ADVANCED TOPICS IN INFORMATION RETRIEVAL AND WEB SEARCH

Lecture 3:

Overview of Traditional IR Methods

S. M. Vahidipour

Tanks Dr. Momtazi

Outline

- **Boolean model**
- Vector space model
- Probabilistic model

IR Task

- **Vocabulary**: $V = \{w_1, w_2, \ldots, w_N\}$ of language
- **Query**: $q = q_1, \ldots, q_m$ where $q_i \in V$
- **Document**: $d_i = d_{i1}, \ldots, d_{ik}$ where $d_{ij} \in V$
- **Collection**: $C = \{d_1, \ldots, d_M\}$
- Set of **relevant documents**: $R(q) \in C$
  - Generally unknown and user-dependent
  - Query is a "hint" on which doc is in $R(q)$
- **Task** = compute $R'(q)$, an approximation of $R(q)$

Boolean Model

- Two possible outcomes for query processing
  - TRUE or FALSE
  - All matching documents are considered equally relevant
- Query usually specified using Boolean operators
  - AND, OR, NOT

Example

- Search for news articles about *President Lincoln*

lincoln

Result:
cars
places
people

Example

- Search for news articles about *President Lincoln*

President AND lincoln

Result:
"Ford Motor Company today announced that Darryl Hazel will succeed Brian Kelley as president of Lincoln Mercury "

Example

- Search for news articles about *President Lincoln*

president AND Lincoln AND NOT (automobile OR car)

Not in result:
"President Lincoln's body departs Washington in a nine-car funeral train."

Example

- Search for news articles about *President Lincoln*

president AND lincoln AND biography AND life AND birthplace AND gettysburg AND NOT (automobile OR car)

Result:
∅

Example

- Search for news articles about *President Lincoln*

presidentAND lincolnAND (biography OR life OR birthplace OR gettysburg)AND NOT (automobile OR car)

Top result might be:
"President's Day - Holiday activities - crafts,mazes,mazes word searches,...'The Life ofWashington' Read the entire searchesTheWashington book online!Abraham Lincoln Research Site ..."

Boolean Model

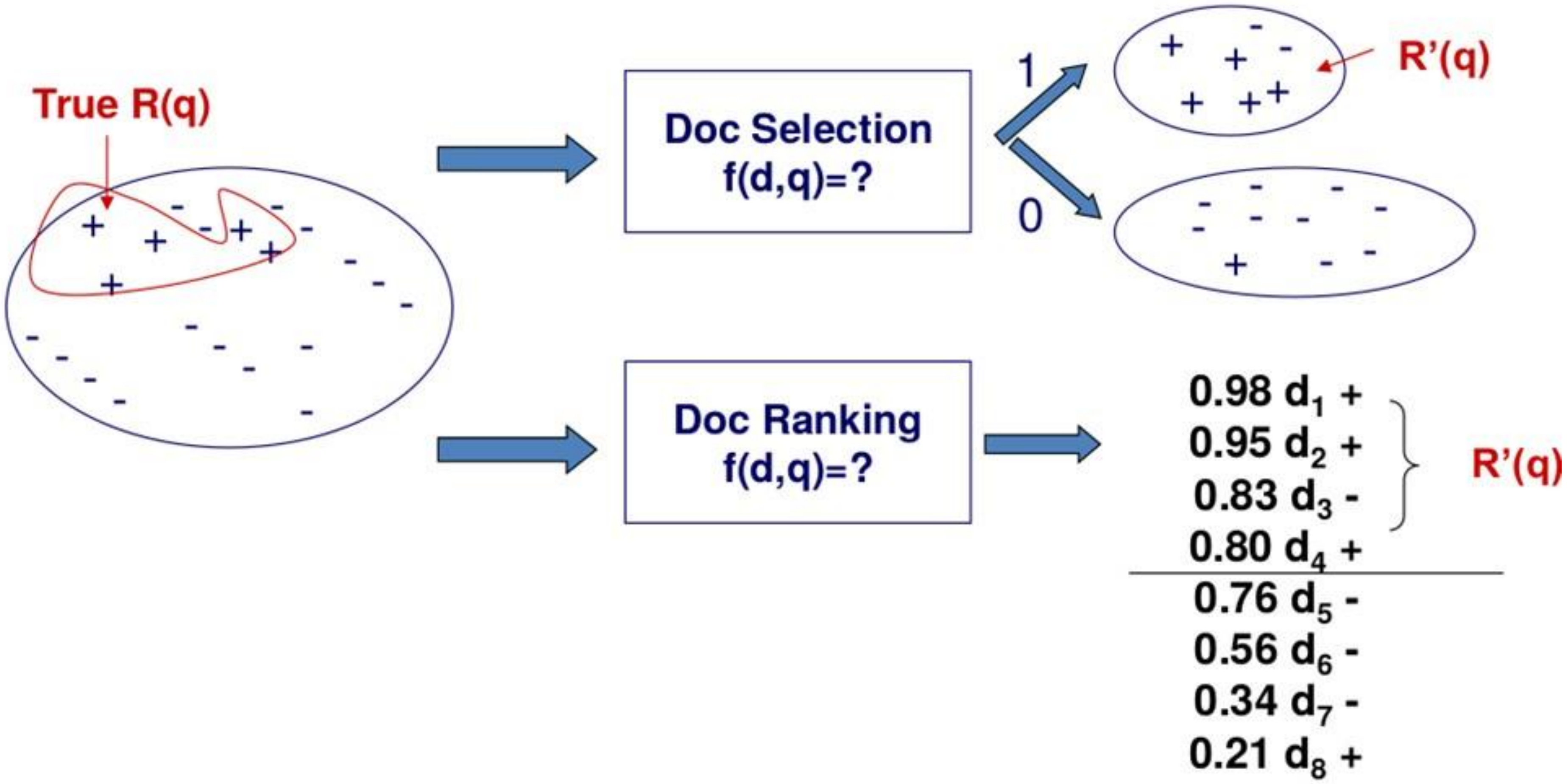
- Advantages
  - Results are predictable and relatively easy to explain
- Disadvantages
  - Relevant documents have no order
  - Complex queries are difficult to write

Document Selection vs Ranking

- Document selection
  - $R'(q) = \{d_c \,|f(d,q) = 1\}$, where $f(d,q) \in \{0,1\}$ is an indicator function or binary classifier
  - System must decide if a doc is relevant or not (**absolute relevance**)

- Document ranking
  - $R'(q) = \{d_c \,|f(d,q) > \theta\}$, where $f(d,q)$ is a relevance measure function
    - θ is a cutoff determined by the user
  - System only needs to decide if one doc is more likely relevant than another (**relative relevance**)

Document Selection vs Ranking



Document Selection Problem

- The classifier is unlikely accurate
  - “Over-constrained” query no relevant documents to return
  - “Under-constrained” query over delivery
  - Hard to find the right position between these two extremes
- Even if it is accurate, all relevant documents are not equally relevant (relevance is a matter of degree!)
  - Prioritization is needed
- Thus, ranking is generally preferred => Vector Space Model

Outline

- Boolean model
- **Vector space model**
- Probabilistic model

Ranked Retrieval

- Providing a relevance ranking of the documents with respect to a query
  - Assign a score to each query-document pair, say in [0, 1].
  - This score measures how well document and query “match”.
  - Sort documents according to scores

Vector Space Model

- Represent a doc/query by a term vector
  - Term: basic concept, e.g., word
  - Each term defines one dimension
  - N terms define an N-dimensional space
  - Query vector: $q = (x_1, \ldots, x_N)$, x_i is query term weight
  - Doc vector: $d = (y_1, \ldots, y_N)$, y_j is doc term weight
- $Relevance\ (q, d) \approx\ similarity(q, d) = f(q, d)$

Vector Space Model

- Main items in calculating scores
  - The importance of the term in query and document:
    - How many times does a query term occur in q and d? => ***Term Frequency (TF)***
  - The general importance of the term in the collection:
    - Is it a frequent or rare term? How often do we see the query term in the entire collection? => $\boldsymbol{Document\ Frequency}\ (\boldsymbol{DF})$
  - Normalization of the scores based on the length of the document:
    - How long is d? => ***Document length (|d|)***

Term Frequency (TF)

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	...
ANTHONY	1	1	0	0	0	1	
BRUTUS	1	1	0	1	0	0	
CAESAR	1	1	0	1	1	1	
CALPURNIA	0	1	0	0	0	0	
CLEOPATRA	1	0	0	0	0	0	
MERCY	1	0	1	1	1	1	
WORSER	1	0	1	1	1	0	
...							

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	...
ANTHONY	157	73	0	0	0	1	
BRUTUS	4	157	0	2	0	0	
CAESAR	232	227	0	2	1	0	
CALPURNIA	0	10	0	0	0	0	
CLEOPATRA	57	0	0	0	0	0	
MERCY	2	0	3	8	5	8	
WORSER	2	0	1	1	1	5	
...							

Term Frequency (TF)

- The term frequency $tf_{t,d}$ of term t in document d is defined as the number of times that t occurs in d.
- Using raw tf for computing query-document match scores, however, is not appropriate, because
  - A document with $tf = 10$ occurrences of the term is more relevant than a document with $tf = 1$ occurrence of the term.
  - But not 10 times more relevant.
  - i.e, Relevance does not increase proportionally with term frequency.

=> Raw Term Frequency → Log Term Frequency

$$w_{t,d} = \begin{cases} 1 + \log_{10} \mathrm{tf}_{t,d} & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$$

Document Frequency

- Rare terms are more informative than frequent terms
  - Consider a term in the query that is rare in the collection (e.g., “arachnocentric”).
  - A document containing this term is very likely to be relevant.
  - We want high weights for rare terms like “arachnocentric”
- Frequent terms are less informative than rare terms.
  - Consider a term in the query that is frequent in the collection (e.g., “good”, “increase”, “line”).
  - A document containing this term is more likely to be relevant than a document that doesn’t, but these words are not sure indicators of relevance.
  - We want positive weights for such words, but lower weights than for rare terms.

=> Using document frequency to factor this into computing the matching score.

Inverse Document Frequency (IDF)

- ❑ df_t is the document frequency, the number of documents that *t* occurs in.
- ❑ df_t is an inverse measure of the informativeness of term *t*.
- ❑ idf weight of term *t* is defined as follows:

$$idf_t = \log_{10}\left(\frac{N}{df_t}\right)$$

(*N* is the number of documents in the collection.)

[$\log_{10}\left(\frac{N}{df_t}\right)$] instead of $\left(\frac{N}{df_t}\right)$ to "dampen" the effect of *idf*

- Note: we use the log transformation for both *TF* and *IDF*

IDF Example

- Compute idf_t using the formula: $idf_t = \log_{10}(1{,}000{,}000/df_t)$

term	df_t	idf_t
calpurnia	1	6
animal	100	4
sunday	1000	3
fly	10,000	2
under	100,000	1
the	1,000,000	0

TF-IDF Weighting

- tf_idf weighting is one of the best known weighting scheme in information retrieval
- tf_idf weight of a term is the product of its tf weight and its idf weight.

$$w_{t,d} = (1 + \log tf_{t,d}).\log(N/df_t)$$

  - increases with the number of occurrences within a document. (term frequency)
  - increases with the rarity of the term in the collection. (inverse document frequency)

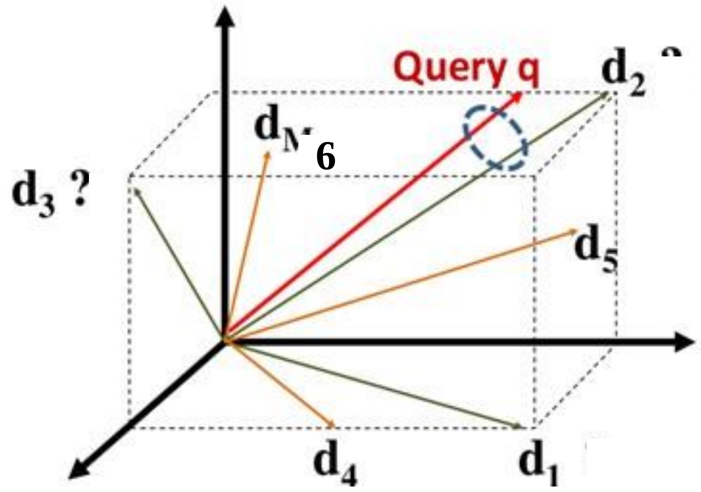
Document Normalization

- Long doc has a better chance to match any query
- Penalize a long documents with a doc length normalizer

Cosine Similarity

$$\cos(\vec{q},\vec{d}) = \text{SIM}(\vec{q},\vec{d}) = \frac{\vec{q}\cdot\vec{d}}{|\vec{q}||\vec{d}|} = \frac{\sum_{i=1}^{|V|} q_i d_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2}\sqrt{\sum_{i=1}^{|V|} d_i^2}}$$

- cosine similarity of $\vec{q}$ and $\vec{d}$
  - q_i is the tf_idf weight of term i in the query.
  - d_i is the tf_idf weight of term i in the document.
  - $|\vec{q}|$ and $|\vec{d}|$ are the lengths of $\vec{q}$ and $\vec{d}$
- Also includes doc length normalization



Vector Space Model

- Advantages
  - Simple computational framework for ranking
  - Any similarity measure or term weighting scheme can be used

- Disadvantages
  - Assumption of term independence

Outline

- Boolean model
- Vector space model
- **Probabilistic model***

* این مبحث در امتحان نخواهد بود. برای مطالعه دانشجویان گرامی قرار گرفته است

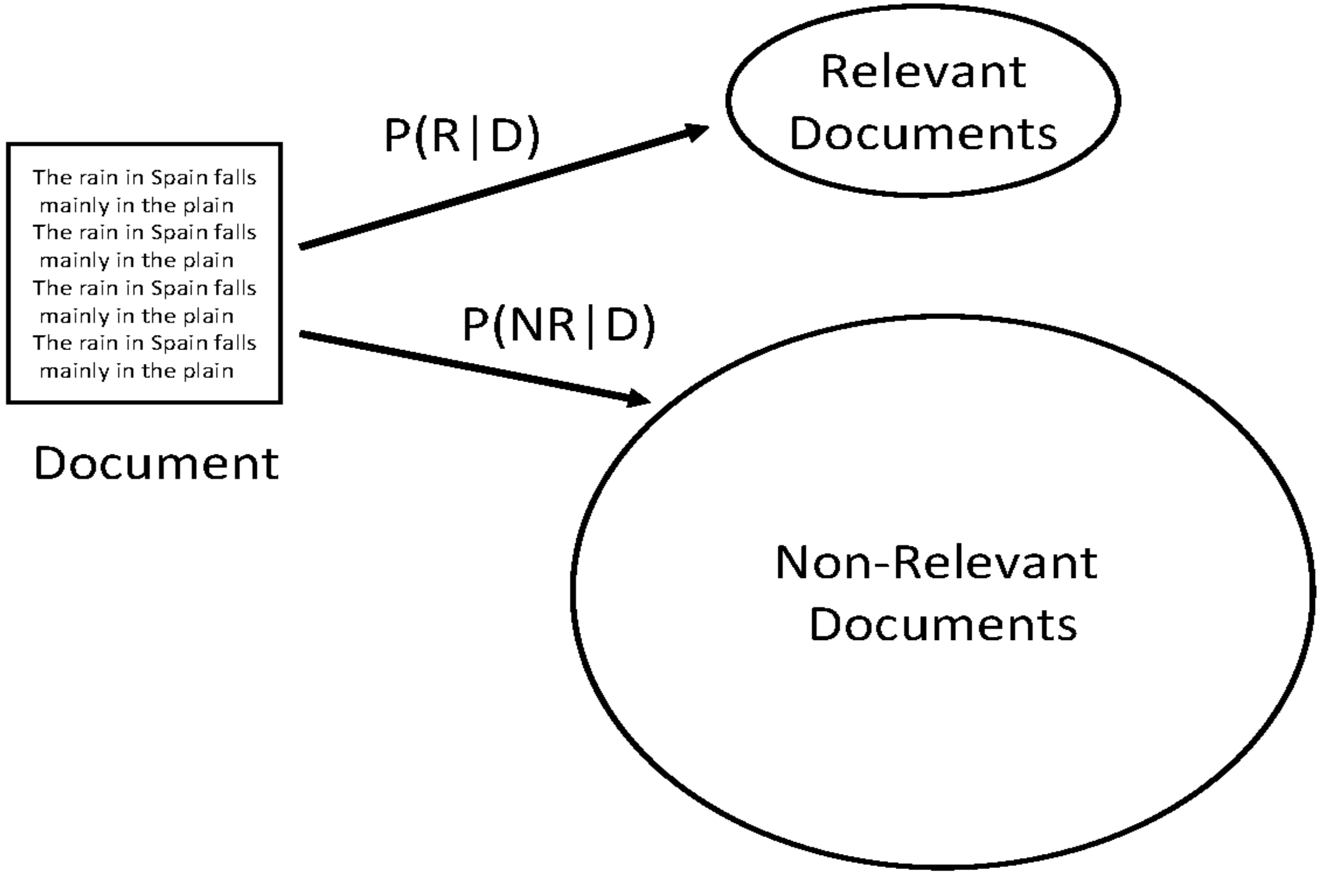
Probabilistics IR Methods

- Classical probabilistic retrieval model
  - Probability ranking principle
    - Binary Independence Model
    - BestMatch25 (Okapi)
- Bayesian networks for text retrieval
- Language model approach to IR
  - Important recent work, will be covered in the next lecture

Probabilistic vs. Other Models

- Vs. Boolean model
  - Probabilistic models support ranking and thus are better than the simple Boolean model
- Vs. Vector space model
  - The vector space model is also a formally defined model that supports ranking
  - … but it ranks documents according to similarity to query
  - The notion of similarity does not translate directly into an assessment of "is the document a good document to give to the user or not?"
  - The most similar document can be highly relevant or completely nonrelevant
  - Probability theory is arguably a cleaner formalization of what we really want an IR system to do: *give relevant documents to the user*

Document Relevance



Actually, we just need a ranking

The Document Ranking Problem

- Assume binary notion of relevance: $R_{d,q}$ is a random variable, such that
  - $R_{d,q} = 1$ if document d is relevant w.r.t query q
  - $R_{d,q} = 0$ otherwise

- Probabilistic ranking orders documents decreasingly by their estimated probability of relevance w.r.t. query: $P(R = 1|d, q)$

- Assume that the relevance of each document is independent of the relevance of other documents

Probability Ranking Principle (PRP)

- If the retrieved documents (w.r.t a query) are ranked decreasingly on their probability of relevance, then the effectiveness of the system will be the best that is obtainable

- Models
  - Binary Independence Model (BIM)
  - Best Match 25 (BM25)

Binary Independence Model (BIM)

- Assumptions:
  - “Binary” (equivalent to Boolean): documents and queries represented as binary term incidence vectors
  - “Independence”: no association between terms (not true, but practically works)
- To make a probabilistic retrieval strategy precise, need to estimate how terms in documents contribute to relevance
  - Find measurable statistics (term frequency, document frequency, document length) that affect judgments about document relevance
  - Combine these statistics to estimate the probability $P(R|d, q)$ of document relevance

Binary Independence Model (BIM)

- $P(R|d, q)$ is modeled using term incidence vectors as $P(R \mid \vec{x}, \vec{q})$

$$P(R=1 \mid \vec{x}, \vec{q}) = \frac{P(\vec{x} \mid R=1, \vec{q})P(R=1 \mid \vec{q})}{P(\vec{x} \mid \vec{q})}$$

$$P(R=1|\vec{x},\vec{q}) + P(R=0|\vec{x},\vec{q}) = 1$$

$$P(R=0 \mid \vec{x}, \vec{q}) = \frac{P(\vec{x} \mid R=0, \vec{q})P(R=0 \mid \vec{q})}{P(\vec{x} \mid \vec{q})}$$

- $P(\vec{x} \mid R=1, \vec{q})$ and $P(\vec{x} \mid R=0, \vec{q})$: probability that if a relevant or nonrelevant document is retrieved, then that document's representation is x

- $P(R=1 \mid \vec{q})$ and $P(R=0 \mid \vec{q})$: prior probability of retrieving a relevant or nonrelevant document for a query
  - Can be estimated from percentage of relevant documents in the collection

BIM Ranking (1)

- Deriving a ranking function for query terms
- Easier: rank documents by their odds of relevance (gives same ranking)

$$O(R|\vec{x},\vec{q}) = \frac{P(R=1|\vec{x},\vec{q})}{P(R=0|\vec{x},\vec{q})} = \frac{\frac{P(R=1|\vec{q})P(\vec{x}|R=1,\vec{q})}{P(\vec{x}|\vec{q})}}{\frac{P(R=0|\vec{q})P(\vec{x}|R=0,\vec{q})}{P(\vec{x}|\vec{q})}}$$

$$= \frac{P(R=1|\vec{q})}{P(R=0|\vec{q})} \cdot \frac{P(\vec{x}|R=1,\vec{q})}{P(\vec{x}|R=0,\vec{q})}$$

$O(R|\vec{q})$ (can be ignored)

BIM Ranking (2)

- Considering the conditional independence assumption: the presence or absence of a word in a document is independent of the presence or absence of any other word (given the query)

$$\frac{P(\vec{x}|R=1,\vec{q})}{P(\vec{x}|R=0,\vec{q})} = \prod_{t=1}^{M} \frac{P(x_t|R=1,\vec{q})}{P(x_t|R=0,\vec{q})}$$

Not Realistic

$$O(R|\vec{x},\vec{q}) = O(R|\vec{q}) \cdot \prod_{t=1}^{M} \frac{P(x_t|R=1,\vec{q})}{P(x_t|R=0,\vec{q})}$$

p_t

u_t

$$O(R|\vec{x},\vec{q}) = O(R|\vec{q}) \cdot \prod_{t:x_t=1} \frac{P(x_t=1|R=1,\vec{q})}{P(x_t=1|R=0,\vec{q})} \cdot \prod_{t:x_t=0} \frac{P(x_t=0|R=1,\vec{q})}{P(x_t=0|R=0,\vec{q})}$$

BIM Ranking (3)

	document	relevant $(R=1)$	nonrelevant $(R=0)$
Term present	$x_t = 1$	p_t	u_t
Term absent	$x_t = 0$	$1-p_t$	$1-u_t$

$$O(R|\vec{x},\vec{q}) = O(R|\vec{q}) \cdot \prod_{t:x_t=1} \frac{P(x_t=1|R=1,\vec{q})}{P(x_t=1|R=0,\vec{q})} \cdot \prod_{t:x_t=0} \frac{P(x_t=0|R=1,\vec{q})}{P(x_t=0|R=0,\vec{q})}$$

$$O(R|\vec{x},\vec{q}) = O(R|\vec{q}) \cdot \prod_{t:x_t=q_t=1} \frac{p_t}{u_t} \cdot \prod_{t:x_t=0,q_t=1} \frac{1-p_t}{1-u_t}$$

Over query terms found in the document

Over all query terms

$$O(R|\vec{x},\vec{q}) = O(R|\vec{q}) \cdot \prod_{t:x_t=q_t=1} \frac{p_t(1-u_t)}{u_t(1-p_t)} \cdot \prod_{t:q_t=1} \cancel{\frac{1-p_t}{1-u_t}}$$

BIM Ranking (4)

- Retrieval Status Value (RSV)
  - To avoid accuracy problems, use log

$$RSV_d = \log \prod_{t:x_t=q_t=1} \frac{p_t(1-u_t)}{u_t(1-p_t)} = \sum_{t:x_t=q_t=1} \log \frac{p_t(1-u_t)}{u_t(1-p_t)}$$

- Simplification
  - If no further information about relevant set
    - Assume p_t constant (e.g., 0.5)
    - Approximate u_t by entire collection (because number of relevant documents is very small).
    - Get idf-like weight
      - No tf-component, because binary features

the number of documents that contain term t

$$\log \frac{0.5(1-\frac{df_t}{N})}{\frac{df_t}{N}(1-0.5)} = \log \frac{N-df_t}{df_t} \approx \log \frac{N}{df_t}$$

Best Match 25 (BM25)

- Okapi BM25 is a probabilistic model that incorporates term frequency (i.e., it's nonbinary) and length normalization.
- BIM was originally designed for short catalog records of fairly consistent length, and it works reasonably in these contexts
- For modern full-text search collections, a model should pay attention to term frequency and document length
- BM25 is one of the most widely used and robust retrieval models

BM25 Ranking (1)

- The simplest score for document d is just *idf* weighting of the query terms present in the document: [log N/df]
- Improve *idf* term by factoring in term frequency and document length.

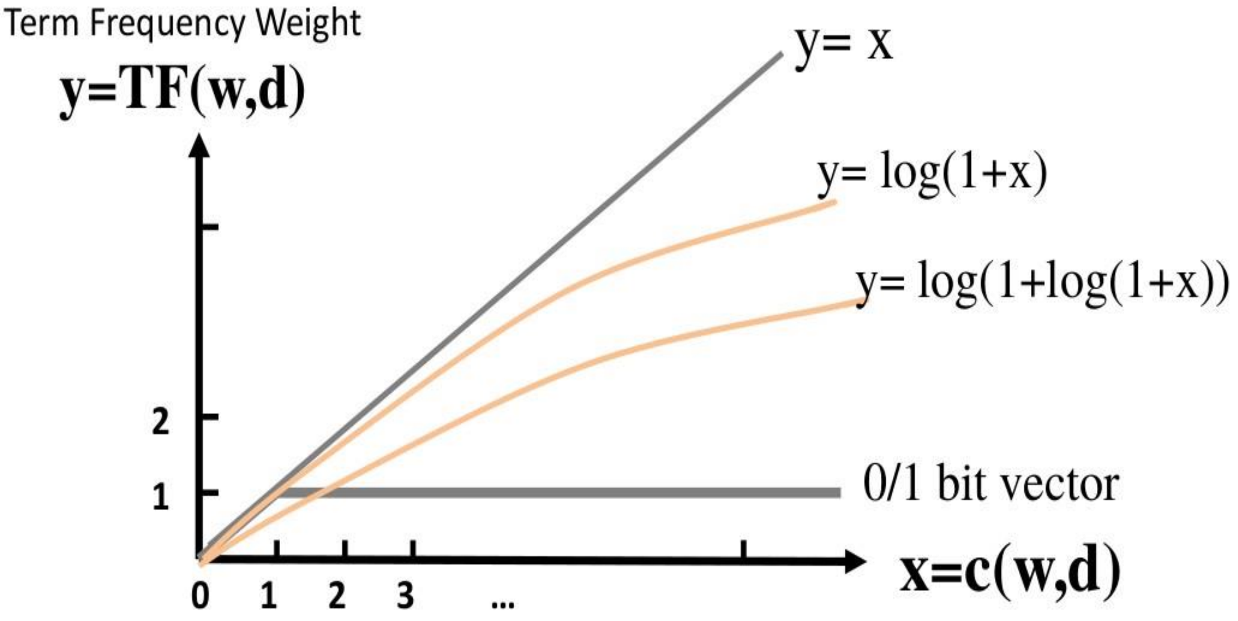
$$\sum_{t \in q} \log \left[\frac{N}{\mathrm{df}_t} \right] \cdot \frac{(k_1 + 1)\mathrm{tf}_{td}}{k_1((1-b) + b \times (L_d / L_{\mathrm{ave}})) + \mathrm{tf}_{td}}$$

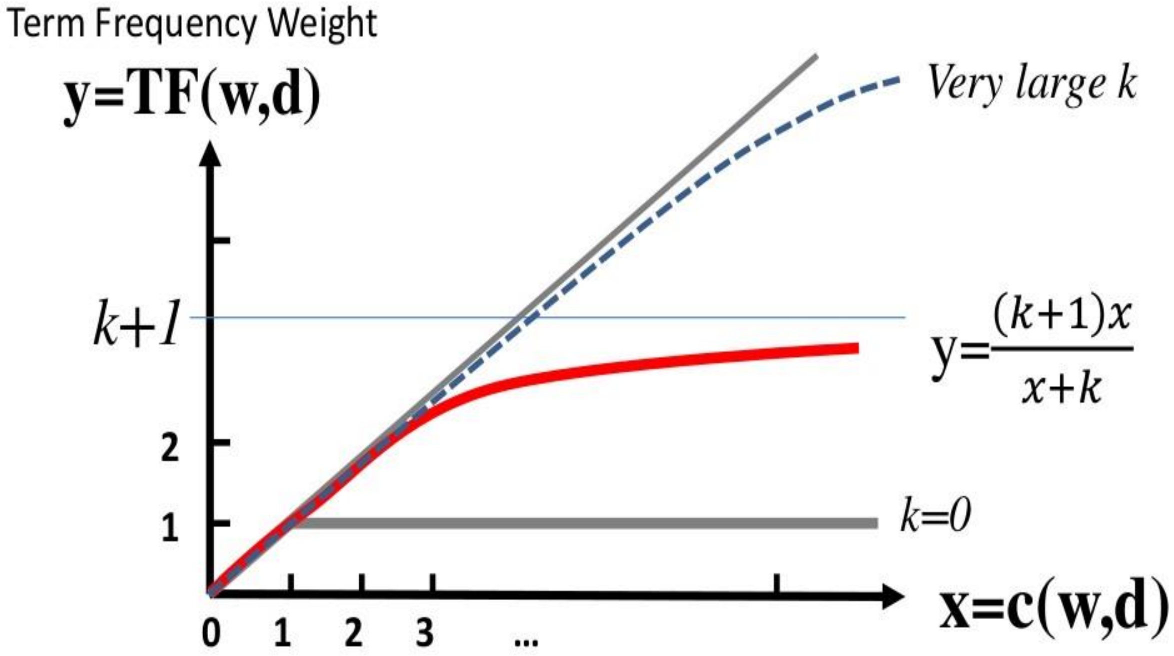
- tf_{td} : term frequency in document d
- L_d (L_{ave}): length of document d (average document length in the whole collection)
- k_1: tuning parameter controlling the document term frequency scaling
- b: tuning parameter controlling the scaling by document length

(The above tuning parameters should ideally be set to optimize performance on a development test collection. In the absence of such optimization, experiments have shown reasonable values are to set k to a value between 1.2 and 2 and b = 0.75)

BM25 Ranking (2)

$$\sum_{t \in q} \log \left[\frac{N}{\mathrm{df}_t} \right] \cdot \frac{(k_1 + 1)\mathrm{tf}_{td}}{k_1((1-b) + b \times (L_d/L_{\mathrm{ave}})) + \mathrm{tf}_{td}}$$





Questions?