sublinear *tf* scaling

$$w(t,D)=\begin{cases}1+\log(tf_{t,D}) & if\ tf_{t,D}>0\\ 0 & otherwise\end{cases}$$

tf of the most often occurring term in *D*

- Maximum *tf* normalization

smoothing

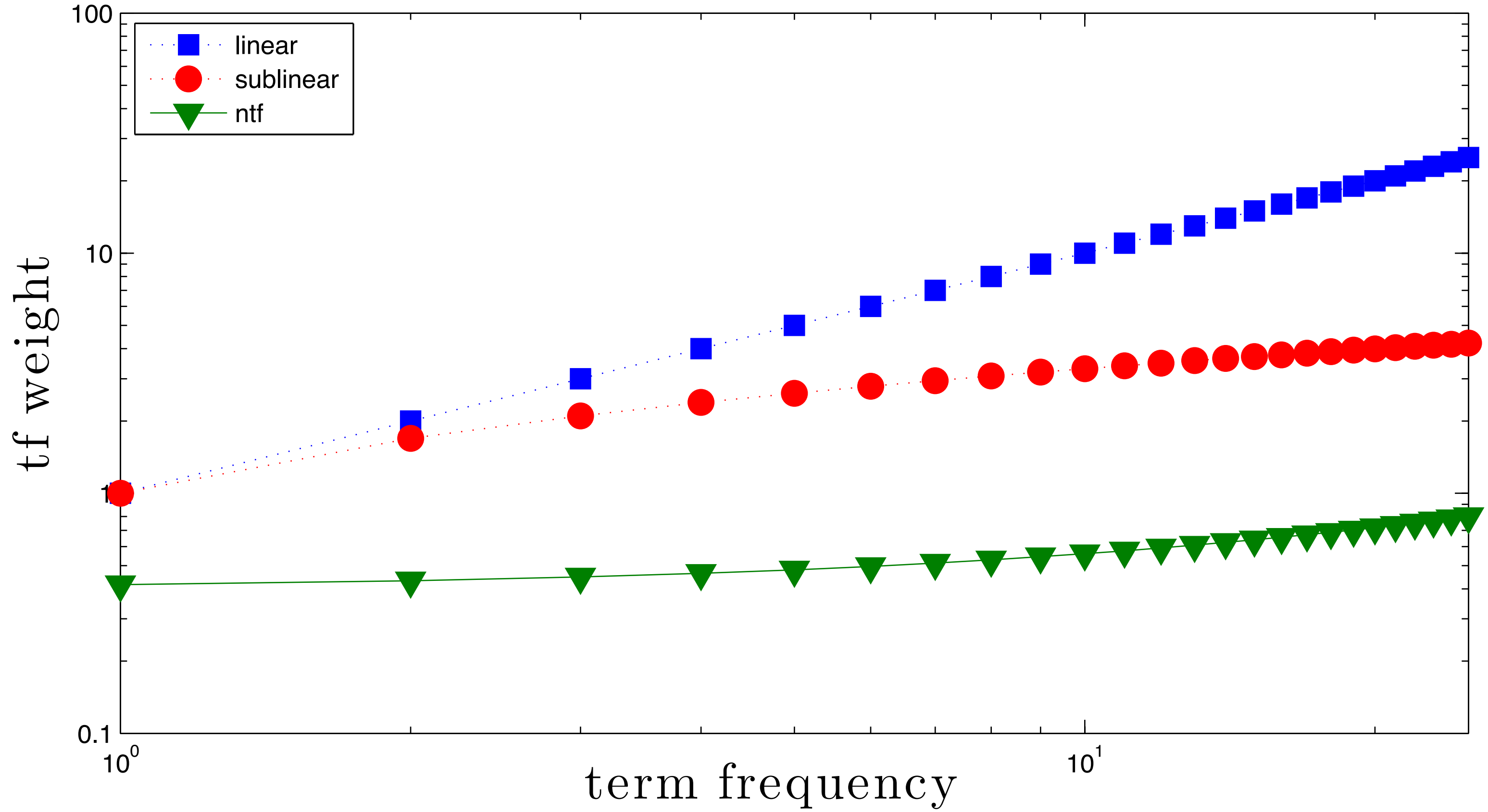
$$ntf_{t,D}=a+(1-a)\frac{tf_{t,D}}{tf_{\max}(D)}$$

$$with\ a\in[0,1]\ and\ often\ a=0.4$$

Tuning of *ntf* is difficult: slight change in stopword list has large effects

TF-IDF variants

between 1-25 terms



Pivoted length normalization

Singhal et al., 1996 [2]

- Normalizing term weights according to document length is important because of:
  - **Higher term frequencies**
  - **More terms** (more chance to encounter a match with a query term), which increases the chances of retrieval of long documents over short ones

- Long documents
  - **Verbose**: cover the same topic repeatedly
  - **Diverse**: cover many different topics

Pivoted length normalization

Singhal et al., 1996 [2]

What about byte length normalization?

$$\frac{tf}{2(1-b+b(\frac{doclen}{av.doc.len}+tf)}$$

- Maximum *tf* normalization
  - Restriction of tf values to a maximum of 1.0 adresses the first aspect
  - Does not adress the second aspect, favors the retrieval of long documents

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

- Cosine normalization
  - Higher tf values increase the denominator
  - More diverse terms yield more individual weights
  - Favours short documents

$$\sqrt{tf^2_{t_1,D} + tf^2_{t_2,D} + tf^2_{t_3,D} + \ldots + tf^2_{t_l,D}}$$

denominator

Higher tf
More terms

Pivoted length normalization

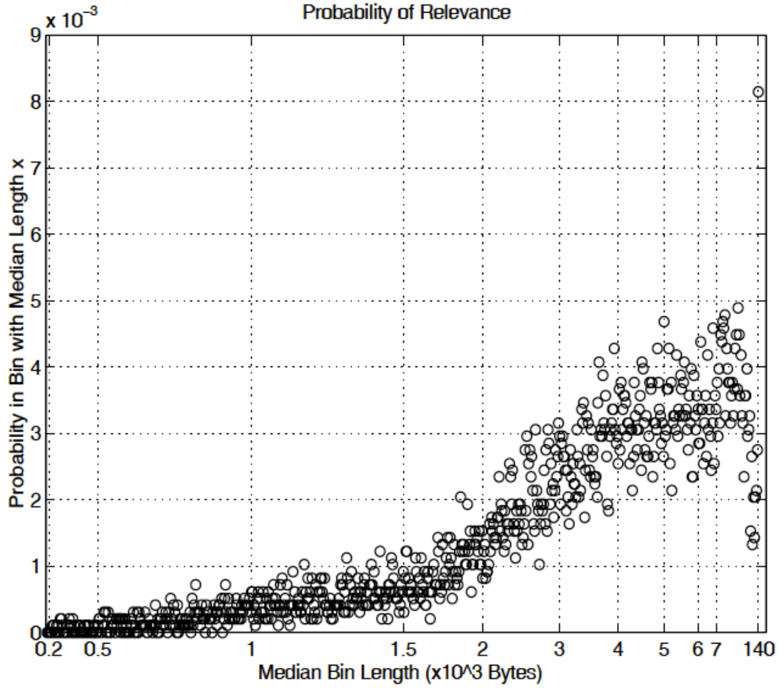
Singhal et al., 1996 [2]

- “Relevance pattern” approach
  - Given a corpus and a set of test queries (together with the respective relevance assessments), the likelihood of relevance is plotted against the document length
- In a good normalization scheme the probability of retrieval for documents of a given length should be similar to the probability of finding a relevant document of that length
- Research question: how does the retrieval pattern diverge from the relevance pattern?
  - Systematic deviations can be alleviated

Source: [2]

Pivoted length normalization

Singhal et al., 1996 [2]



① Documents divided into 'bins' according to their length

② Compute probability of a randomly selected relevant document belonging to a particular bin

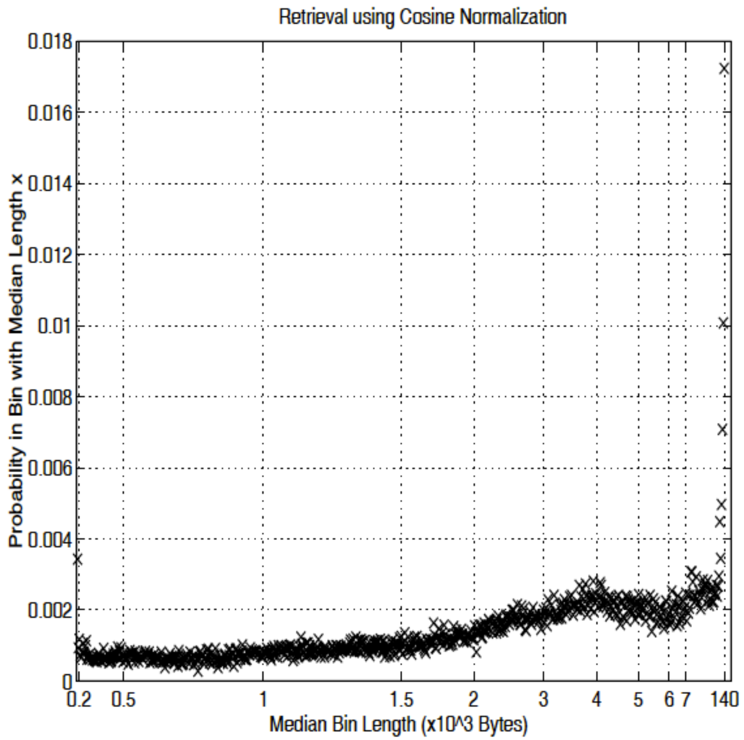
$$P(D \in bin_i \mid D \text{ is relevant})$$

50 queries, ~740,000 TREC documents; 1000 documents/bin; ~9,800 (query,relevant-docid) pairs;

Source: [2]

Pivoted length normalization

Singhal et al., 1996 [2]



① Documents divided into 'bins' according to their length

② Compute probability of a randomly selected retrieved document belonging to a particular bin

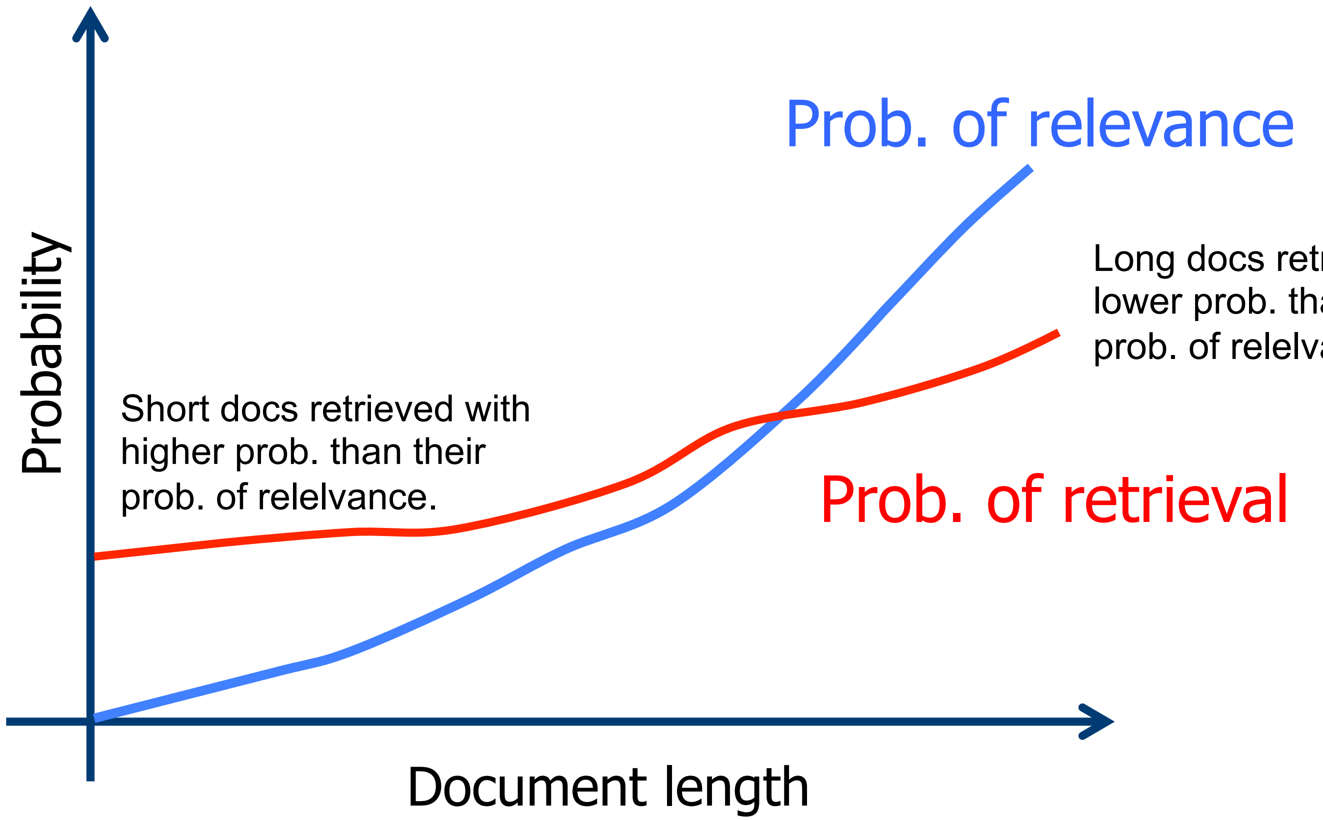
$$P(D \in bin_i \mid D\ is\ retrieved)$$

50 queries, ~740,000 TREC documents; 1000 documents/bin; 50,000 (query,retrieved-docid) pairs (the top 1000 retrieved per query);

Source: [2]

Pivoted length normalization

Singhal et al., 1996 [2]



Source: after [2]

Pivoted length normalization

Singhal et al., 1996 [2]

Prob. of relevance

l with
eir

Idea: adapt cosine based normalization

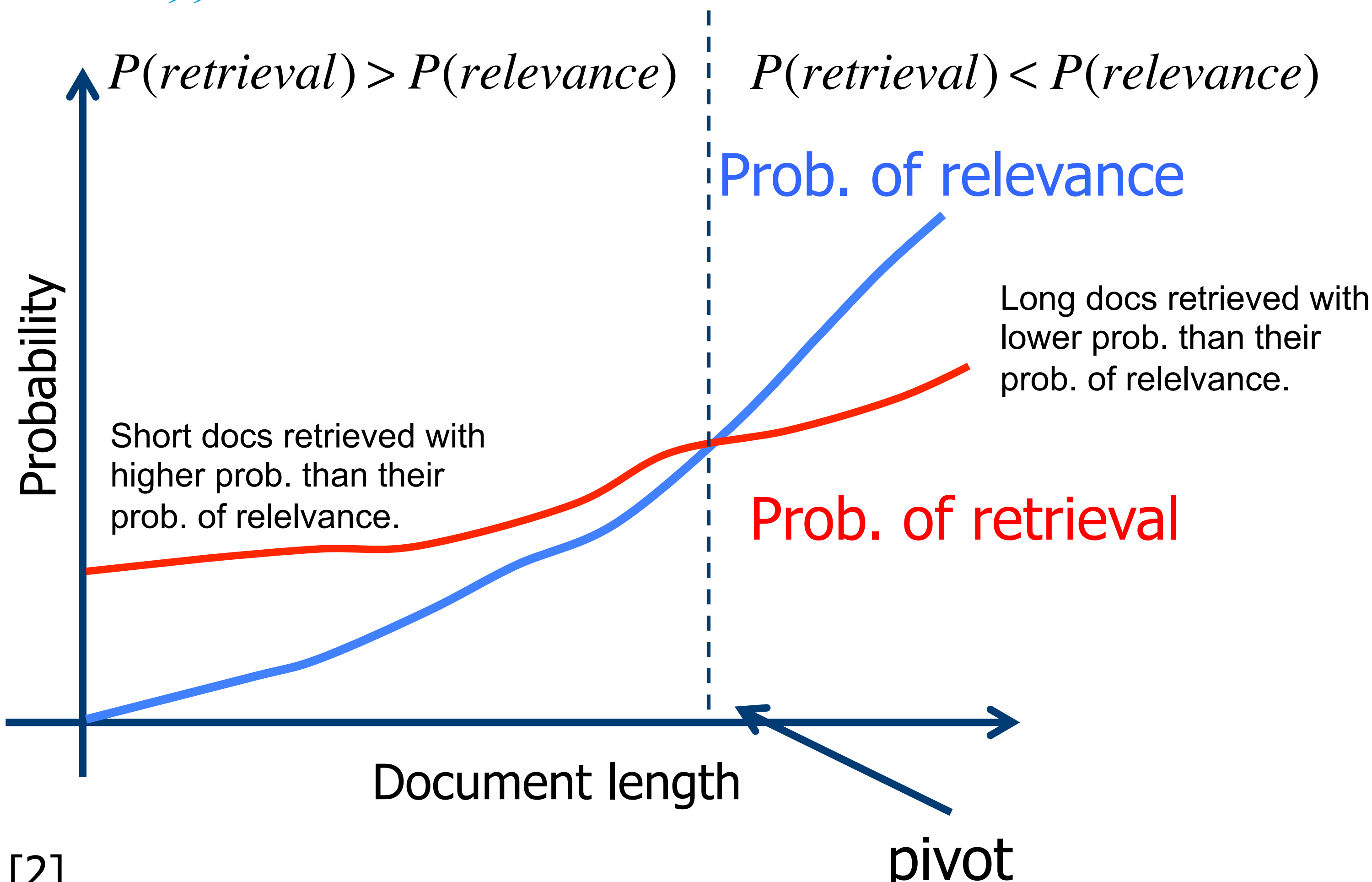
- The retrieval of longer documents is promoted.
- The retrieval of short documents is decreased.

Document length

Source: after [2]

Pivoted length normalization

Singhal et al., 1996 [2]

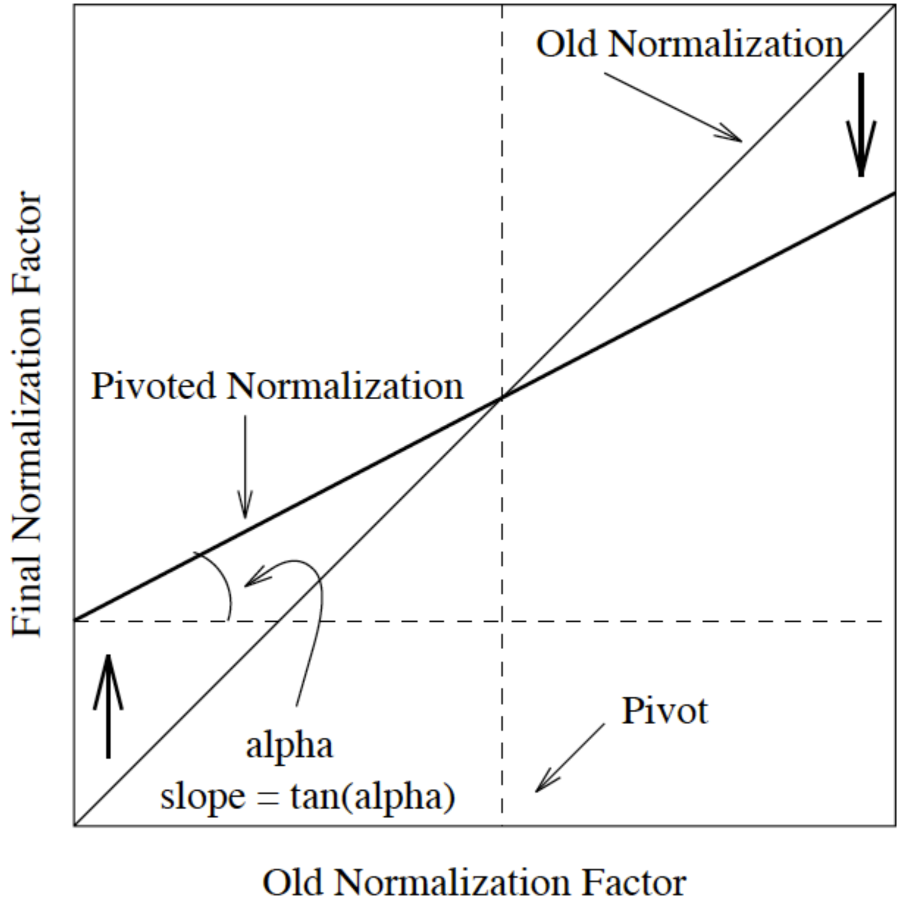


Source: after [2]

Pivoted length normalization

Singhal et al., 1996 [2]

- "Pivoted" normalization sch
  1. Retrieve a set of docume
     (e.g. cosine)
  2. Plot retrieval and relevar
  3. "Tilt" normalization funct
     - Increase normalization o
       normalization on the othe



$$pivoted\,norm. = (1 - slope) \times pivot + slope \times old\ \ norm.$$

Source: [2]

Pivoted length normalization

Singhal et al., 1996 [2]

What happens to a document of average length?

- Revised term weight

$$\frac{tf-idf\ weight}{(1-slope)\times pivot+slope\times old\ norm.}$$

- Fix pivot value to the *average old normalization factor*

$$\frac{tf-idf\ weight\times \boxed{av.old\ norm.}}{(1-slope)\times pivot+slope\times old\ norm.}$$

$$=\frac{tf-idf\ weight\times av.old\ norm.}{(1-slope)\times av.old\ norm+slope\times old\ norm.}\times\frac{\frac{1}{av.old\ norm.}}{\frac{1}{av.old\ norm.}}$$

$$=\frac{tf-idf\ weight}{\boxed{(1-slope)+slope\times\frac{old\ norm.}{av.old\ norm.}}}$$

Adding a constant does not change the relative ranking of documents.

normalization factor

A first look at retrieval evaluation

Retrieval evaluation

Covered in depth in a later lecture.

- Goal: evaluation measures that reflect the users‘ *satisfaction* with the system
- User satisfaction in terms of [3]
  - *coverage* of the corpus
  - *time lag* between query and retrieved results
  - *presentation* of the output
  - required user *effort*
  - proportion of relevant results actually retrieved (*recall*)
  - proportion of retrieved results that is relevant (*precision*)

→ system effectiveness

Assumption: the more effective the system, the more satisfied the user.

Ad hoc retrieval

Evaluation setup

① Corpus of documents

② A set of topics (information needs)

- ~50 is deemed sufficient

③ Relevance judgments

- Is document D_i **relevant** or **non-relevant** to topic *T01*?
- Assume binary decision

"qrels"
(TREC slang)

Ad hoc retrieval

Topic examples

TREC 2001 Web adhoc topic

<top>
<num> Number: 503

<title> Vikings in Scotland?

<desc> Description: What hard evidence proves that the Vikings visited or lived in Scotland?

<narr> Narrative: A document that merely states that the Vikings visited or lived in Scotland is not relevant. A relevant document must mention the source of the information, such as relics, sagas, runes or other records from those times.
</top>

title query (short query)

descr. query (long query) rarely used.

Information need description for assessors.

Ad hoc retrieval

Topic examples

TREC 2006 Blog track

<top>
<num> Number: 851

<title> "March of the Penguins”

<desc> Description: Provide opinion of the film documentary "March of the Penguins".

<narr> Narrative: Relevant documents should include opinions concerning the film documentary "March of the Penguins". Articles or comments about penguins outside the context of this film documentary are not relevant.
</top>

title query (short query)

descr. query (long query) rarely used.

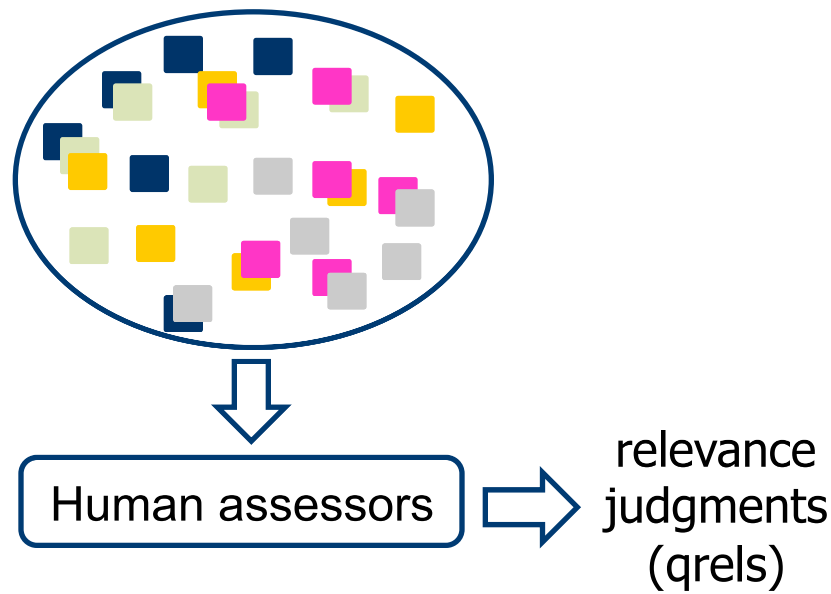
Information need description for assessors.

TREC depth pooling

Commonly used today

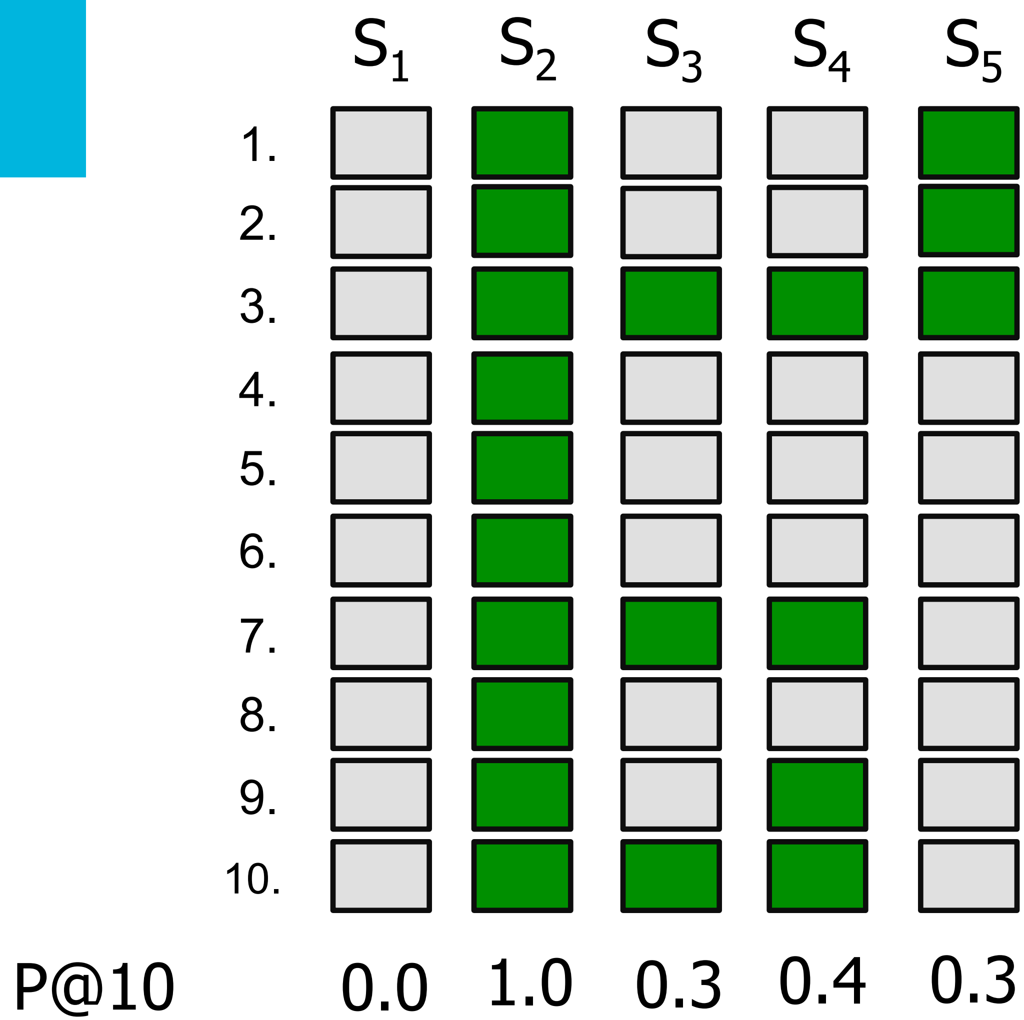
Example: TREC-8
- 50 queries, 129 systems
- 86,830 pooled documents
- ~30 seconds to judge a document
- 723 hours of assessor time

Pool until depth k (e.g. 100)



Precision

One query, five systems.



$$precision = \frac{\#\,rel.retrieved}{\#\,retrieved}$$

Measures a system's ability to only retrieve relevant items.

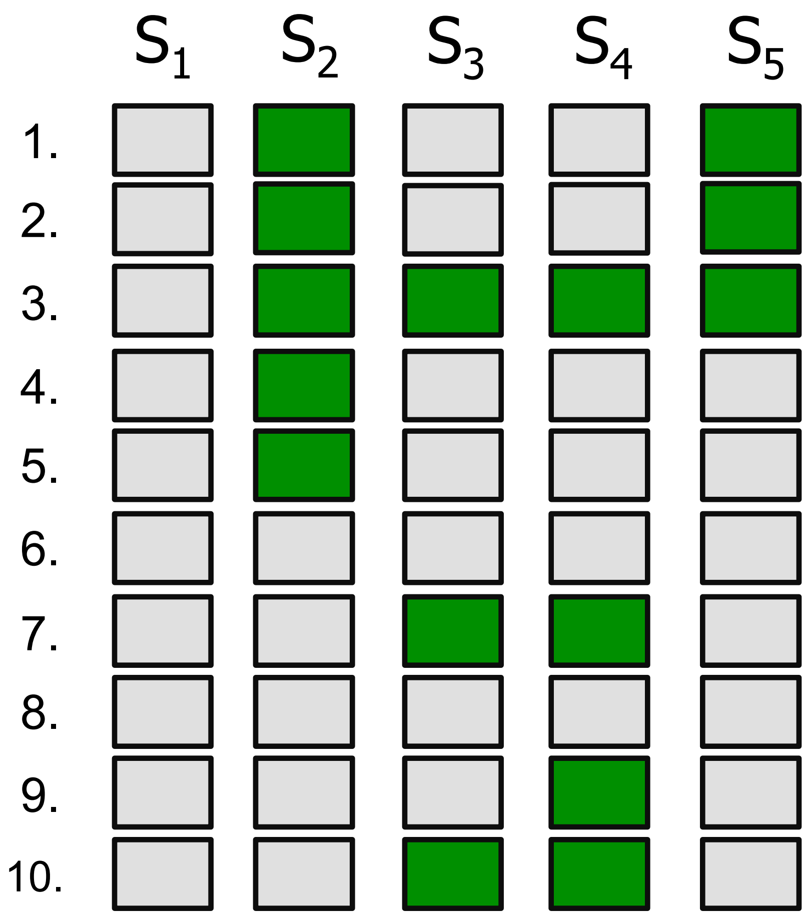
relevant

non-relevant

R-precision:
P@R where R=#rel. documents

Recall

One query, five systems.



Recall 0.0 1.0 0.6 0.8 0.6

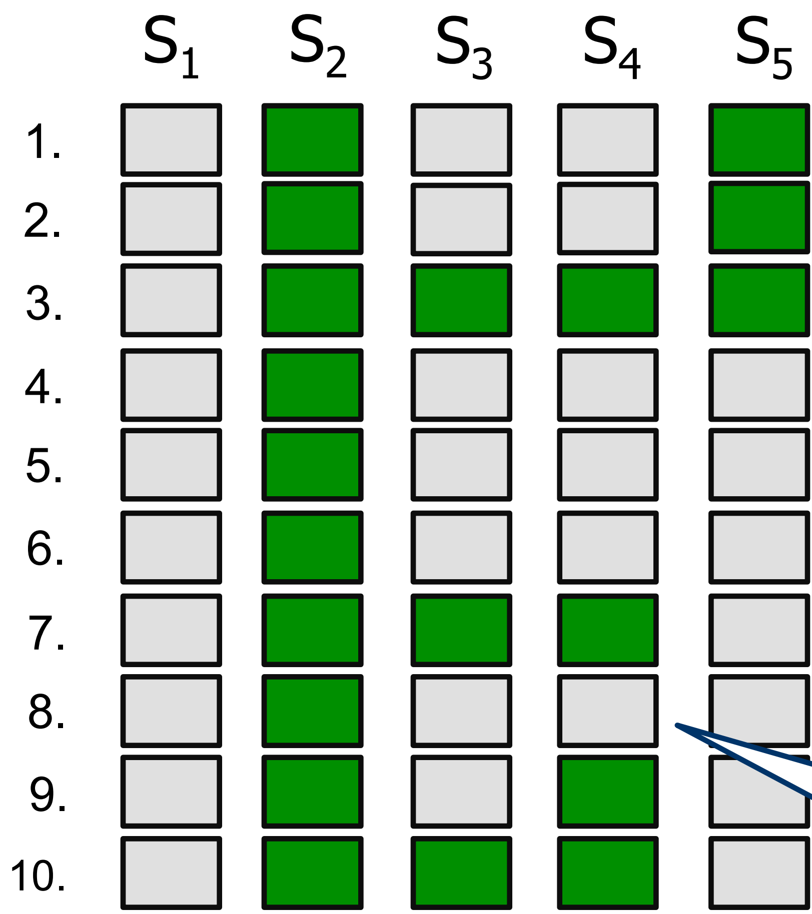
$$recall = \frac{\#\, rel.retrieved}{\#\, relevant \in corpus}$$

Measures a system's ability to retrieve all R relevant items.

Assume $R=5$.

Recall (and precision) are set-based measures. Retrieved are *ranked* lists.

Average Precision



AvP 0.0 1.0 0.09 0.13 0.3

Takes the order of into account.

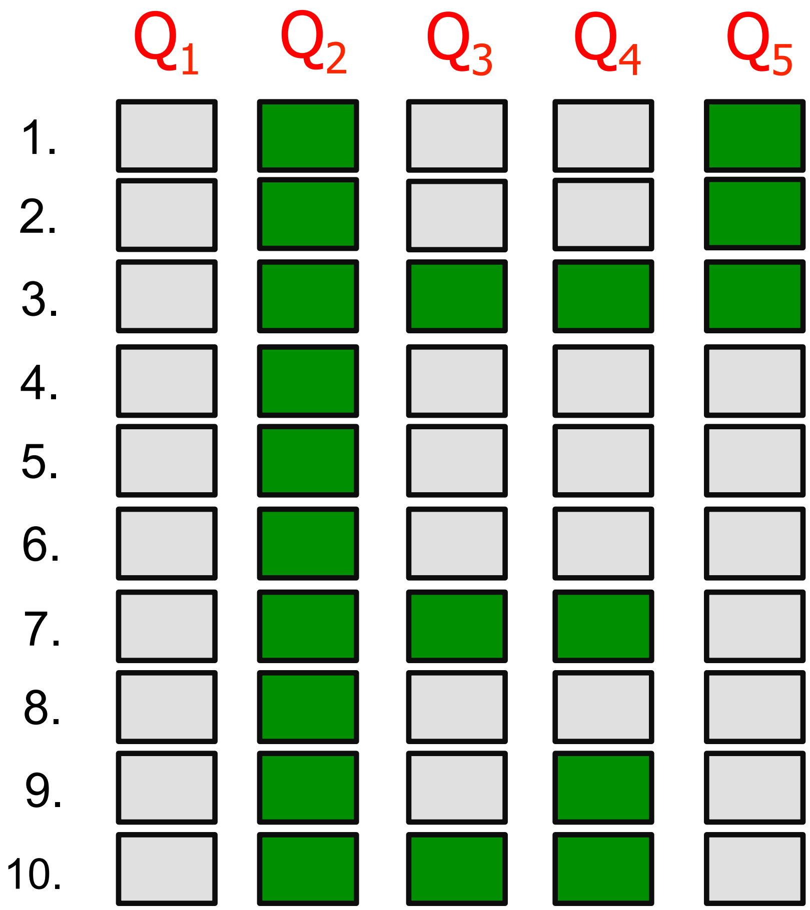
Takes the number R of rel. documents into account. Assume $R=10$.

$$AvP = \frac{\sum_{k=1}^{s} P@k \times rel(k)}{R}$$

$$AvP = \frac{\frac{1}{3}+\frac{2}{7}+\frac{3}{9}+\frac{4}{10}}{10}$$

Mean Average Precision

One system, five queries.



AvP 0.0 1.0 0.09 0.13 0.3

Given a set of queries, the average effectiveness is the mean over AvP.

$$MAP = \frac{1}{|\mathbf{Q}|} \sum_{Q \in \mathbf{Q}} \frac{\sum_{k=1}^{s} P@k \times rel(k)}{R}$$

MAP=0.364

Example: pivoted length normalization

Singhal et al., 1996 [2]

	Cosine	Pivoted Cosine
AP (196 queries)	0.4000	0.4173 (+4.3%)
DOE (80 queries)	0.3046	0.3211 (+5.3%)
FR (111 queries)	0.2314	0.2785 (+20.3%)
WSJ (200 queries)	0.3525	0.3899 (+10.6%)
ZF (122 queries)	0.2829	0.3441 (+21.7%)
TREC (200 queries)	0.3007	0.3357 (+11.6%)

Mean average precision

Efficient index traversal

Inexact top k retrieval

A look at efficient implementation

① Find a set C of documents that are potentially similar to the query with
② Return the top-k documents in C

- Reduces the cost of computation (cosine computations)
- Not likely to change the user's perception of the result quality
  - $S(Q,D)$ is itself an approximation of user's perceived relevance

Inexact top k retrieval

Index elimination

- Originally: given Q, we consider all documents with at least one occurrence of q_i
- IDF elimination: only consider documents with terms exceeding an *idf* threshold (low *idf* → long postings lists)
  - Thus, $low\ idf \approx stopwords$
  - Query adaptive or static
- Many query term selection: consider documents containing many (or all) query terms
  - Requires traversal of postings lists, but no cosine computation
  - Can lead to the selection of fewer than k documents

Inexact top *k* retrieval

Champion lists

- Offline computation of the r documents with the heighest weights for term t
- Given Q, set C is the union of the champion lists for each of the query terms
  - cosine computation restricted to documents in C only
- Choice of *r* is critical for the heuristic’s efficiency

Inexact top k retrieval

Impact ordering

- Posting lists so far always had a common ordering
  - Typically by document identifier
  - Required for a concurrent traversal of all the query terms' posting lists (document-at-a-time scoring)
- Impact order: sort posting lists by decreasing order of $tf_{t,D}$
  - Posting lists for different terms will be in different orders!
  - Stop traversal of posting list for t after having seen r documents of tf drops below a fixed threshold
  - Query terms are traversed according to decreasing idf so that terms contributing most to the $S(Q,D)$ score are considered first
    - Adaptive (ignore insignificant query terms)

Inexact top k retrieval

Cluster pruning

- Offline: compute clusters of document vectors

① Select $\sqrt{N}$ documents at random (set S)
② For all other documents, compute their nearest neighbour in S; form clusters

- At query time, process documents from a small number of clusters
  - Given Q, find nearest $S_i \in S$
  - Candidate set C consists of the S_i cluster
  - Compute cosine scores for documents in C only

Inexact top *k* retrieval

Tiered indices

tier 1 ($tf > 10$)

Dantes: 1 → 7 → 18 → 43

Albert: 4 → 7 → 96

tier 2 ($tf > 5$)

Dantes: 17 → 33

Albert: 60 → 61

tier 3 (*rest*)

Dantes: 60

Albert: 54 → 82

Summary

- Vector space model
  - A classic IR retrieval model
- Evaluation
  - Average precision
- Effective retrieval

Sources

① Introduction to Information Retrieval. Manning et al. 2008.
② Pivoted Document Length Normalization. Singhal et al. 1996.
③ Information retrieval. Keith van Rijsbergen. 1979.
④ A vector space model for automatic indexing. Salton et al. 1975.
⑤ Managing gigabytes, Witten et al. 1999.
⑥ Relevance: a review of the literature and a framework for thinking on the notion of information science. Part II: nature and manifestations of relevance. Tefko Saracevic. 2007.