Information Retrieval

Retrieval Models I: Boolean, VSM, BIRM and BM25

Qianni Zhang

Introduction: IR models

Definition: Retrieval Model?

- Provide a mathematical framework for defining the search process
  - includes explanation of assumptions
  - basis of many ranking algorithms

- Theories about relevance

Introduction: Relevance

Definition: What is Relevance?

- Complex concept that has been studied for some time
  - Many factors to consider
  - A human is not a device that reliably reports a gold standard judgment of relevance of a document to a query.
  - Humans and their relevance judgments are idiosyncratic and variable.
  - People often disagree when making relevance judgments

but

- The success of an IR system depends on how good it is at satisfying the needs of these idiosyncratic humans, one information need at a time.

Introduction: Relevance

Definition: What is Relevance?

- The relevance of one document is treated as independent of the relevance of other documents in the collection.
- Relevance of a document to an information need is treated as an absolute, objective decision.
- Judgments of relevance are subjective, varying across people
- Retrieval models make various assumptions about relevance to simplify problem
  - e.g., topical vs. user relevance
  - e.g., binary vs. multi-valued relevance

Retrieval Models I

Roadmap for this lecture

- Notations
- Components of a retrieval model

- Boolean model (recap, and a bit more mathematical)
- Vector space model (VSM)
- Binary independence retrieval model (BIRM)
- BM25 (Best-Match version 25)

Introduction: Notation reviewed

- *D: set of documents*
- *Q: set of queries*
- $d \rightarrow q$: d implies q as in classical logic
- $d \cap q$: the intersection of the set d and the set q
- $|d|$: the cardinal of the set d, i.e. the number of elements in the set d
- $d \cup q$: the union of the set d and the set q
- $a \vee b$: a or b
- $a \wedge b$: a and b

$$\sum_{i=1,n} a_i = a_1 + a_2 + ... + a_n$$

$$\prod_{i=1,n} a_i = a_1 \cdot a_2 \cdot ... \cdot a_n$$

Introduction

Components of a retrieval model

- For each retrieval model, we will make explicit the three components:
  - Document representation d
  - Query q
  - Ranking function $R(d, q)$; also score(d,q) or RSV(d,q)

Introduction

Basic Concepts

- Each document represented by a set of representative keywords or index terms
- An index term is a **word** or **group of consecutive words** in a document whose semantics is useful for remembering (summarizing) the document main themes
- However, search engines assume that all words are index terms (full text representation)

Introduction

Considerations – points of discussion:

- Correlation of index terms
  - E.g.: computer and network
  - Consideration of such correlation information does not consistently improve the final ranking result
    - Complex and slow operations
- Important Assumption/Simplification
  - Index term weights are mutually independent!
    (bag-of-words modelling)
- However, the appearance of one word often attracts the appearance of the other (e.g., “Computer” and “Network”)

Boolean model

- Simple model based on set theory and Boolean algebra
- A query is specified as boolean expressions with and, or, not operations (connectives)
  - Precise semantics, neat formalism and simplicity
  - Terms are either present or absent, i.e., {0,1}
- Retrieve documents that make the query true.

$$R(d,q) = \begin{cases} 1 \text{ if } d \rightarrow q \\ 0 \text{ otherwise} \end{cases}$$

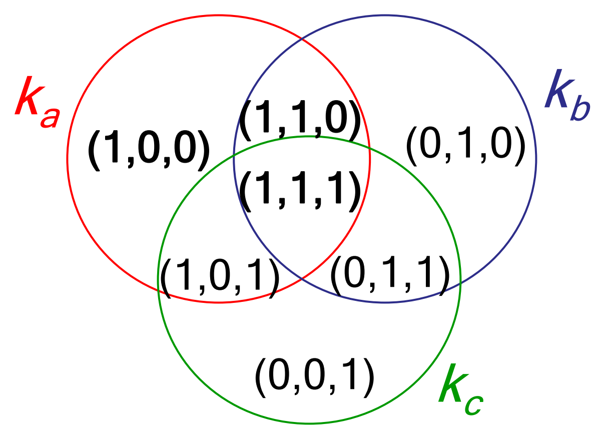
Boolean model

- Query (and document): logical combination of index terms
- A query can be expressed as a *disjunctive normal form* (DNF) composed of conjunctive components
  - $\vec{q}_{dnf}$: the DNF for a query q
  - $\vec{q}_{cc}$: conjunctive components (binary weighted vectors) of $\vec{q}_{dnf}$

- For instance, a query [$q = k_a \wedge (k_b \vee \neg k_c)$] can be written as a DNF

$$\vec{q}_{dnf} = (1,1,1) \vee (1,1,0) \vee (1,0,0)$$

Conjunctive components
(binary weighted vectors)



Boolean model

- Query (and document): logical combination of index terms
- A query can be expressed as a *disjunctive normal form* (DNF) composed of conjunctive components
  - $\vec{q}_{dnf}$: the DNF for a query q
  - $\vec{q}_{cc}$: conjunctive components (binary weighted vectors) of $\vec{q}_{dnf}$

- For instance, a query [$q = k_a \wedge (k_b \vee \neg\, k_c)$] can be written as a DNF

$$\vec{q}_{dnf} = (1,1,1) \vee (1,1,0) \vee (1,0,0)$$

Conjunctive components
(binary weighted vectors)

ka ∧ *(kb* ∨ ¬ *kc)*
=*(ka* ∧*kb)* ∨*(ka* ∧¬*kc)*
= *(ka* ∧*kb* ∧*kc)* ∨*(ka* ∧ *kb* ∧¬*kc)*
∨ *(ka* ∧*kb* ∧¬*kc)* ∨*(ka* ∧ ¬*kb* ∧ ¬*kc)*
= *(ka* ∧*kb* ∧*kc)* ∨*(ka* ∧*kb* ∧¬*kc)* ∨*(ka* ∧¬*kb* ∧¬*kc)*
=> *dnf=(1,1,1)* ∨*(1,1,0)* ∨*(1,0,0)*

Boolean model

- Example:
  - Dataset: 1 million documents (1 doc <= 1000 words)
  - Terms: typically 500,000
  - A 500K x 1M matrix with half a trillion 0's and 1's
  - >99.8% entries are 0's

- Solution: Inverted index
  - *dictionary* of terms (*vocabulary* or *lexicon*)
  - *Posting*: list that records which documents the term occurs in. Each item in the list – which records that a term appeared in a document (and, later, often, the positions in the document)

Boolean model

Boolean model

- Query (and document):
  q = (sailing ∧ boats) ∨ (bowskill ∧ ¬ south_coast)
- "Query evaluation" based on inverted file:
  - sailing = { *d1*, *d2*, d*3, d4* }
  - boats = { *d1, d2* }
  - bowskill= { *d1, d2, d3* }
  - south coast= { *d1* }

$$R(d,q) = \begin{cases} 1 \text{ if } d \to q \\ 0 \text{ otherwise} \end{cases}$$

- No ranking: either a document is retrieved or not: {*d1, d2, d3*}

Boolean model

Advantages

- Simple queries are easy to understand and relatively easy to implement (simplicity and neat model formulation)
- Results are predictable, relatively easy to explain
- Efficient processing since many documents can be eliminated from search
- The dominant language (model) in commercial (bibliographic) systems until the WWW

Boolean model

Disadvantages

- Retrieval based on binary decision criteria with no notion of partial matching (no term weighting)
  - No notion of a partial match to the query condition
  - No ranking (ordering) of the documents is provided (absence of a grading scale)
  - Term frequency counts in documents are not considered
  - Much more like a data retrieval model

Boolean model

disadvantages

- Information need has to be translated into a Boolean expression which most users find awkward
- The Boolean queries formulated by the users are most often too simplistic (difficult to specify what is wanted)
- As a consequence, the Boolean model frequently returns either too few or too many documents in response to a user query

However, the Boolean model is still dominant model with commercial document database systems

Vector space model (VSM)

Introduction

- Also called Vector Model
- Some perspectives
  - Use of binary weights is too limiting
  - Non-binary weights provide consideration for partial matches (TFIDF or other weights)
  - These term weights are used to compute a degree of similarity between a query and each document
  - Ranked set of documents provides better matching for user information need

Vector space model (VSM)

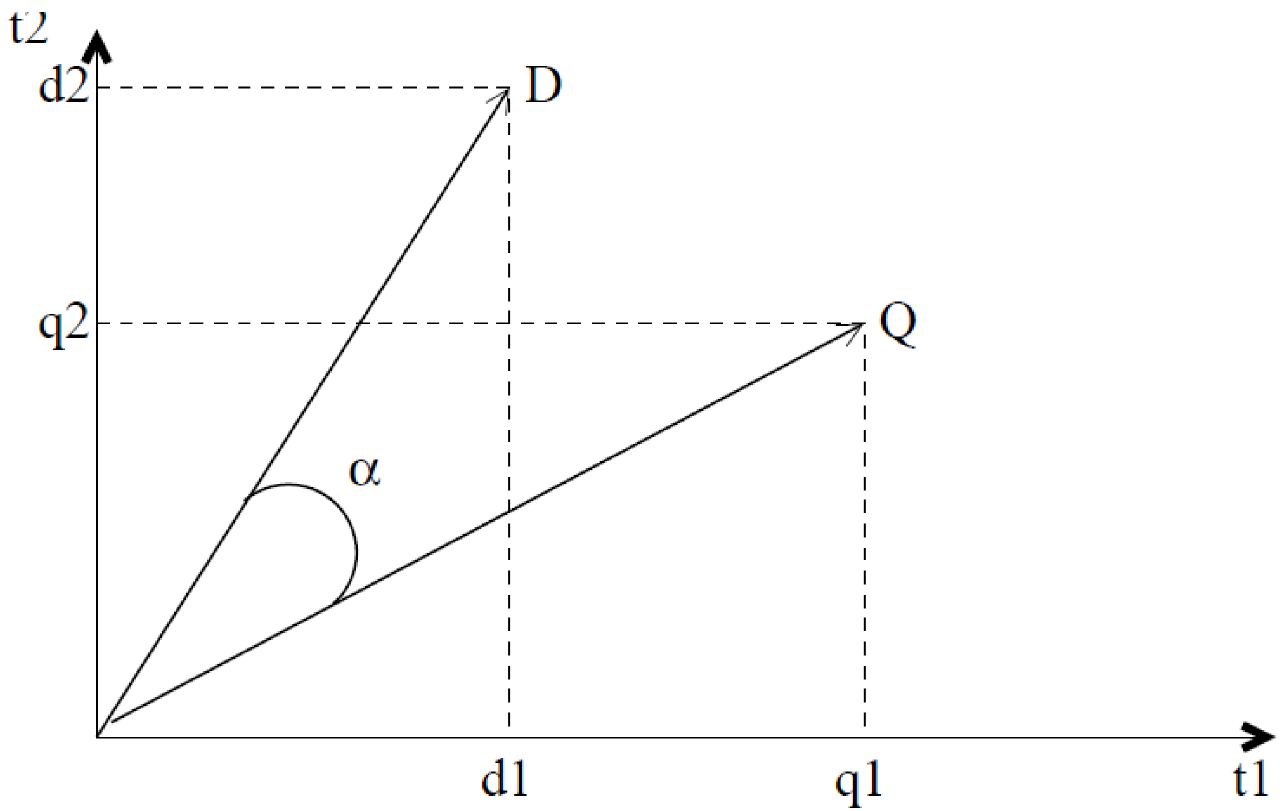
Introduction

- Set of n terms $\{t_1, t_2, \ldots, t_n\}$
- Document represented as a vector: $d =< d_1, d_2, \ldots, d_n >$
- Query represented as a vector: $q =< q_1, q_2, \ldots, q_n >$
  - d_i = weight of term t_i in document d (e.g., based on tf x idf)
  - q_i = weight of term t_i in query q (e.g., 1 if $t_i \in q$; 0 otherwise)
- Ranking (similarity) based on the angle between the query and document vectors
- Ranking function called retrieval status value (RSV):

$$R(d,q) = RSV(d,q) = \frac{\sum_{i=1,n} d_i q_i}{(\sum_{i=1,n} d_i^2)^{1/2} (\sum_{i=1,n} q_i^2)^{1/2}} = \cos\alpha$$

Vector space model (VSM)

Graphical interpretation



Here $n = 2$, meaning two terms in the collection.

Vector space model (VSM)

Vector Notation

- Ranking function (retrieval status value):

$$R(d,q) = \frac{\sum_{i=1,n} d_i q_i}{(\sum_{i=1,n} d_i^2)^{1/2} (\sum_{i=1,n} q_i^2)^{1/2}} = \cos\alpha$$

Document length normalisation

$$R(d,q) = sim(\vec{d},\vec{q}) = \cos\alpha = \frac{\vec{d}\cdot\vec{q}}{\sqrt{\vec{d}^2}\cdot\sqrt{\vec{q}^2}}$$

- The same for documents
- Can be discarded, won't affect the final ranking
- If discarded, equivalent to the projection of the query on the document vector

Vector space model (VSM)

Vector Notation

- Recap: the vector model with TF-IDF weights is a good ranking strategy with general collections, for example

$$w_{t,q} = (1 + \log tf_{t,q}) \times \log \frac{N}{df_t}$$

  - This equation should only be applied for values of term frequency greater than zero
  - If the term frequency is zero, the respective weight is also zero
- The vector model is usually as good as the known ranking alternatives. It is also simple and fast to compute

Vector space model (VSM)

Similarity Calculation

- Consider two documents $\vec{d}_1$, $\vec{d}_2$ and a query $\vec{q}$
- $\vec{d}_1 = (0.5,0.8,0.3), \vec{d}_2 = (0.9,0.4,0.2), \vec{q} = (1.0,1.0,0)$

$$R(\vec{d}_1,\vec{q}) = \frac{(0.5\times 1.0)+(0.8\times 1.0)}{\sqrt{(0.5^2+0.8^2+0.3^2)(1.0^2+1.0^2)}}$$

$$R(\vec{d}_2,\vec{q}) = \frac{(0.9\times 1.0)+(0.4\times 1.0)}{\sqrt{(0.9^2+0.4^2+0.2^2)(1.0^2+1.0^2)}}$$

Vector space model (VSM)

Advantages/Disadvanatges

- Advantages
  - Simple computational framework for ranking
  - Term-weighting improves quality of the answer set
  - Partial matching allows retrieval of docs that approximate the query conditions
  - Cosine ranking formula sorts documents according to degree of similarity to the query
  - Any similarity measure or term weighting scheme could be used
  - Document normalization is naturally built-in into the ranking

- Disadvantages
  - Assumption of term independence
  - Implicit assumptions about relevance and text representation

Binary Independence Retrieval Model (BIRM)

Probability Ranking Principle [Robertson(1977)]

- “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.”

In other words: IR as a Classification Problem

Binary Independence Retrieval Model (BIRM)

Probabilistic Retrieval Models

"Given a user query *q* and a document *d*, estimate the probability that the user will find *d* relevant."

- Known as the Binary Independence Retrieval Model (BIRM)
  - "Binary": all weights of index terms are binary (0 or 1)
  - "Independence": index terms are independent

Probabilistic Retrieval Models

BIRM

- is based on information related to presence and absence of terms in relevant and non-relevant documents
- information acquired through relevance feedback process:
  - user stating which of the retrieved documents are relevant / non-relevant (covered later)
- Capture the IR problem using a probabilistic framework
  - Bayes' decision rule

Binary Independence Retrieval Model (BIRM)

A document is described by presence/absence of terms:

$d = < x_1, x_2, \ldots, x_n >$ with n = number of terms.

$$x_i = \begin{cases} 1 & \text{if document } d \text{ indexed by } t_i \\ 0 & \text{otherwise} \end{cases}$$

1. compute for given query q:
   - $P(r \mid d, q)$, the probability of d being relevant (r)
   - $P(\neg r \mid d, q)$, the probability of d not being relevant ($\neg r$)
2. then decide whether document represented by d is relevant to query q.

The decision is expressed by the Bayes' decision rule.

Binary Independence Retrieval Model (BIRM)

BIRM: The Bayes' Decision Rule

- For each query q defined as a set of terms, we have a set of relevant documents (binary vectors)
  - $P(r \mid d, q)$: probability of judgement being relevant (r) given document d and query q
  - $P(\neg r \mid d, q)$: probability of judgement being *not* relevant ($\neg r$) given document d and query q

---

- Bayesian decision rule:

if $P(r \mid d, q) > P(\neg r \mid d, q)$
    retrieve d
else
    do not retrieve d

Binary Independence Retrieval Model (BIRM)

BIRM: Bayes' decision rule and retrieval function

Bayes' decision rule:

"if $P(r \mid d, q) > P(\neg r \mid d, q)$ then retrieve d; otherwise don't"

From above decision rule, a retrieval function $R(d, q)$ is derived:

$$R(d,q) = \begin{cases} R(d,q) > C & \text{retrieve dcument represented by } d \\ R(d,q) \le C & \text{do not retrieve dcument represented by } d \end{cases}$$

for some constant C

Binary Independence Retrieval Model (BIRM)

BIRM: How is $R(d, q)$ obtained?

if $P(r \mid d, q) > P(\neg r \mid d, q)$
    retrieve d
else
    do not retrieve d

- The Baye's decision rule says: if $P(r|d, q) > P(\neg r|d, q)$ then d is relevant for query q; otherwise d is not relevant.
- To implement this rule, need to compute $P(r|d, q)$ and $P(\neg r|d, q)$
- Since these probabilities are with respect to same query q, simplify the above to $P(r|d)$ and $P(\neg r|d)$

-> We show how to obtain $R(d, q) = R(d)$

Binary Independence Retrieval Model (BIRM)

BIRM: Bayes' Theorem

- The rule is implemented through the use of Bayes' theorem

$$P(r \mid d) = \frac{P(d \mid r) \cdot P(r)}{P(d)} \quad P(\neg r \mid d) = \frac{P(d \mid \neg r) \cdot P(\neg r)}{P(d)}$$

- $P(d)$: probability of observing d at random, i.e. probability of d irrespective of whether it is relevant or not.
- $P(d|r)$: probability of observing d given relevance
- $P(d|\neg r)$: probability of observing d given non relevance
- $P(r)$: prior probability of observing a relevant document
- $P(\neg r)$: prior probability of observing a non relevant document
- Note that from probability theory:
  $P(d) = P(d|r) \cdot P(r) + P(d|\neg r) \cdot P(\neg r)$

Binary Independence Retrieval Model (BIRM)

BIRM: Bayes' Theorem and Bayes Decision Rule

$$P(r \mid d) > P(\neg r \mid d)$$

can be rewritten as:

$$\frac{P(d \mid r) \cdot P(r)}{P(d)} > \frac{P(d \mid \neg r) \cdot P(\neg r)}{P(d)}$$

which is the same as:

$$P(d \mid r) \cdot P(r) > P(d \mid \neg r) \cdot P(\neg r)$$

The above can be rewritten as

$$\frac{P(d \mid r) \cdot P(r)}{P(d \mid \neg r) \cdot P(\neg r)} > 1$$

Binary Independence Retrieval Model (BIRM)

BIRM: Independence Assumption

We recall that $d = < x_1, x_2, \ldots , x_n >$ where $x_i = 1$ or 0.

BIRM assume independence with respect to relevance:

$$P(d \mid r) = P(< x_1, \ldots, x_n > \mid r) = \prod_{i=1,n} P(x_i \mid r)$$

BIRM assume independence with respect to non relevance:

$$P(d \mid \neg r) = P(< x_1, \ldots, x_n > \mid \neg r) = \prod_{i=1,n} P(x_i \mid \neg r)$$

Binary Independence Retrieval Model (BIRM)

BIRM: Notations

$a_i := P(x_i = 1 | r)$: probability that term t_i occurs in a relevant document

$1 - a_i = P(x_i = 0 | r)$: probability that term t_i does not occur in a relevant document

$b_i := P(x_i = 1 | \neg r)$: probability that term t_i occurs in a non-relevant document

$1 - b_i = P(x_i = 0 | \neg r)$: probability that term t_i does not occur in a non-relevant document

(In literature, you often find p_i and q_i. Leads to confusion with P and q!)

Binary Independence Retrieval Model (BIRM)

BIRM: Using the notations

$$P(d \mid r) = \prod_{i=1,n} P(x_i \mid r) = \prod_{i=1,n} a_i^{x_i} (1 - a_i)^{1 - x_i}$$

$$P(d \mid \neg r) = \prod_{i=1,n} P(x_i \mid \neg r) = \prod_{i=1,n} b_i^{x_i} (1 - b_i)^{1 - x_i}$$

Example: Document *d* = <0,1,1,0,0,1> and *n* = 6 (6 terms):

$$P(<0,1,1,0,0,1> \mid r) = (1 - a_1) \cdot a_2 \cdot a_3 \cdot (1 - a_4) \cdot (1 - a_5) \cdot a_6$$

$$P(<0,1,1,0,0,1> \mid \neg r) = (1 - b_1) \cdot b_2 \cdot b_3 \cdot (1 - b_4) \cdot (1 - b_5) \cdot b_6$$

Binary Independence Retrieval Model (BIRM)

BIRM: The way to the retrieval function $R(d)$

We return now to slide 42:

$$\frac{P(d \mid r) \cdot P(r)}{P(d \mid \neg r) \cdot P(\neg r)} > 1$$

For a set of documents, $P(r)/P(\neg r)$ is constant, so only deal with:

$$\frac{P(d \mid r)}{P(d \mid \neg r)} > 1$$

Using the indepenpendence assumptions, and notations (slide 45):

$$\frac{\prod_{i=1,n} P(x_i \mid r)}{\prod_{i=1,n} P(x_i \mid \neg r)} = \frac{\prod_{i=1,n} a_i^{x_i} (1-a_i)^{1-x_i}}{\prod_{i=1,n} b_i^{x_i} (1-b_i)^{1-x_i}} > 1$$

Binary Independence Retrieval Model (BIRM)

BIRM: The way to the retrieval function $R(d)$

From the following:

$$\frac{\prod_{i=1,n} a_i^{x_i}(1-a_i)^{1-x_i}}{\prod_{i=1,n} b_i^{x_i}(1-b_i)^{1-x_i}} > 1$$

We take the log:

$$\log \frac{\prod_{i=1,n} a_i^{x_i}(1-a_i)^{1-x_i}}{\prod_{i=1,n} b_i^{x_i}(1-b_i)^{1-x_i}} > \log(1) = 0$$

This gives (because of $(1-a_i)^{1-xi} = (1-a_i)/(1-a_i)^{xi}$):

$$\sum_{i=1,n} x_i \log \frac{a_i(1-b_i)}{b_i(1-a_i)} + \sum_{i=1,n} \log \frac{1-a_i}{1-b_i} > 0$$

Binary Independence Retrieval Model (BIRM)

BIRM: The way to the retrieval function $R(d)$

From:

$$\sum_{i=1,n} x_i \log \frac{a_i(1-b_i)}{b_i(1-a_i)} + \sum_{i=1,n} \log \frac{1-a_i}{1-b_i} > 0$$

We obtain:

$$\boxed{R(d) = \sum\nolimits_{i=1,n} c_i \cdot x_i + C}$$

where

$$c_i = \log \frac{a_i(1-b_i)}{b_i(1-a_i)}$$

$$C = \sum_{i=1,n} \log \frac{1-a_i}{1-b_i}$$

Binary Independence Retrieval Model (BIRM)

BIRM: Why such a $R(d)$

- c_i are weights associated with terms t_i, e.g. discrimination power.
- Simple addition:
  - for $c_i > 0$, term t_i occurring in document is a good indication of relevance
  - for $c_i < 0$, term t_i occurring in document is a good indication of non-relevance
  - for $c_i = 0$, term t_i occurring in document means nothing
- C constant for all documents given the same query
- Retrieval strategy:
  - if $R(d) > C$ then retrieve d; otherwise do not retrieve d or simply rank by $R(d)$ value (ignore C)

Binary Independence Retrieval Model (BIRM)

BIRM: Estimating c_i

For each term t_i:

	Relevant	Non-relevant	
$x_i = 1$	r_i	$n_i - r_i$	n_i
$x_i = 0$	$R - r_i$	$N - n_i - R + r_i$	$N - n_i$
	R	$N - R$	N

n_i : number of documents with term t_i
r_i : number of relevant documents with term t_i
R : number of relevant documents
N : number of documents

These data can be extracted after a relevance feedback process: user points out the relevant documents from a list of retrieved documents.

Binary Independence Retrieval Model (BIRM)

BIRM: Estimating c_i

We recall:

- a_i($1 - a_i$): probability that a relevant document contains (does not contain) the term t_i
- b_i($1 - b_i$): probability that a non relevant document contains (does not contain) the term t_i

$$a_i = \frac{r_i}{R} \quad b_i = \frac{n_i - r_i}{N - R}$$

so

$$c_i = \log \frac{a_i(1-b_i)}{b_i(1-a_i)} = \log \frac{r_i/(R-r_i)}{(n_i - r_i)/(N - n_i - R + r_i)}$$

BIRM: Estimating c_i - RSJ weights

$$c_i = \log \frac{r_i/(R-r_i)}{(n_i-r_i)/(N-n_i-R+r_i)}$$

is usually re-written:

$$c_i = \log \frac{(r_i+0.5)/(R-r_i+0.5)}{(n_i-r_i+0.5)/(N-n_i-R+r_i+0.5)}$$

0.5 is added to keep the c_i value from being infinite when r_i and R are small.

c_i is also referred to as term weight in BIRM; also referred to as Robertson-Spark Jones (RSJ) weights and written $w^{(1)}$.

Binary Independence Retrieval Model (BIRM)

BIRM: How does it work in practice?

- When no sample is available, R is not known
  - set $a_i = 0.5$ and $b_i = n_i / N$
  - leads to $c_i = \log (N - n_i) / n_i$
  - which can be viewed as a probabilistic *idf*
  - $R(d)$ thus with *idf* weights produces initial ranking
- Relevance feedback is then applied, and R, r_i can be defined, which has been shown to improve ranking.

Binary Independence Retrieval Model (BIRM)

BIRM: Example - Using the original c_i weights

2 terms t_1 and t_2; $d = (x_1, x_2)$; 20 documents $d_1, \ldots d_{20}$;
the query is made of term t_1 and t_2

d	Rel	x_1	x_2	d	Rel	x_1	x_2	d	Rel	x_1	x_2
d_1	r	1	1	d_2	r	1	1	d_3	r	1	1
d_4	r	1	1	d_5	$\neg r$	1	1	d_6	r	1	0
d_7	r	1	0	d_8	r	1	0	d_9	r	1	0
d_{10}	$\neg r$	1	0	d_{11}	$\neg r$	1	0	d_{12}	r	0	1
d_{13}	r	0	1	d_{14}	r	0	1	d_{15}	$\neg r$	0	1
d_{16}	$\neg r$	0	1	d_{17}	$\neg r$	0	1	d_{18}	r	0	0
d_{19}	$\neg r$	0	0	d_{20}	$\neg r$	0	0				

$N = 20$; $R = 12$; $r_1 = 8$; $r_2 = 7$; $n_1 = 11$ and $n_2 = 11$

Binary Independence Retrieval Model (BIRM)

BIRM: Example

$$a_1 = r_1 / R = 8/12; \quad a_2 = 7/12;$$

$$b_1 = (n_1 - r_1)/(N - R) = (11-8)/(20-12) = 3/8; \; b_2 = 4/8$$

Thus: (use ln for the logs)

$$c_1 = \log \frac{a_1(1-b_1)}{b_1(1-a_1)} = \log 10/3 = 1.20$$

$$c_2 = \log 7/5 = 0.34$$

Retrieval function: $R(D) = 1.20 x_1 + 0.34 x_2 + C$

BIRM: Example - Result

Retrieval results (here we ignore C):

Rank	Document	$R(d)$
Rank 1	d_1, d_2, d_3, d_4, d_5	1.54
Rank 6	$d_6, d_7, d_8, d_9, d_{10}, d_{11}$	1.20
Rank 12	$d_{12}, d_{13}, d_{14}, d_{15}, d_{16}, d_{17}$	0.34

BIRM: Summary

- Probabilistic model uses probability theory to model the "uncertainty" in the retrieval process.
- Assumptions (here independence assumptions) are made explicit
- Term weight (c_i) without relevance information is inverse document frequency (this can be proven).
- Relevance feedback can improve the ranking by giving better probability estimates of term weights.
- No use of within-document term frequencies or document lengths.

Best Match Okapi Model (BM25)

Building on the probabilistic model: Okapi weighting

- Okapi system is based on the probabilistic model
- BIRM does not perform as well as the vector space model
  - does not use term frequency (*tf*) and document length (*dl*)
  - hurt performance on long documents
- What Okapi does:
  - add a *tf* component like in the vector space model
  - separate document and query length normalization
  - several tuning constants, which depend on the collection

Best Match Okapi Model (BM25)

BM25 (Best-match Okapi weight)

$$R(d,q) = BM25(d,q) =$$

$$\sum_{t \in q} \left(w_t \cdot \frac{(k_1+1)tf(t,d)}{K+tf(t,d)} \cdot \frac{(k_3+1)tf(t,q)}{k_3+tf(t,q)}\right) + k_2 \cdot |q| \cdot \frac{avgdl-dl}{avgdl+dl}$$

$$K = k_1((1-b)+(b \cdot dl)/avdl)$$

w_t:	term weight based on relevance feedback
$tf(t,d), tf(t,q)$:	within term frequencies – document and query
k_1, k_2, k_3, b:	tuning parameters
dl, $avgdl$:	document length and average document length

Best Match Okapi Model (BM25)

BM25 - Parameters

$$\sum_{t \in q} (w_t \cdot \frac{(k_1+1) tf(t,d)}{K + tf(t,d)} \cdot \frac{(k_3+1) tf(t,q)}{k_3 + tf(t,q)}) + k_2 \cdot |q| \cdot \frac{avgdl - dl}{avgdl + dl}$$

$$K = k_1((1-b) + (b \cdot dl) / avdl)$$

k_1 : governs the importance of within document frequency $tf(t, d)$

k_2: compensation factor for the high within document frequency values in large documents

k_3: governs the importance of within query frequency $tf(t, q)$

b: relative importance of within document frequency and document length

The theoretical basis for the Okapi formula is the use of Poisson distributions to model within document frequency in relevant documents, and in non-relevant documents (not discussed here).

Best Match Okapi Model (BM25)

- A common simplified form:

$$RSV^{BM25} = \sum_{i \in q} \log \frac{N}{df_i} \cdot \frac{(k_1 + 1) tf_i}{k_1((1 - b) + b \frac{dl}{avdl}) + tf_i}$$

- Typically, k_1 is set around [1.2,2] and b around 0.75

Best Match Okapi Model (BM25)

- Many formulations and interpretations of BM25
- $\log \frac{N}{df_i}$: the IDF term
- Can be re-written as $\log \frac{N - n(q_i) + 0.5}{n(q_i) + 0.5}$
- N is the total number of documents in the collection
- $n(q_i)$ is the number of documents containing q_i

Best Match Okapi Model (BM25)

BM25 (Best-match Okapi weight)

- Experiments show:
  - k_2 =0; k_3 large; b closer to 1
  - Leading for instance to (with k_1 = 1 and b = 0.75):

$$BM25(d,q) = \sum_{t \in q} (w_t \cdot \frac{tf(t,d)}{K + tf(t,d)})$$

  - $K = 0.25 + (0.75\ dl)/avdl$
- In experiments, Okapi weights give the best performance. BM25 often used as baseline model in retrieval experiments.

Best Match Okapi Model (BM25)

Summary

- The vector space model (VSM) is a “geometrical” model; it is viewed as “basic”. Established in the 70s.
- The BIRM is one of the important pieces of IR theory.
    - Ranking based on the probability of relevance is optimal with respect to a cost function where the costs for reading relevant documents are low and the costs for reading non-relevant documents are high; probability ranking principle [Robertson 1977].
- BM25 Okapi model is often the most “effective” model, the model to “beat” in retrieval experiments.
- BM25F (BM25 Field) – take document structure and anchor text into account