Introduction to

Information Retrieval

Probabilistic Information Retrieval

Christopher Manning and Pandu Nayak

FROM BOOLEAN TO RANKED RETRIEVAL … IN TWO STEPS

Ranked retrieval

- Thus far, our queries have all been Boolean
  - Documents either match or don't
- Can be good for expert users with precise understanding of their needs and the collection
  - Can also be good for applications: Applications can easily consume 1000s of results
- Not good for the majority of users
  - Most users incapable of writing Boolean queries
    - Or they are, but they think it's too much work
  - Most users don't want to wade through 1000s of results.
    - This is particularly true of web search

Problem with Boolean search: feast or famine

- Boolean queries often result in either too few (=0) or too many (1000s) results.
- Query 1: "*standard user dlink 650*" → 200,000 hits
- Query 2: "*standard user dlink 650 no card found*": 0 hits
- It takes a lot of skill to come up with a query that produces a manageable number of hits.
  - AND gives too few; OR gives too many

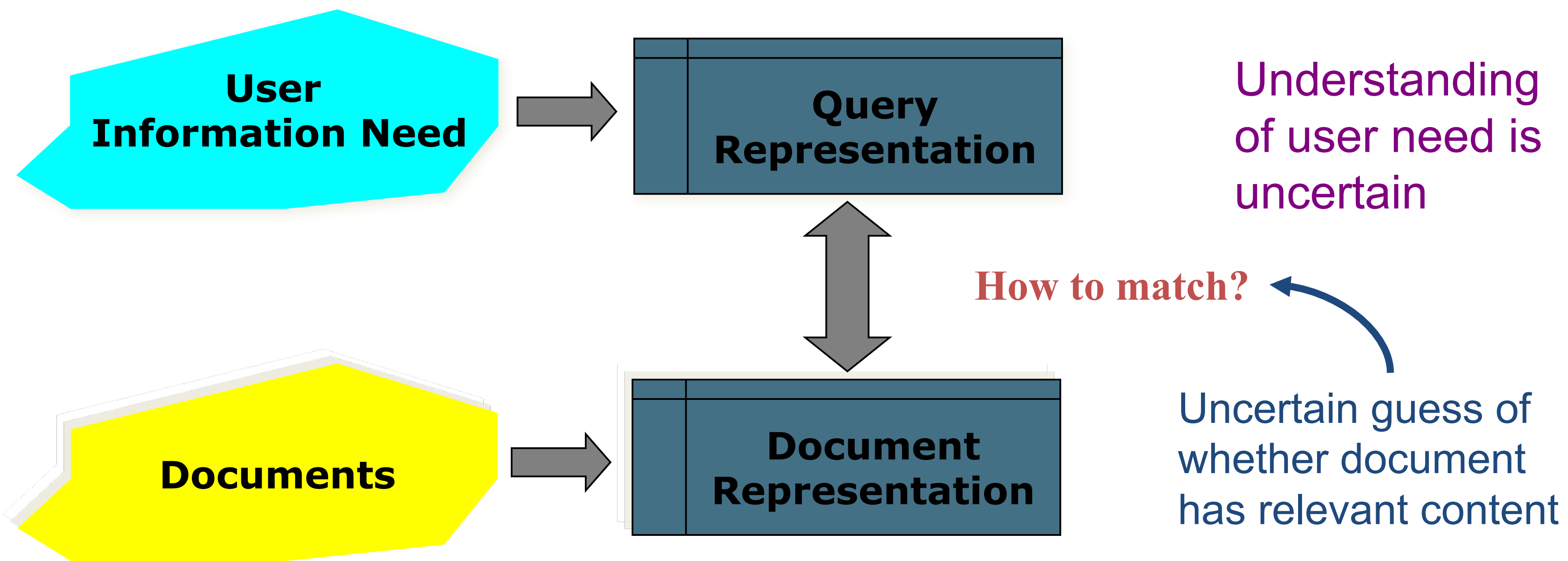
Who are these people?

Karen Spärck Jones

Stephen Robertson

Keith van Rijsbergen

Why probabilities in IR?



In traditional IR systems, matching between each document and query is attempted in a semantically imprecise space of index terms.

Probabilities provide a principled foundation for uncertain reasoning.
Can we use probabilities to quantify our uncertainties?

Probabilistic IR topics

1. Classical probabilistic retrieval model
   - Probability ranking principle, etc.
   - Binary independence model (≈ Naïve Bayes text cat)
   - (Okapi) BM25
2. Bayesian networks for text retrieval
3. Language model approach to IR
   - An important emphasis in recent work

Probabilistic methods are one of the oldest but also one of the currently hot topics in IR

- *Traditionally: neat ideas, but didn't win on performance*
- *It seems to be different now*

The document ranking problem

- We have a collection of documents
- User issues a query
- A list of documents needs to be returned
- **Ranking method is the core of modern IR systems:**
  - **In what order do we present documents to the user?**
  - We want the "best" document to be first, second best second, etc….
- **Idea: Rank by probability of relevance of the document w.r.t. information need**
  - $P(R=1|\text{document}_i, \text{query})$

Recall a few probability basics

- For events *A* and *B:*

$$p(A,B) = p(A \cap B) = p(A \mid B)p(B) = p(B \mid A)p(A)$$

- Bayes' Rule

$$p(A \mid B) = \frac{p(B \mid A)p(A)}{p(B)} = \frac{p(B \mid A)\boxed{p(A)}}{\sum_{X=A,\bar{A}} p(B \mid X)p(X)}$$

Prior

Posterior

- Odds:

$$O(A) = \frac{p(A)}{p(\bar{A})} = \frac{p(A)}{1 - p(A)}$$

The Probability Ranking Principle

"If a reference retrieval system's response to each request is a ranking of the documents in the collection in order of decreasing probability of relevance to the user who submitted the request, where the probabilities are estimated as accurately as possible on the basis of whatever data have been made available to the system for this purpose, the overall effectiveness of the system to its user will be the best that is obtainable on the basis of those data."

- [1960s/1970s] S. Robertson, W.S. Cooper, M.E. Maron; van Rijsbergen (1979:113); Manning & Schütze (1999:538)

Probability Ranking Principle

Let x represent a document in the collection.
Let R represent **relevance** of a document w.r.t. given (fixed) query and let **R=1** represent relevant and **R=0** not relevant.

Need to find $p(R=1|x)$ – probability that a document x is **relevant.**

$$p(R=1|x) = \frac{p(x|R=1)p(R=1)}{p(x)}$$

$p(R=1), p(R=0)$ - prior probability of retrieving a relevant or non-relevant document

$$p(R=0|x) = \frac{p(x|R=0)p(R=0)}{p(x)}$$

$p(x|R=1), p(x|R=0)$ - probability that if a relevant (not relevant) document is retrieved, it is x.

$$p(R=0|x) + p(R=1|x) = 1$$

Probability Ranking Principle (PRP)

- Simple case: no selection costs or other utility concerns that would differentially weight errors
- PRP in action: Rank all documents by $p(R=1|x)$
- Theorem: Using the PRP is optimal, in that it minimizes the loss (Bayes risk) under 1/0 loss
  - Provable if all probabilities correct, etc. [e.g., Ripley 1996]

Probability Ranking Principle

- More complex case: retrieval costs.
  - Let *d* be a document
  - *C* – cost of not retrieving a relevant document
  - *C'* – cost of retrieving a non-relevant document
- Probability Ranking Principle: if

$$C' \cdot p(R=0 \mid d) - C \cdot p(R=1 \mid d) \le C' \cdot p(R=0 \mid d') - C \cdot p(R=1 \mid d')$$

for all *d' not yet retrieved*, then *d* **is the next document to be retrieved**

- **We won't further consider cost/utility from now on**

Probability Ranking Principle

- How do we compute all those probabilities?
  - Do not know exact probabilities, have to use estimates
  - Binary Independence Model (BIM) – which we discuss next – is the simplest model
- Questionable assumptions
  - "Relevance" of each document is independent of relevance of other documents.
    - Really, it's bad to keep on returning **duplicates**
  - Boolean model of relevance
  - That one has a single step information need
    - Seeing a range of results might let user refine query

Probabilistic Retrieval Strategy

- Estimate how terms contribute to relevance
  - How do other things like term frequency and document length influence your judgments about document relevance?
    - Not at all in BIM
    - A more nuanced answer is the Okapi (BM25) formulae [next time]
      - Spärck Jones / Robertson
- Combine to find document relevance probability
- Order documents by decreasing probability

Probabilistic Ranking

Basic concept:

"For a given query, if we know some documents that are relevant, terms that occur in those documents should be given greater weighting in searching for other relevant documents.

By making assumptions about the distribution of terms and applying Bayes Theorem, it is possible to derive weights theoretically."

Van Rijsbergen

Binary Independence Model

- Traditionally used in conjunction with PRP
- **“Binary” = Boolean**: documents are represented as binary incidence vectors of terms (cf. IIR Chapter 1):
  - $\vec{x} = (x_1, \ldots, x_n)$
  - $x_i = 1$ iff term *i* is present in document *x*.
- **“Independence”:** terms occur in documents independently
- Different documents can be modeled as the same vector

Binary Independence Model

- Queries: binary term incidence vectors
- Given query ***q***,
  - for each document ***d*** need to compute ***p(R|q,d)***.
  - replace with computing ***p(R|q,x)*** where ***x*** is binary term incidence vector representing ***d***.
  - Interested only in ranking
- Will use odds and Bayes' Rule:

$$O(R \mid q,\vec{x}) = \frac{p(R=1 \mid q,\vec{x})}{p(R=0 \mid q,\vec{x})} = \frac{\dfrac{p(R=1 \mid q)\,p(\vec{x} \mid R=1,q)}{p(\vec{x} \mid q)}}{\dfrac{p(R=0 \mid q)\,p(\vec{x} \mid R=0,q)}{p(\vec{x} \mid q)}}$$

Binary Independence Model

$$O(R \mid q, \vec{x}) = \frac{p(R=1 \mid q, \vec{x})}{p(R=0 \mid q, \vec{x})} = \frac{p(R=1 \mid q)}{p(R=0 \mid q)} \cdot \frac{p(\vec{x} \mid R=1, q)}{p(\vec{x} \mid R=0, q)}$$

Constant for a given query

Needs estimation

- Using **Independence** Assumption:

$$\frac{p(\vec{x} \mid R=1, q)}{p(\vec{x} \mid R=0, q)} = \prod_{i=1}^{n} \frac{p(x_i \mid R=1, q)}{p(x_i \mid R=0, q)}$$

$$O(R \mid q, \vec{x}) = O(R \mid q) \cdot \prod_{i=1}^{n} \frac{p(x_i \mid R=1, q)}{p(x_i \mid R=0, q)}$$

Binary Independence Model

$$O(R \mid q, \vec{x}) = O(R \mid q) \cdot \prod_{i=1}^{n} \frac{p(x_i \mid R=1, q)}{p(x_i \mid R=0, q)}$$

- Since x_i is either 0 or 1:

$$O(R \mid q, \vec{x}) = O(R \mid q) \cdot \prod_{x_i=1} \frac{p(x_i = 1 \mid R=1, q)}{p(x_i = 1 \mid R=0, q)} \cdot \prod_{x_i=0} \frac{p(x_i = 0 \mid R=1, q)}{p(x_i = 0 \mid R=0, q)}$$

- Let $p_i = p(x_i = 1 \mid R=1, q); \;\; r_i = p(x_i = 1 \mid R=0, q);$

- Assume, for all terms not occurring in the query $(q_i=0)$ $p_i = r_i$

$$O(R \mid q, \vec{x}) = O(R \mid q) \cdot \prod_{\substack{x_i=1 \\ q_i=1}} \frac{p_i}{r_i} \cdot \prod_{\substack{x_i=0 \\ q_i=1}} \frac{(1-p_i)}{(1-r_i)}$$

	document	relevant (R=1)	not relevant (R=0)
term present	$x_i = 1$	p_i	r_i
term absent	$x_i = 0$	$(1 - p_i)$	$(1 - r_i)$

Binary Independence Model

$$O(R \mid q, \vec{x}) = O(R \mid q) \cdot \prod_{x_i = q_i = 1} \frac{p_i}{r_i} \cdot \prod_{\substack{x_i = 0 \\ q_i = 1}} \frac{1 - p_i}{1 - r_i}$$

All matching terms

Non-matching query terms

$$O(R \mid q, \vec{x}) = O(R \mid q) \cdot \prod_{\substack{x_i = 1 \\ q_i = 1}} \frac{p_i}{r_i} \cdot \prod_{\substack{x_i = 1 \\ q_i = 1}} \left(\frac{1 - r_i}{1 - p_i} \cdot \frac{1 - p_i}{1 - r_i} \right) \prod_{\substack{x_i = 0 \\ q_i = 1}} \frac{1 - p_i}{1 - r_i}$$

$$O(R \mid q, \vec{x}) = O(R \mid q) \cdot \prod_{x_i = q_i = 1} \frac{p_i (1 - r_i)}{r_i (1 - p_i)} \cdot \prod_{q_i = 1} \frac{1 - p_i}{1 - r_i}$$

All matching terms

All query terms

Binary Independence Model

$$O(R \mid q, \vec{x}) = \boxed{O(R \mid q)} \cdot \boxed{\prod_{x_i = q_i = 1} \frac{p_i(1-r_i)}{r_i(1-p_i)}} \cdot \boxed{\prod_{q_i = 1} \frac{1-p_i}{1-r_i}}$$

Constant for each query

Only quantity to be estimated for rankings

Retrieval Status Value:

$$RSV = \log \prod_{x_i = q_i = 1} \frac{p_i(1-r_i)}{r_i(1-p_i)} = \sum_{x_i = q_i = 1} \log \frac{p_i(1-r_i)}{r_i(1-p_i)}$$

Binary Independence Model

All boils down to computing RSV.

$$RSV = \log \prod_{x_i = q_i = 1} \frac{p_i(1-r_i)}{r_i(1-p_i)} = \sum_{x_i = q_i = 1} \log \frac{p_i(1-r_i)}{r_i(1-p_i)}$$

$$RSV = \sum_{x_i = q_i = 1} c_i; \qquad c_i = \log \frac{p_i(1-r_i)}{r_i(1-p_i)}$$

The c_i are log odds ratios

They function as the term weights in this model

So, how do we compute c_i's from our data ?

Binary Independence Model

- Estimating RSV coefficients in theory
- For each term i look at this table of document counts:

$$c_i = \log \frac{p_i(1-r_i)}{r_i(1-p_i)}$$

Documents	Relevant	Non-Relevant	Total
$x_i=1$	s	$n-s$	n
$x_i=0$	$S-s$	$N-n-S+s$	$N-n$
Total	S	$N-S$	N

- Estimates: $p_i \approx \frac{s}{S}$ $\quad r_i \approx \frac{(n-s)}{(N-S)}$

$$c_i \approx K(N,n,S,s) = \log \frac{s/(S-s)}{(n-s)/(N-n-S+s)}$$

For now, assume no zero terms. Remember smoothing.

Estimation – key challenge

- If non-relevant documents are approximated by the whole collection, then r_i (prob. of occurrence in non-relevant documents for query) *is n/N* and

$$\log \frac{1-r_i}{r_i} = \log \frac{N-n-S+s}{n-s} \approx \log \frac{N-n}{n} \approx \log \frac{N}{n} = IDF!$$

Collection vs. Document frequency

- Collection frequency of *t* is the total number of occurrences of *t* in the collection (incl. multiples)
- Document frequency is number of docs *t* is in
- Example:

Word	Collection frequency	Document frequency
insurance	10440	3997
try	10422	8760

- Which word is a better search term (and should get a higher weight)?

Estimation – key challenge

- p_i (probability of occurrence in relevant documents) cannot be approximated as easily
- p_i can be estimated in various ways:
  - from relevant documents if you know some
    - Relevance weighting can be used in a feedback loop
  - constant (Croft and Harper combination match) – then just get idf weighting of terms (with $p_i=0.5$)

$$RSV = \sum_{x_i=q_i=1} \log \frac{N}{n_i}$$

  - proportional to prob. of occurrence in collection
    - Greiff (SIGIR 1998) argues for 1/3 + 2/3 df_i/N

Probabilistic Relevance Feedback

1. Guess a preliminary probabilistic description of *R*=1 documents; use it to retrieve a set of documents
2. Interact with the user to refine the description: learn some definite members with *R* = 1 and *R* = 0
3. Re-estimate p_i and r_i on the basis of these
   - If *i* appears in V_i within set of documents V: $p_i = |V_i|/|V|$
   - Or can combine new information with original guess (use Bayesian prior):

$$p_i^{(2)} = \frac{|V_i| + \kappa p_i^{(1)}}{|V| + \kappa}$$

κ is prior weight

4. Repeat, thus generating a succession of approximations to relevant documents

Iteratively estimating p_i and r_i (= Pseudo-relevance feedback)

1. Assume that p_i *is* constant over all x_i in query and r_i as before
   - $p_i = 0.5$ (even odds) for any given doc
2. Determine guess of relevant document set:
   - *V* is fixed size set of highest ranked documents on this model
3. We need to improve our guesses for p_i and r_i, so
   - Use distribution of x_i in docs in V. Let V_i be set of documents containing x_i
     - $p_i = |V_i| / |V|$
   - Assume if not retrieved then not relevant
     - $r_i = (n_i - |V_i|) / (N - |V|)$
4. Go to 2. until converges then return ranking

PRP and BIM

- Getting reasonable approximations of probabilities is possible.
- Requires restrictive assumptions:
  - ***Term independence***
  - ***Terms not in query don't affect the outcome***
  - ***Boolean representation of documents/queries/relevance***
  - ***Document relevance values are independent***
- Some of these assumptions can be removed
- Problem: either require partial relevance information or seemingly only can derive somewhat inferior term weights

Removing term independence

- In general, index terms aren't independent
- Dependencies can be complex
- van Rijsbergen (1979) proposed simple model of dependencies as a tree
  - Exactly Friedman and Goldszmidt's Tree Augmented Naive Bayes (AAAI 13, 1996)
- Each term dependent on one other
- In 1970s, estimation problems held back success of this model

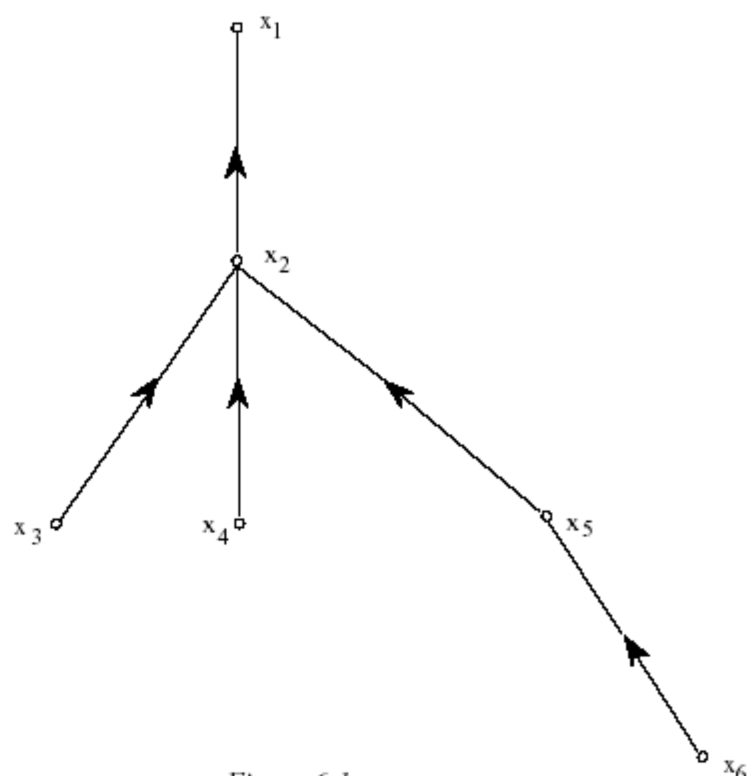


Figure 6.1.

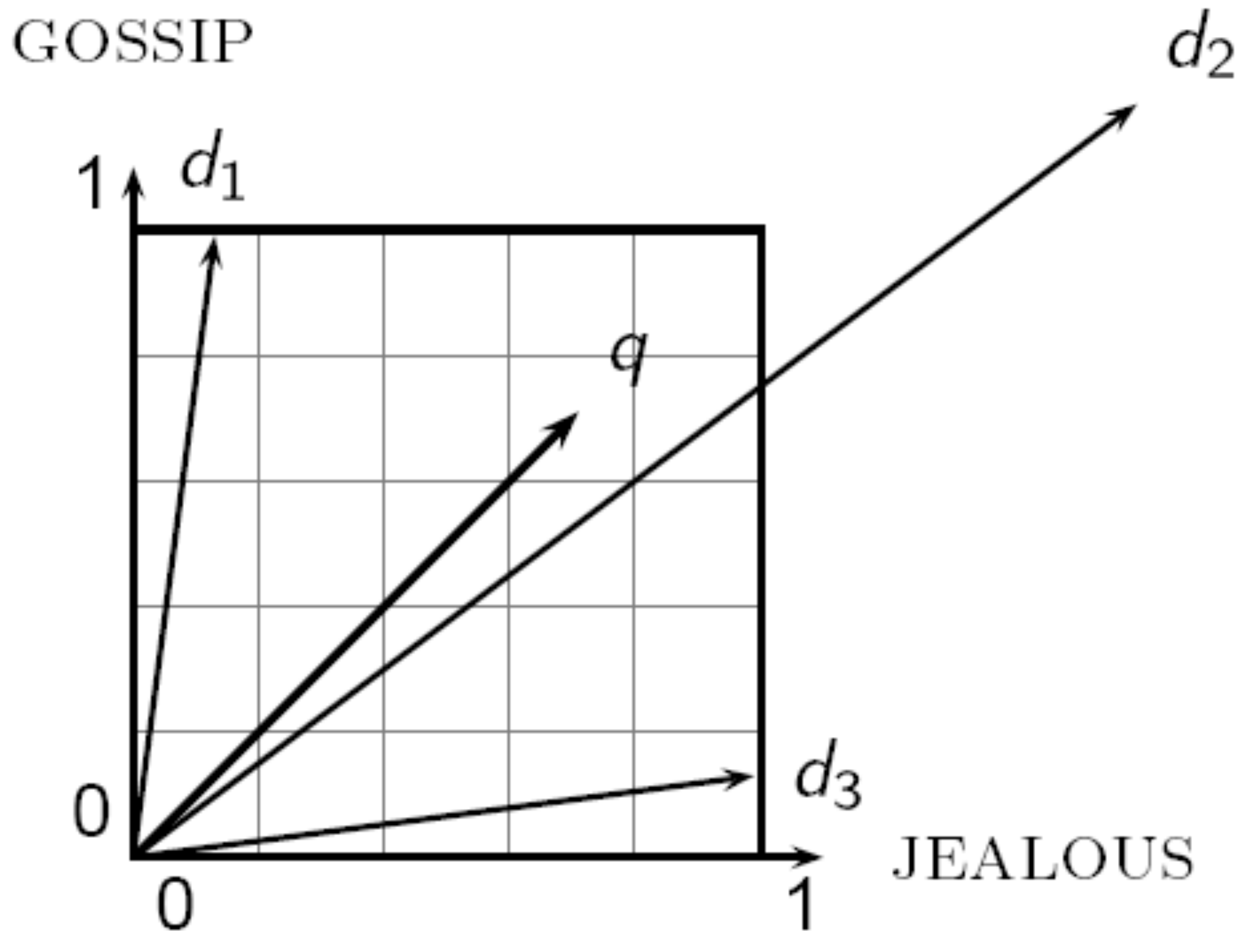
Second step: Term frequency

- Right in the first lecture, we said that a page should rank higher if it mentions a word more
  - Perhaps modulated by things like page length
- We might want a model with term frequency in it.
- We'll see a probabilistic one next time – **BM25**
- Quick summary of vector space model

Summary – vector space ranking

- Represent the query as a weighted term frequency/inverse document frequency (tf-idf) vector
- Represent each document as a weighted tf-idf vector
- Compute the cosine similarity score for the query vector and each document vector
- Rank documents with respect to the query by score
- Return the top K (e.g., $K = 10$) to the user

Cosine similarity



tf-idf weighting has many variants

Term frequency		Document frequency		Normalization	
n (natural)	$\mathrm{tf}_{t,d}$	n (no)	1	n (none)	1
l (logarithm)	$1 + \log(\mathrm{tf}_{t,d})$	t (idf)	$\log \frac{N}{\mathrm{df}_t}$	c (cosine)	$\frac{1}{\sqrt{w_1^2 + w_2^2 + \ldots + w_M^2}}$
a (augmented)	$0.5 + \frac{0.5 \times \mathrm{tf}_{t,d}}{\max_t(\mathrm{tf}_{t,d})}$	p (prob idf)	$\max\{0, \log \frac{N - \mathrm{df}_t}{\mathrm{df}_t}\}$	u (pivoted unique)	$1/u$
b (boolean)	$\begin{cases} 1 & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$			b (byte size)	$1/\mathit{CharLength}^{\alpha}$, $\alpha < 1$
L (log ave)	$\frac{1 + \log(\mathrm{tf}_{t,d})}{1 + \log(\mathrm{ave}_{t \in d}(\mathrm{tf}_{t,d}))}$				

Resources

S. E. Robertson and K. Spärck Jones. 1976. Relevance Weighting of Search Terms. *Journal of the American Society for Information Sciences* 27(3): 129–146.

C. J. van Rijsbergen. 1979. *Information Retrieval.* 2nd ed. London: Butterworths, chapter 6. [Most details of math] http://www.dcs.gla.ac.uk/Keith/Preface.html

N. Fuhr. 1992. Probabilistic Models in Information Retrieval. *The Computer Journal*, 35(3),243–255. [Easiest read, with BNs]

F. Crestani, M. Lalmas, C. J. van Rijsbergen, and I. Campbell. 1998. Is This Document Relevant? … Probably: A Survey of Probabilistic Models in Information Retrieval. *ACM Computing Surveys* 30(4): 528–552.
http://www.acm.org/pubs/citations/journals/surveys/1998-30-4/p528-crestani/
[Adds very little material that isn't in van Rijsbergen or Fuhr]