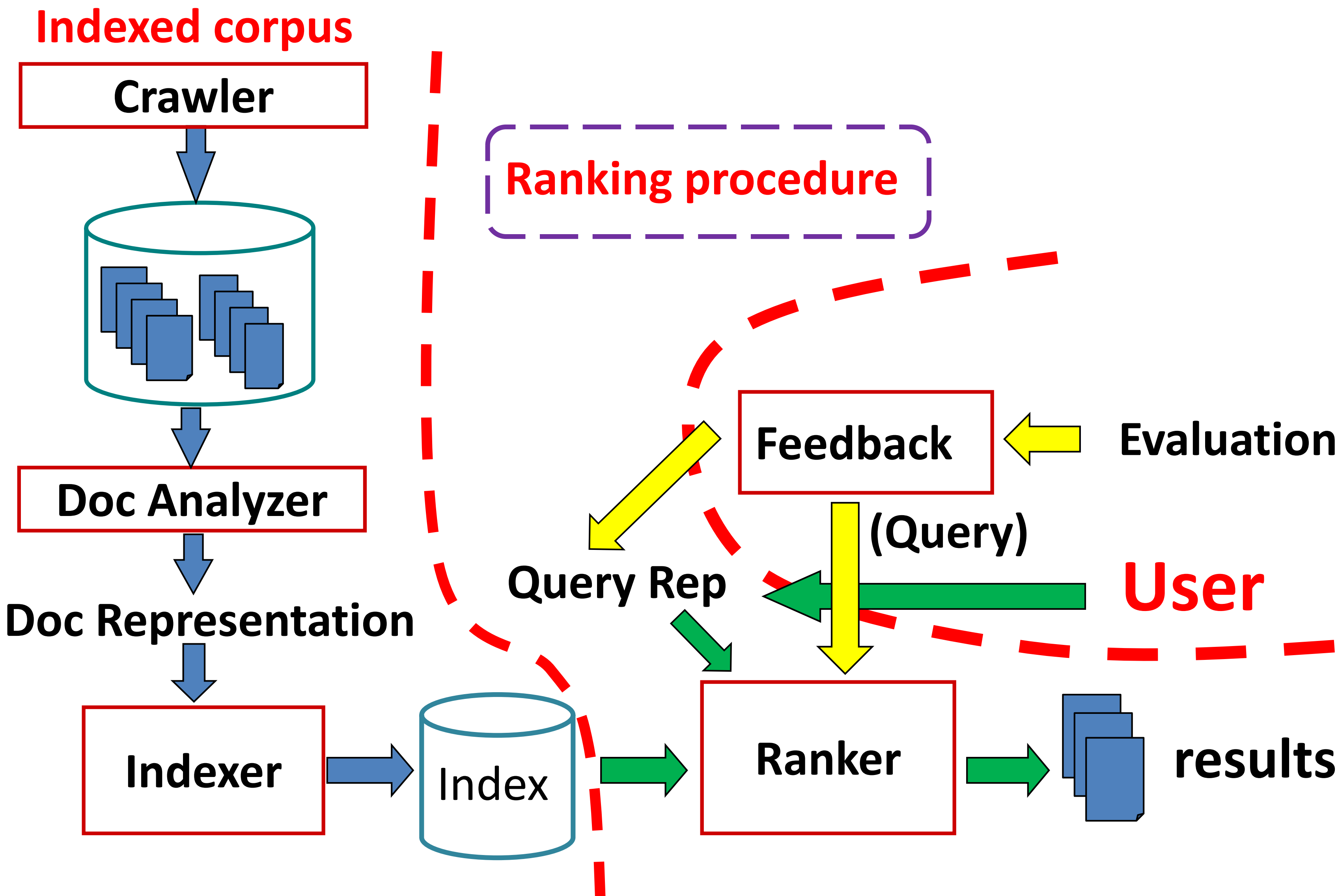
Boolean Model

Hongning Wang
CS@UVa

Abstraction of search engine architecture

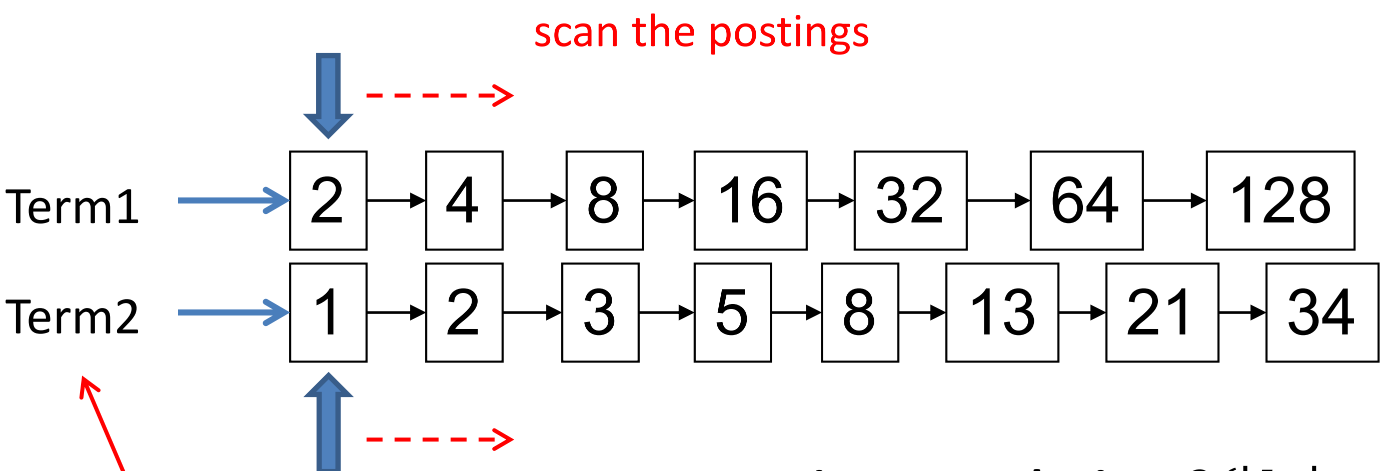


Search with Boolean query

- Boolean query
  - E.g., “obama” AND “healthcare” NOT “news”
- Procedures
  - Lookup query term in the dictionary
  - Retrieve the posting lists
  - Operation
    - AND: intersect the posting lists
    - OR: union the posting list
    - NOT: diff the posting list

Search with Boolean query

- Example: AND operation



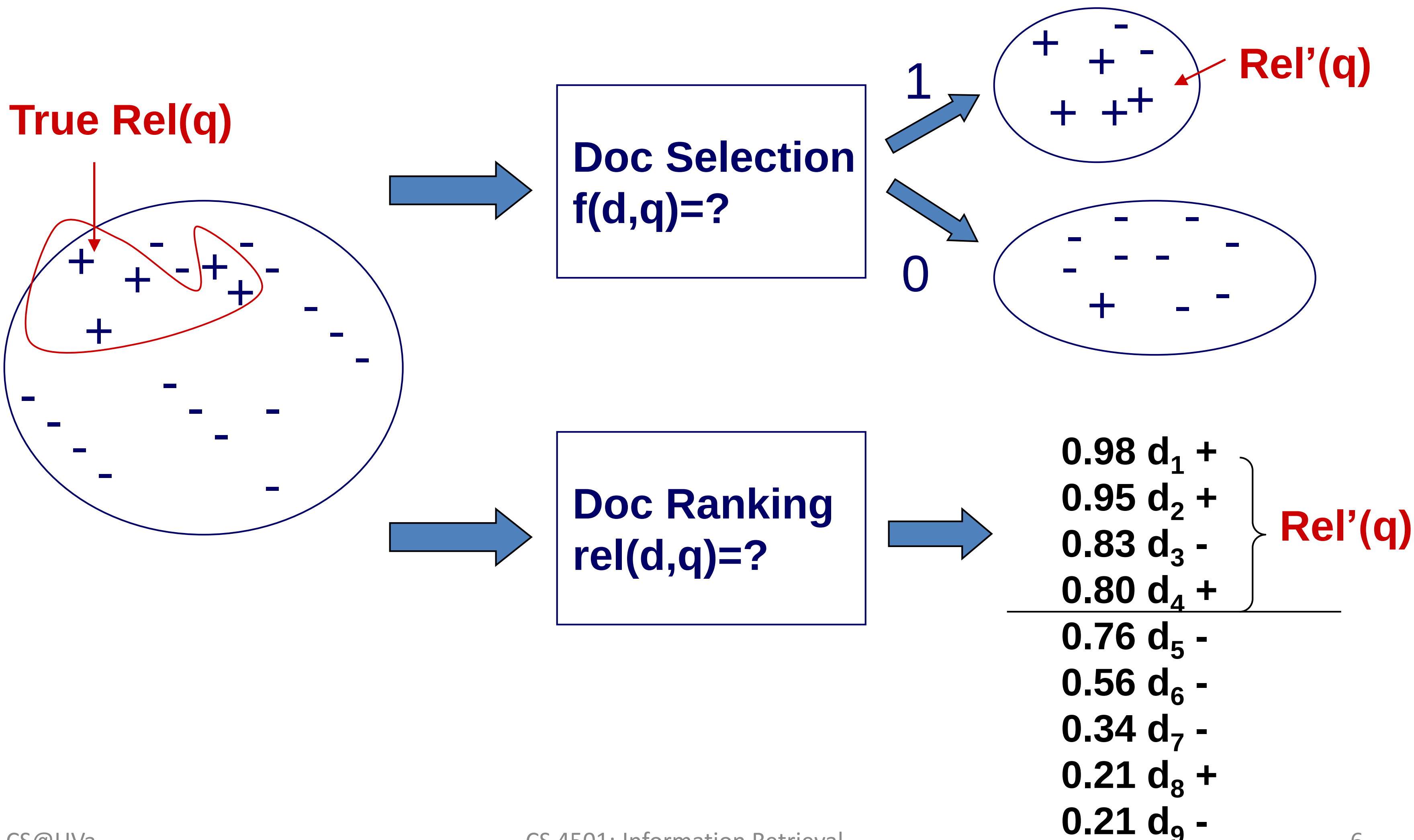
Time complexity: $O(|L_1| + |L_2|)$

Trick for speed-up: when performing multi-way join, starts from lowest frequency term to highest frequency ones

Deficiency of Boolean model

- The query is unlikely precise
  - "Over-constrained" query (terms are too specific): no relevant documents can be found
  - "Under-constrained" query (terms are too general): over delivery
  - It is hard to find the right position between these two extremes (hard for users to specify constraints)
- Even if it is accurate
  - Not all users would like to use such queries
  - All relevant documents are **not equally** relevant
    - No one would go through all the matched results
- Relevance is a matter of degree!

Document Selection vs. Ranking



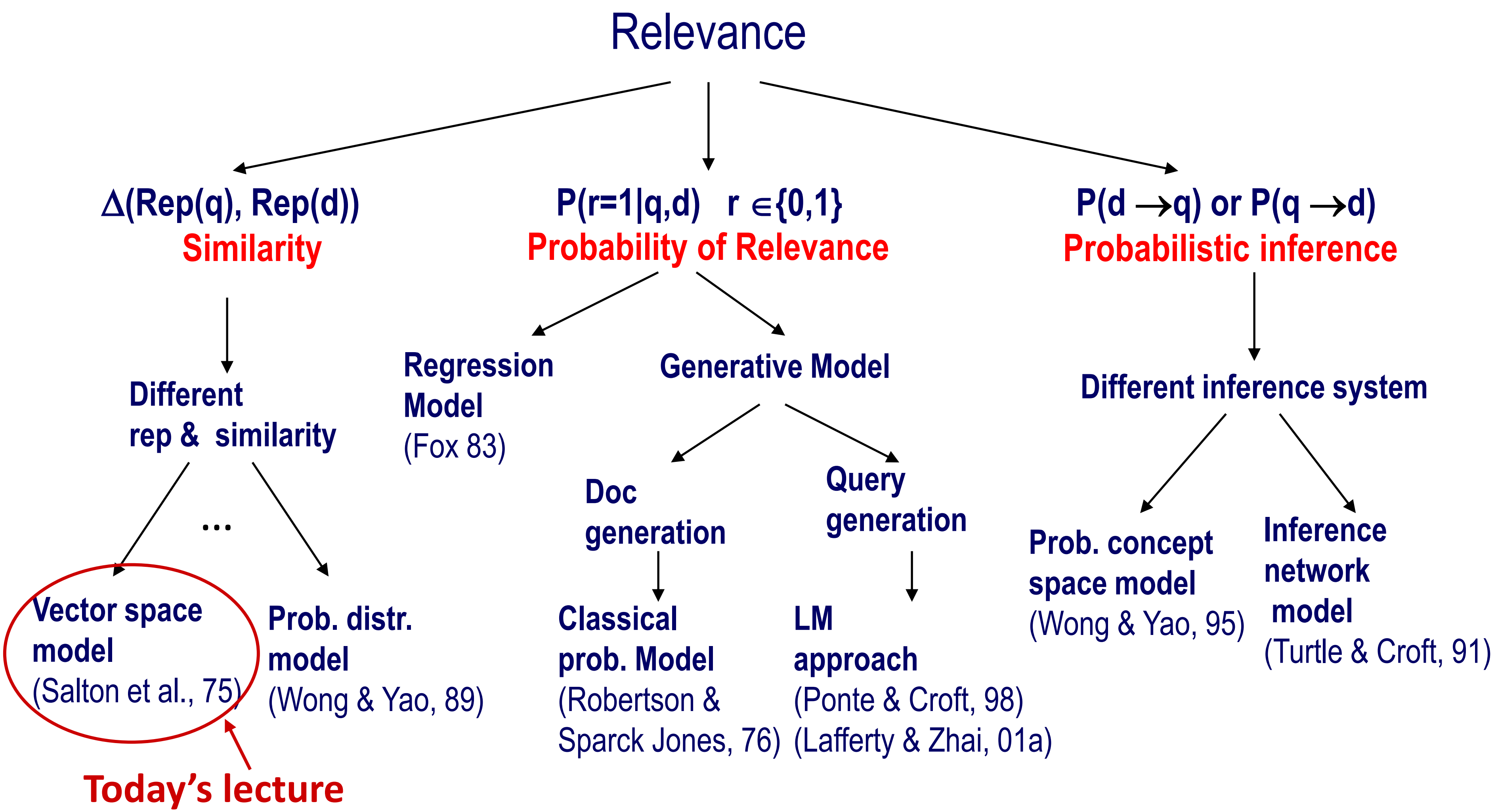
Ranking is often preferred

- Relevance is a matter of degree
  - Easier for users to find appropriate queries
- A user can stop browsing anywhere, so the boundary is controlled by the user
  - Users prefer coverage would view more items
  - Users prefer precision would view only a few
- Theoretical justification: Probability Ranking Principle

Retrieval procedure in modern IR

- Boolean model provides all the ranking candidates
  - Locate documents satisfying Boolean condition
    - E.g., “obama healthcare” -> “obama” OR “healthcare”
- Rank candidates by relevance
  - Important: the notation of relevance
- Efficiency consideration
  - Top-k retrieval (Google)

Notion of relevance



Intuitive understanding of relevance

- Fill in magic numbers to describe the relation between documents and words

	information	retrieval	retrieved	is	helpful	for	you	everyone
Doc1	1	1	0	1	1	1	0	1
Doc2	1	0	1	1	1	1	1	0

E.g., 0/1 for Boolean models,
probabilities for probabilistic models

Some notations

- Vocabulary $V=\{w_1, w_2, \ldots, w_N\}$ of language
- Query $q = t_1,\ldots,t_m$, where $t_i \in V$
- Document $d_i = t_{i1},\ldots,t_{in}$, where $t_{ij} \in V$
- Collection $C= \{d_1, \ldots, d_k\}$
- Rel(q,d): relevance of doc d to query q
- Rep(d): representation of document d
- Rep(q): representation of query q

Vector Space Model

Hongning Wang
CS@UVa

Relevance = Similarity

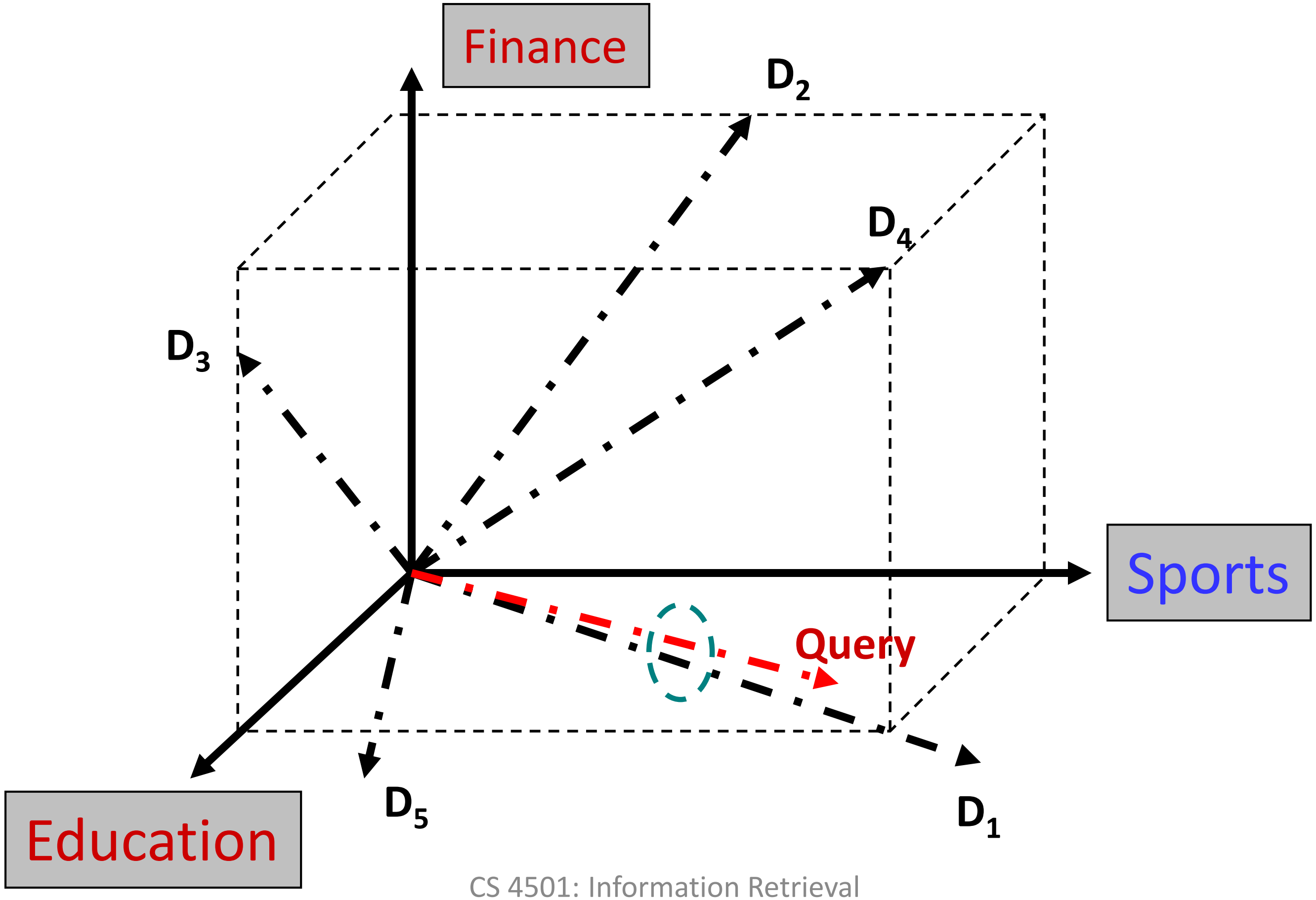
- Assumptions
  - Query and documents are represented in the same form
    - A query can be regarded as a “document”
  - Relevance(d,q) $\propto$ similarity(d,q)
- R(q) = {d$\in$C|rel(d,q)>θ}, rel(q,d)=Δ(Rep(q), Rep(d))
- Key issues
  - How to represent query/document?
  - How to define the similarity measure $\Delta(x,y)$?

Vector space model

- Represent both doc and query by <u>concept</u> vectors
  - Each concept defines one dimension
  - *K* concepts define a high-dimensional space
  - Element of vector corresponds to concept weight
    - E.g., $d=(x_1,\ldots,x_k)$, x_i is "importance" of concept i
- Measure relevance
  - Distance between the query vector and document vector in this concept space

VS Model: an illustration

- Which document is closer to the query?



What the VS model doesn't say

- How to define/select the "basic concept"
  - Concepts are assumed to be orthogonal
- How to assign weights
  - Weight in query indicates importance of the concept
  - Weight in doc indicates how well the concept characterizes the doc
- How to define the similarity/distance measure

What is a good "basic concept"?

- Orthogonal
  - Linearly independent basis vectors
    - "Non-overlapping" in meaning
    - No ambiguity
- Weights can be assigned automatically and accurately
- Existing solutions
  - Terms or N-grams, i.e., bag-of-words
  - Topics, i.e., topic model ← We will come back to this later

How to assign weights?

- Important!
- Why?
  - Query side: not all terms are equally important
  - Doc side: some terms carry more information about the content
- How?
  - Two basic heuristics
    - TF (Term Frequency) = Within-doc-frequency
    - IDF (Inverse Document Frequency)

TF weighting

- Idea: a term is more important if it occurs more frequently in a document
- TF Formulas
  - Let $f(t,d)$ be the frequency count of term t in doc d
  - Raw TF: $tf(t,d) = f(t,d)$

TF normalization

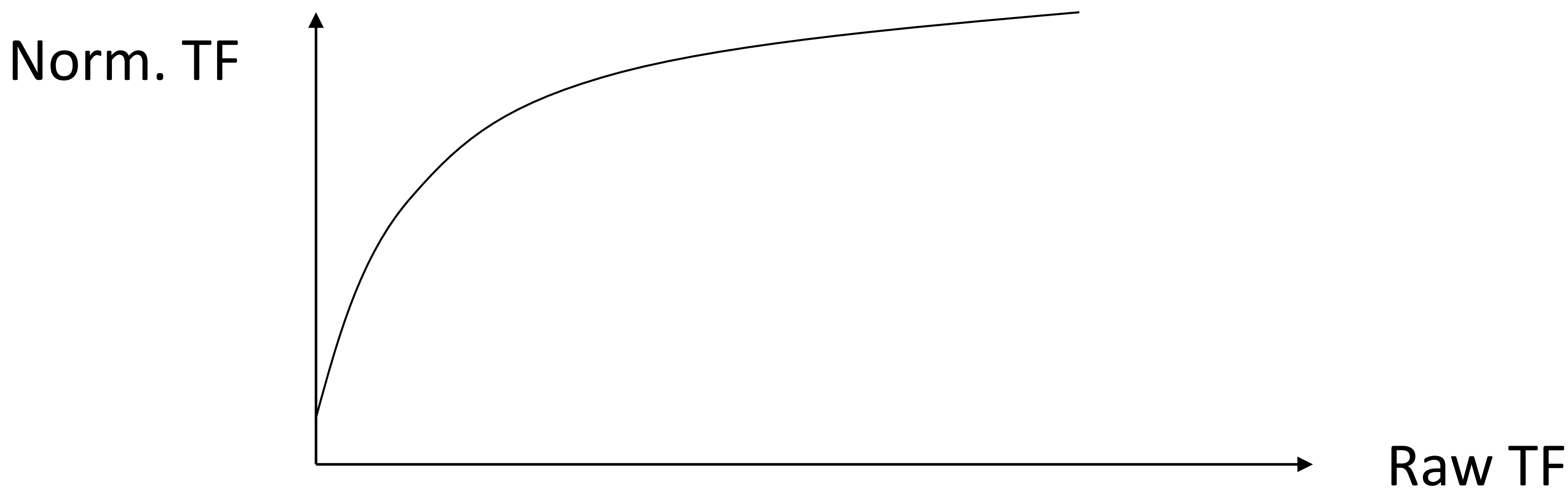
- Query: *iphone 6s*
  - D1: iPhone 6s receives pre-orders on September 12
  - D2: iPhone 6 has three color options.
  - D3: iPhone 6 has three color options. iPhone 6 has three color options. iPhone 6 has three color options.

TF normalization

- Two views of document length
  - A doc is long because it is verbose
  - A doc is long because it has more content
- Raw TF is inaccurate
  - Document length variation
  - "Repeated occurrences" are less informative than the "first occurrence"
  - Relevance does not increase proportionally with number of term occurrence
- Generally penalize long doc, but avoid over-penalizing
  - Pivoted length normalization

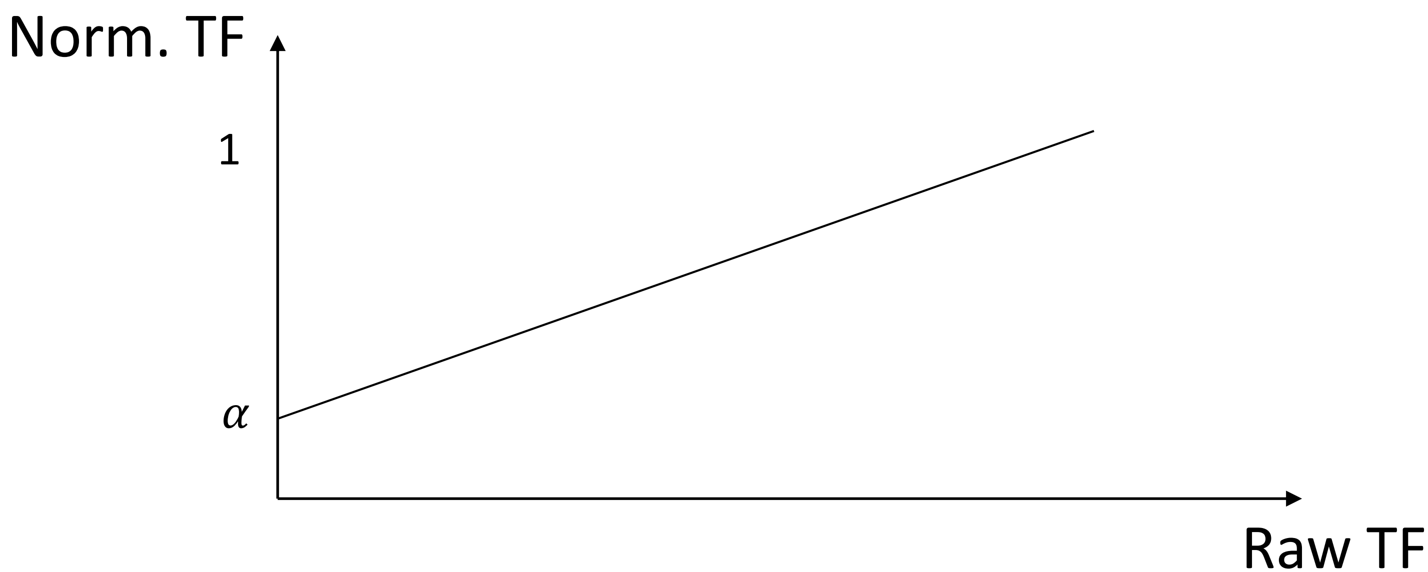
TF normalization

- Sublinear TF scaling
  - $tf(t,d) = \begin{cases} 1 + \log f(t,d), if\ f(t,d) > 0 \\ 0, otherwise \end{cases}$



TF normalization

- Maximum TF scaling
  - $tf(t,d) = \alpha + (1-\alpha)\frac{f(t,d)}{\max_t f(t,d)}$
  - Normalize by the most frequent word in this doc



Document frequency

- Idea: a term is more discriminative if it occurs only in fewer documents

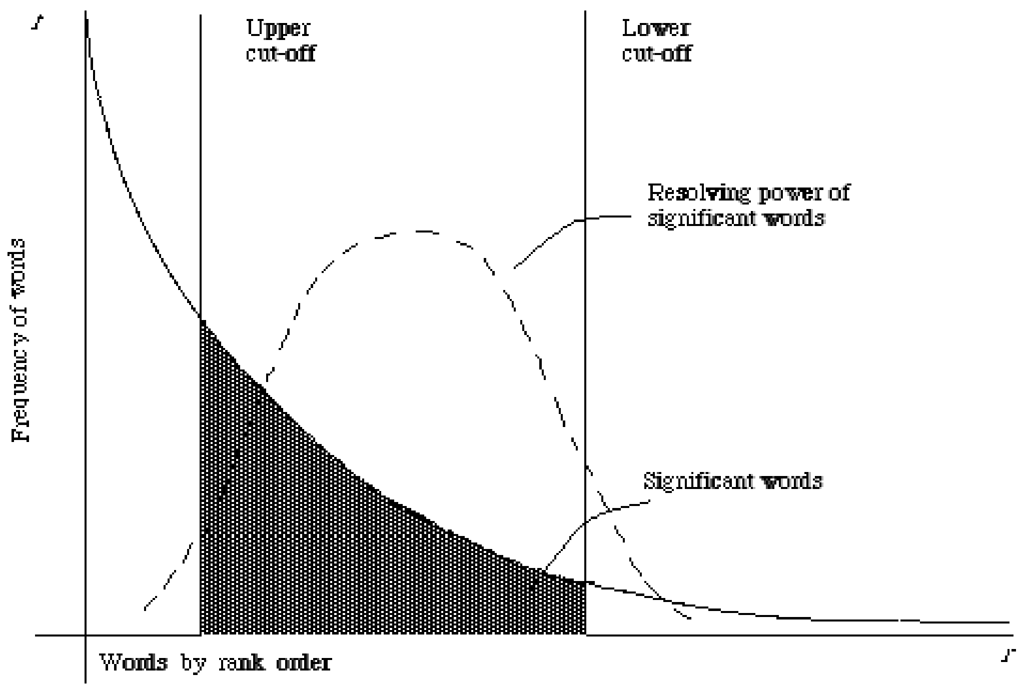


Figure 2.1. A plot of the hyperbolic curve relating f, the frequency of occurrence and r, the rank order (Adaped from Schultz[44] page 120)

IDF weighting

- Solution
  - Assign higher weights to the rare terms
  - Formula
    - $IDF(t) = 1 + \log(\frac{N}{df(t)})$

Non-linear scaling

Total number of docs in collection

Number of docs containing term t

  - A corpus-specific property
    - Independent of a single document

Why document frequency

- How about total term frequency?
  - $ttf(t) = \sum_d f(t,d)$

Table 1. Example total term frequency v.s. document frequency in Reuters-RCV1 collection.

Word	ttf	df
try	10422	8760
insurance	10440	3997

  - Cannot recognize words frequently occurring in a subset of documents

TF-IDF weighting

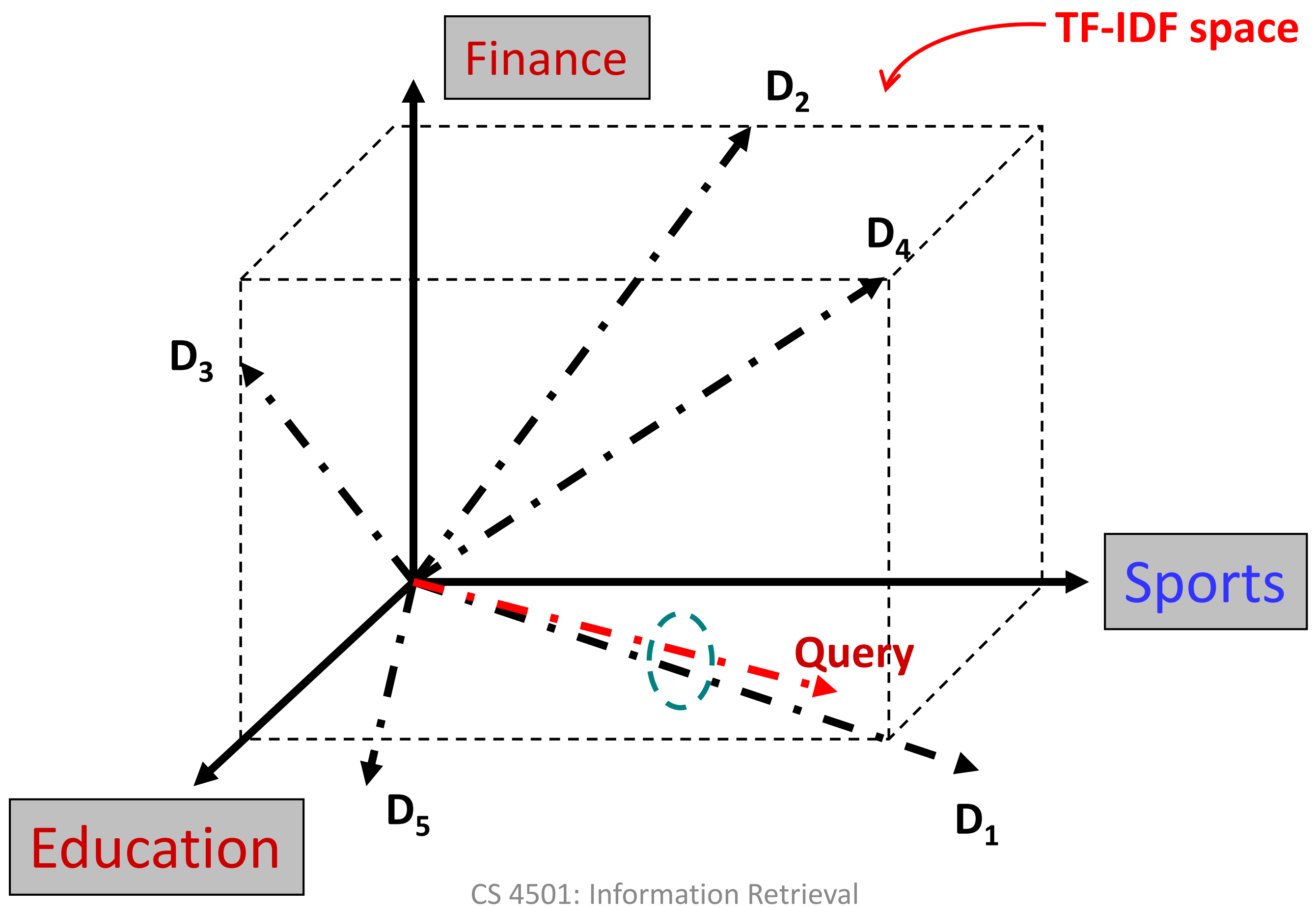
- Combining TF and IDF
  - Common in doc → high tf → high weight
  - Rare in collection→ high idf→ high weight
  - $w(t,d) = TF(t,d) \times IDF(t)$
- Most well-known document representation schema in IR! (G Salton et al. 1983)

"Salton was perhaps the leading computer scientist working in the field of information retrieval during his time." - wikipedia

Gerard Salton Award – highest achievement award in IR

How to define a good similarity measure?

- Euclidean distance?

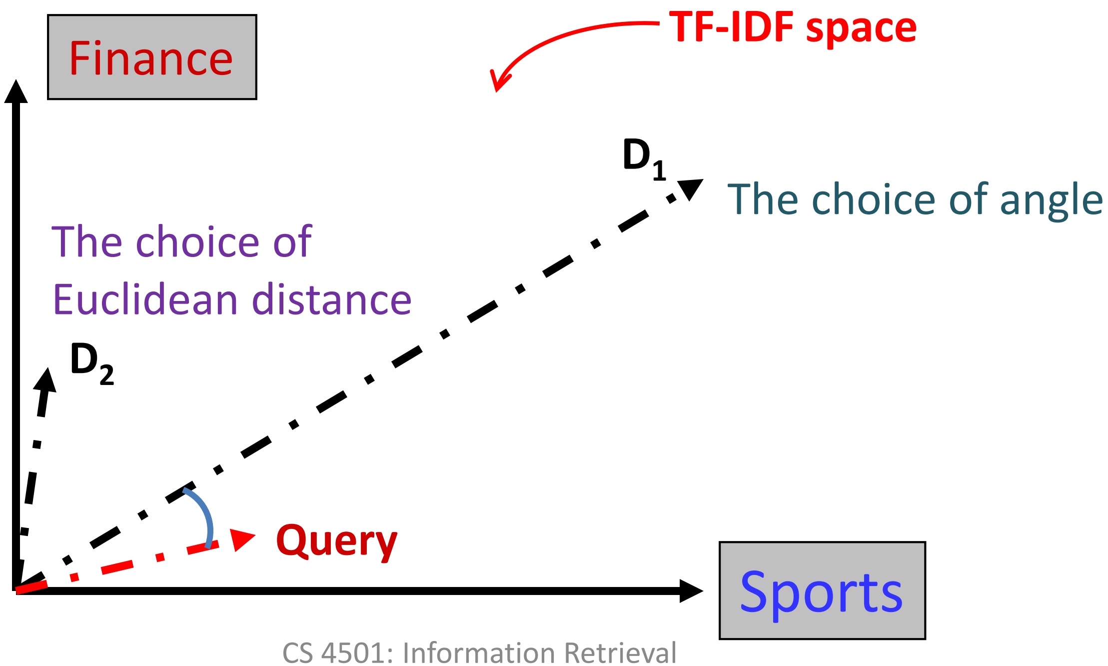


How to define a good similarity measure?

- Euclidean distance
  - $dist(q,d) = \sqrt{\sum_{t \in V} [tf(t,q) idf(t) - tf(t,d) idf(t)]^2}$
  - Longer documents will be penalized by the extra words
  - We care more about how these two vectors are overlapped

From distance to angle

- Angle: how vectors are overlapped
  - Cosine similarity – projection of one vector onto another



Cosine similarity

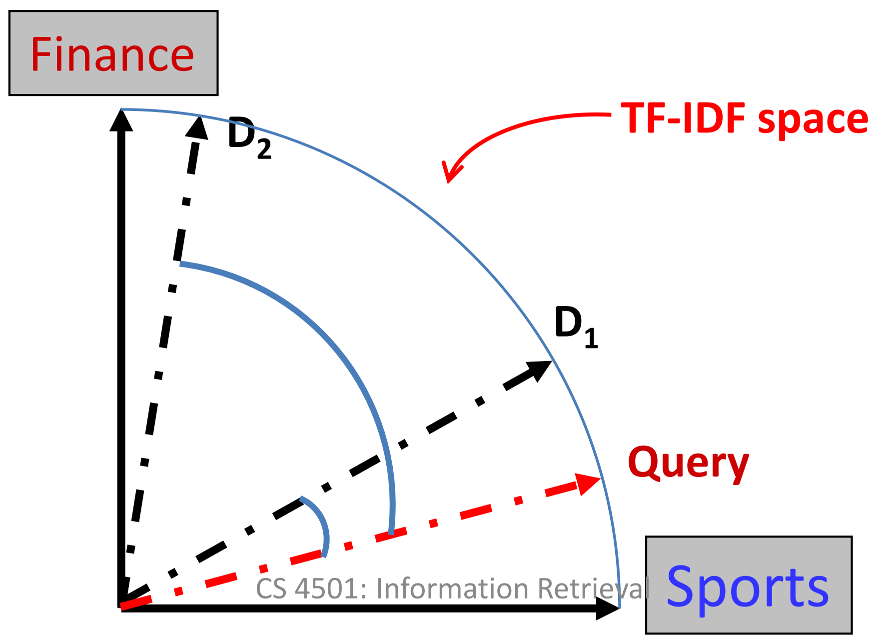
- Angle between two vectors

  TF-IDF vector

  - $cosine(V_q, V_d) = \frac{V_q \times V_d}{|V_q|_2 \times |V_d|_2} = \boxed{\frac{V_q}{|V_q|_2}} \times \frac{V_d}{|V_d|_2}$

  Unit vector

  - Document length normalized



Fast computation of cosine in retrieval

- $cosine(V_q, V_d) = V_q \times \frac{V_d}{|V_d|_2}$
  - $|V_q|_2$ would be the same for all candidate docs
  - Normalization of V_d can be done in indexing time
  - Only count $t \in q \cap d$
  - Score accumulator for each query term when intersecting postings from inverted index

Fast computation of cosine in retrieval

- Maintain a score accumulator for each doc when scanning the postings

Query = "info security"

S(d,q)=g(t_1)+…+g(t_n) [sum of TF of matched terms]

Info: (d1, 3), (d2, 4), (d3, 1), (d4, 5)

Security: (d2, 3), (d4, 1), (d5, 3)

Can be easily applied to TF-IDF weighting!

	Accumulators:	d1	d2	d3	d4	d5
info	(d1,3) =>	**3**	0	0	0	0
	(d2,4) =>	3	4	0	0	0
	(d3,1) =>	3	4	**1**	0	0
	(d4,5) =>	3	4	1	5	0
security	(d2,3) =>	3	**7**	1	5	0
	(d4,1) =>	3	7	1	**6**	0
	(d5,3) =>	3	7	1	6	**3**

Keep only the most promising accumulators for top *K* retrieval

Advantages of VS Model

- Empirically effective! (Top TREC performance)
- Intuitive
- Easy to implement
- Well-studied/Mostly evaluated
- The Smart system
  - Developed at Cornell: 1960-1999
  - Still widely used
- Warning: Many variants of TF-IDF!

Disadvantages of VS Model

- Assume term independence
- Assume query and document to be the same
- Lack of “predictive adequacy”
  - Arbitrary term weighting
  - Arbitrary similarity measure
- Lots of parameter tuning!

What you should know

- Document ranking v.s. selection
- Basic idea of vector space model
- Two important heuristics in VS model
  - TF
  - IDF
- Similarity measure for VS model

Today's reading

- Chapter 1: Boolean retrieval
  - 1.3 Processing Boolean queries
  - 1.4 The extended Boolean model versus ranked retrieval
- Chapter 6: Scoring, term weighting and the vector space model
  - 6.2 Term frequency and weighting
  - 6.3 The vector space model for scoring
  - 6.4 Variant tf-idf functions