Boolean model

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

Boolean model

- The Boolean model is a simple retrieval model based on
  - set theory, Boolean algebra

- Index term's significance is represented by binary weights
  - $w_{i,j} \in \{0,1\}$

- The set of index terms for a document d_j is denoted as R_{d_j}
- The set of documents for an index term t_i is denoted as R_{t_i}

- Queries are defined as Boolean expressions over index terms
  - Boolean operators AND, OR, NOT

- The relevance is modelled as a binary property of the documents
  - $SC(q, d_j) = 0$ or $SC(q, d_j) = 1$

Boolean model: example

- Given a set of index terms $\{t_1, t_2, t_3\}$
- Given a set of documents
  - $d_1 = [1,1,1]^T$
  - $d_2 = [1,0,0]^T$
  - $d_3 = [0,1,0]^T$

- Calculate the set of documents for each index term
  - $R_{t_1} = \{d_1, d_2\}$
  - $R_{t_2} = \{d_1, d_3\}$
  - $R_{t_3} = \{d_1\}$

Boolean model: example

- Each query can be expressed in terms of R_{t_i}
- $q = t_1 \rightarrow R_{t_1} = \{d_1, d_2\}$
- $q = t_1 \wedge t_2 \rightarrow R_{t_1} \cap R_{t_2} = \{d_1, d_2\} \cap \{d_1, d_3\} = \{d_1\}$
- $q = t_1 \vee t_2 \rightarrow R_{t_1} \cup R_{t_2} = \{d_1, d_2\} \cup \{d_1, d_3\} = \{d_1, d_2, d_3\}$
- $q = \neg t_3 \rightarrow R_{t_3}^C = \{d_1\}^C = \{d_2, d_3\}$

Boolean model: queries in DNF

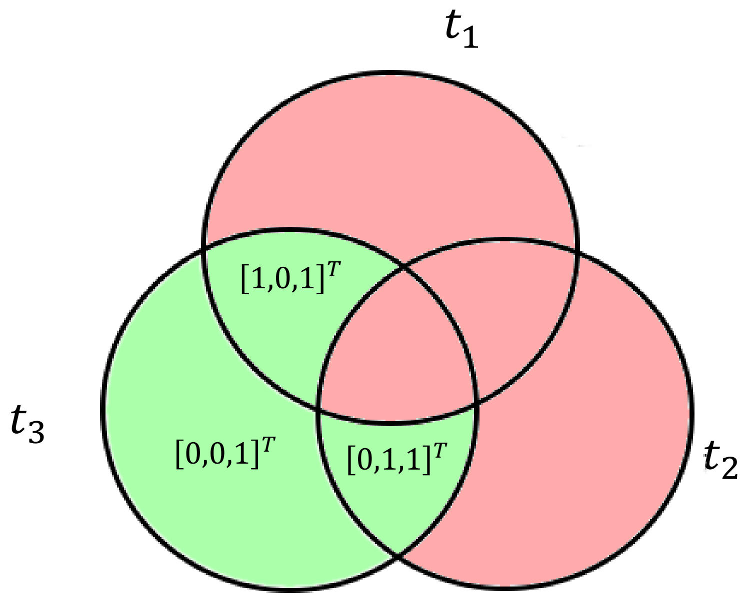
- Each query can be also expressed in a Disjunctive Normal Form
  - $q = t_3 \land \neg(t_1 \land t_2) \rightarrow \mathrm{R}_{\mathrm{t}_3} \cap \left(R_{t_1} \cap R_{t_2}\right)^C$

t_1	t_2	t_3
0	*0*	*0*
0	*0*	*1*
0	*1*	*0*
0	*1*	*1*
1	*0*	*0*
1	*0*	*1*
1	*1*	*0*
1	*1*	*1*

  - $q_{dnf} = (\neg t_1 \land \neg t_2 \land t_3) \lor (\neg t_1 \land t_2 \land t_3) \lor (t_1 \land \neg t_2 \land t_3)$

Boolean model: queries in DNF

- $q = t_3 \wedge \neg(t_1 \wedge t_2)$
- $q_{dnf} = (\neg t_1 \wedge \neg t_2 \wedge t_3) \vee (\neg t_1 \wedge t_2 \wedge t_3) \vee (t_1 \wedge \neg t_2 \wedge t_3)$



- Each disjunction represents an ideal set of documents

- The query is satisfied by a document if such document is contained in a disjunction term

Boolean model: conclusions

- Pros
  - Precise semantics
  - Structured queries
  - Intuitive for experts
  - Simple and neat formalism
    - Adopted by many of early commercial bibliographic systems

- Cons
  - No ranking
    - Retrieval strategy is based on a binary decision criterion
  - Not simple to translate an information need into a Boolean expression

Vector space model

- Boolean model

- **Vector space model**

- Probabilistic model

Vector space model

- Documents and queries are represented as vectors in the index terms space

- M = number of index terms = index terms space dimensionality = dictionary size

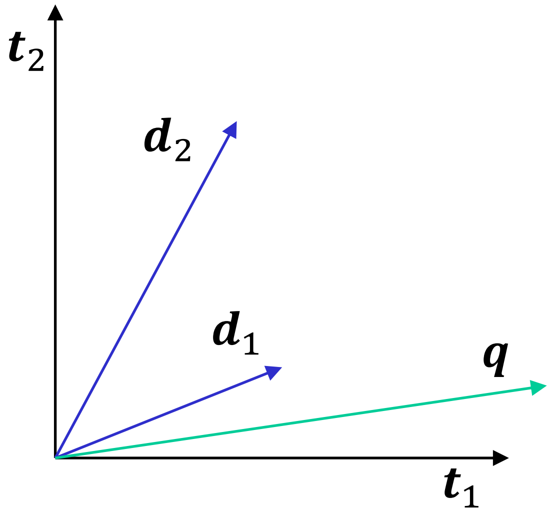
- Index term's significance is represented by real valued weights
  - $w_{i,j} \geq 0$ is associated to the pair $\left(t_i, d_j\right)$

- Index terms are represented by unit vectors and form a canonical basis for the vector space
  - Index term vector for term t_i
    - $\boldsymbol{t}_i = [0, \dots, 1, \dots, 0]^T$

Vector space model

- Document d_j is represented by a vector $\boldsymbol{d}_j$ which is a linear combination of index term vectors weighted by the index term significance for the document
  - $\boldsymbol{d}_j = \sum_{i=1}^{M} w_{i,j} \cdot \boldsymbol{t}_i$

- Query q is represented by a vector $\boldsymbol{q}$ in the index terms space
  - $\boldsymbol{q} = \left[w_{1,q}, w_{2,q}, \dots, w_{M,q}\right]^T$



Vector space model: similarity

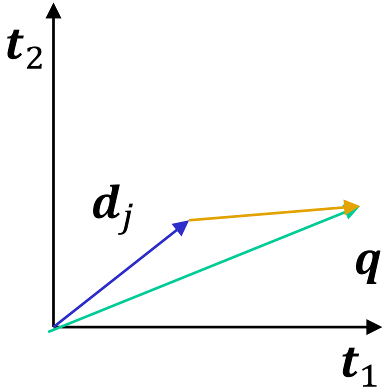
- The Similarity Coefficient between a query and each document is real valued, thus producing a ranked list of documents
  - It takes into consideration documents which match the query terms only partially
  - Ranked document answer set is more effective than document answer set retrieved by the Boolean model
    - Better match of user information need

- There are various *measures* that can be used to asses the similarity between documents/query
- A measure of similarity between documents shall fulfil the following
  - If d_1 is near d_2, then d_2 is near d_1
  - If d_1 is near d_2, and d_2 is near d_3, then d_1 is not far from d_3
  - No document is closer to d than d itself

Vector space model: similarity

- Euclidean distance
  - Length of difference vector

$$d_{L_2}(\boldsymbol{q}, \boldsymbol{d}_j) = \left\|\boldsymbol{q} - \boldsymbol{d}_j\right\|_2 = \sqrt{\sum_{i=1}^{M} \left(w_{i,q} - w_{i,j}\right)^2}$$



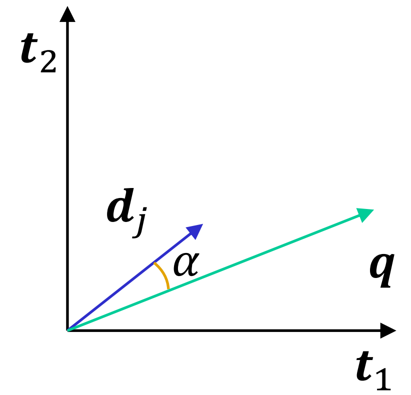
  - Can be converted in a similarity coefficient in different ways
    - $SC(q, d_j) = e^{-\left\|\boldsymbol{q} - \boldsymbol{d}_j\right\|_2}$
    - $SC(q, d_j) = \frac{1}{1 + \left\|\boldsymbol{q} - \boldsymbol{d}_j\right\|_2}$
  - Issue of normalization
    - Euclidian distance applied to un-normalized vectors tend to make any large document to be not relevant to most queries, which are typically short

Vector space model: similarity

- Cosine similarity
  - Cosine of the angle between two vectors

$$SC(q, d_j) = \cos(\alpha) = \frac{\boldsymbol{q}^T \boldsymbol{d}_j}{\|\boldsymbol{q}\| \|\boldsymbol{d}_j\|} = $$

$$= \frac{\sum_{i=1}^{M} w_{i,q} w_{i,j}}{\sqrt{\sum_{i=1}^{M} (w_{i,q})^2 \sum_{i=1}^{M} (w_{i,j})^2}}$$



  - *It's a similarity, not a distance!*
    - Triangle inequality holds for distances but not for similarities
  - The cosine measure normalizes the results by considering the length of the document vector
  - Given two vectors, their similarity is determined by their directions
  - For normalized vectors, the cosine similarity is equal to the *inner product*

Vector space model: similarity

- Jaccard's similarity

$$SC(q, d_j) = \frac{\boldsymbol{q}^T \boldsymbol{d}_j}{\boldsymbol{q}^T \boldsymbol{q} + \boldsymbol{d}_j^T \boldsymbol{d}_j - \boldsymbol{q}^T \boldsymbol{d}_j} =$$

$$= \frac{\sum_{i=1}^{M} w_{i,q} w_{i,j}}{\sum_{i=1}^{M} \left(w_{i,q}\right)^2 + \sum_{i=1}^{M} \left(w_{i,j}\right)^2 - \sum_{i=1}^{M} w_{i,q} w_{i,j}}$$

  - Extension of Jaccard's similarity coefficient for binary vectors

- Other similarity measures
  - Dice's similarity coefficient
  - Overlap coefficient

Vector space model: index term weighting

- Boolean model used binary weights for index terms in a document
  - terms with different discriminative power have the same weight
  - normalization might not be enough to compensate for differences in documents lengths. A longer document has more opportunity to have some components that are relevant to a query

- In Vector Space Model index terms weights are non-negative real-valued
  - Index term weights should be made proportional to its importance, both in the document and in the document collection

Vector space model: index term weighting

- In Vector Space Model the index term weighting is defined as

$$w_{i,j} = tf_{i,j} \cdot idf_i$$

- $tf_{i,j} \rightarrow$ frequency of term t_i in document d_j
  - provides one measure of how well term t_i describes the document contents
- $idf_i \rightarrow$ inverse document frequency of term t_i for the whole document collection
  - terms which appear in many documents are not very useful for distinguishing a relevant document from a non-relevant one
- $w_{i,j}$
  - increases with the *number of occurrences* of term t_i within document d_j
  - Increases with the *rarity* of term t_i across the whole document collection

Vector space model: index term weighting

- $tf_{i,j} \rightarrow$ frequency of term t_i in document d_j

- Define $freq_{i,j}$ = number of occurrences of term t_i in document d_j

- Three possible models for $tf_{i,j}$
  - $tf_{i,j} = freq_{i,j}$
    - simplest model
  - $tf_{i,j} = \dfrac{freq_{i,j}}{\max_i freq_{i,j}}$
    - normalized model
  - $tf_{i,j} = \begin{cases} 1 + \log_2(freq_{i,j}) & \text{if } freq_{i,j} \geq 1 \\ 0 & \text{otherwise} \end{cases}$
    - prevents bias toward longer documents

Vector space model: index term weighting

- $idf_i \rightarrow$ inverse document frequency of term t_i for the whole documents collection

- Define N = number of documents in the collection
- Define n_i = number of documents containing term t_i

$$idf_i = \log_2 \frac{N}{n_i}$$

Vector space model: example

- *Given*
  - Query
    - q = *"gold silver truck"*
  - Documents collection
    - d_1 = "shipment of gold damaged in a fire"
    - d_2 = "delivery of silver arrived in a silver truck"
    - d_3 = "shipment of gold arrived in a truck"
  - The term frequency model
    - $tf_{i,j} = freq_{i,j}$

- *Determine the ranking of the documents collection with respect to the given query using a Vector Space Model with the following similarity measures*
  - Euclidean distance
  - Cosine similarity

Vector space model: example

- Dictionary = {“shipment”, “of”, “gold”, “damaged”, “in”, “a”, “fire”, “delivery”, “silver”, “arrived”, “truck”}
- $N = 3 \rightarrow$ number of documents in the collection
- idf_i for each term t_i

t_i	ship ment	of	gold	dam aged	in	a	fire	deliv ery	silver	arriv ed	truck
n_i	2	3	2	1	3	3	1	1	1	2	2
idf_i	0.58	0	0.58	1.58	0	0	1.58	1.58	1.58	0.58	0.58

- Purge dictionary removing terms with $idf_i = 0$
  - Dictionary = {“shipment”, ”gold”, ”damaged”, ”fire”, ”delivery”, ”silver”, ”arrived”, ”truck”}

Vector space model: example

- $tf_{i,j}$ for each couple (t_i, d_j)

$t_{\boldsymbol{i}}$	shipment	gold	damaged	fire	delivery	silver	arrived	truck
$tf_{i,1}$	1	1	1	1	0	0	0	0
$tf_{i,2}$	0	0	0	0	1	2	1	1
$tf_{i,3}$	1	1	0	0	0	0	1	1

- $w_{i,j}$ for each couple (t_i, d_j)

$t_{\boldsymbol{i}}$	shipment	gold	damaged	fire	delivery	silver	arrived	truck
$w_{i,1}$	0.58	0.58	1.58	1.58	0	0	0	0
$w_{i,2}$	0	0	0	0	1.58	3.16	0.58	0.58
$w_{i,3}$	0.58	0.58	0	0	0	0	0.58	0.58

Vector space model: example

- $tf_{i,q}$ and $w_{i,q}$ for each term t_i

$t_{\boldsymbol{i}}$	shipment	gold	damaged	fire	delivery	silver	arrived	truck
$tf_{i,q}$	0	1	0	0	0	1	0	1
$w_{i,q}$	0	0.58	0	0	0	1.58	0	0.58

- Let's write the documents and the query as vectors
  - $\boldsymbol{d}_1 = [0.58, 0.58, 1.58, 1.58, 0, \quad 0, \quad 0, \quad 0 \quad]$
  - $\boldsymbol{d}_2 = [0, \quad 0, \quad 0, \quad 0, \quad 1.58, 3.16, 0.58, 0.58]$
  - $\boldsymbol{d}_3 = [0.58, 0.58, 0, \quad 0, \quad 0, \quad 0, \quad 0.58, 0.58]$
  - $\boldsymbol{q} \; = [0, \quad 0.58, 0, \quad 0, \quad 0, \quad 1.58, 0, \quad 0.58]$

Vector space model: example

- Euclidean distance as Similarity Coefficient
  - $d_{L_2}(\boldsymbol{q}, \boldsymbol{d}_1) = \sqrt{0.58^2 + 1.58^2 + 1.58^2 + 1.58^2 + 0.58^2} = 2.86$
  - $d_{L_2}(\boldsymbol{q}, \boldsymbol{d}_2) = \sqrt{0.58^2 + 1.58^2 + 1.58^2 + 0.58^2} = 2.38$
  - $d_{L_2}(\boldsymbol{q}, \boldsymbol{d}_3) = \sqrt{0.58^2 + 1.58^2 + 0.58^2} = 1.78$

  - $SC(q, d_1) = \frac{1}{1+d_{L_2}(\boldsymbol{q},\boldsymbol{d}_1)} = 0.26$
  - $SC(q, d_2) = \frac{1}{1+d_{L_2}(\boldsymbol{q},\boldsymbol{d}_2)} = 0.30$
  - $SC(q, d_3) = \frac{1}{1+d_{L_2}(\boldsymbol{q},\boldsymbol{d}_3)} = 0.36$

  - Rank: $d_3 > d_2 > d_1$

Vector space model: example

- Cosine similarity as Similarity Coefficient
  - $SC(q, d_1) = \frac{\boldsymbol{q}^T \boldsymbol{d}_1}{\|\boldsymbol{q}\| \|\boldsymbol{d}_1\|} = \frac{0.34}{1.79 \cdot 2.39} = 0.08$
  - $SC(q, d_2) = \frac{\boldsymbol{q}^T \boldsymbol{d}_2}{\|\boldsymbol{q}\| \|\boldsymbol{d}_2\|} = \frac{5.37}{1.79 \cdot 3.64} = 0.82$
  - $SC(q, d_3) = \frac{\boldsymbol{q}^T \boldsymbol{d}_3}{\|\boldsymbol{q}\| \|\boldsymbol{d}_3\|} = \frac{0.68}{1.79 \cdot 1.17} = 0.33$

  - Rank: $d_2 > d_3 > d_1$

Vector space model: conclusions

- Pros
  - Term-weighting scheme improves retrieval performance w.r.t. Boolean model
  - Partial matching strategy allows retrieval of documents that approximate the query conditions
  - Ranked output and output magnitude control
  - Flexibility and intuitive geometric interpretation

- Cons
  - Assumption of independency between terms
  - Impossibility of formulating "structured" queries (No operator (OR, AND, NOT, etc..)
  - Terms are axes of a vector space (Even with stemming, may have 20.000+ dimensions)

Probabilistic model

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

Probabilistic model

- The probabilistic model computes the Similarity Coefficient between queries and documents as the *probability that a document will be relevant to a query*

$$P(relevant|q, dj)$$

- Given a query q, let R_q denote the set of documents relevant to the query q (the ideal answer set)

- The set R_q is unknown

Probabilistic model

- First, generate a preliminary probabilistic description of the ideal answer set R_q which is used to retrieve a first set of documents.
  - From relevant documents if some are known
    - relevance feedback
  - Prior domain knowledge

Probabilistic model

- The probabilistic model can be used together with relevance feedback
- An interaction with the user is then initiated with the purpose of improving the probabilistic description of the ideal answer set.
  - The *user* takes a look at the retrieved documents and *decides which ones are relevant* and which ones are not (in truth, only the first top documents need to be examined).
  - The *system* then uses this information to *refine the description* of the ideal answer set.
  - By repeating this process many times, it is expected that such a description will evolve and become closer to the real description of the ideal answer set.

Probabilistic model: probability basics

- Complementary events $p(a) + p(\bar{a}) = 1$
- Joint probability $p(a, b) = p(a \cap b)$
- Conditional probability $p(a|b) = \frac{p(a \cap b)}{p(b)} = \frac{p(a,b)}{p(b)}$
- Marginal probability $p(a) = \sum_{x \in \{b, \bar{b}\}} p(a|x) \cdot p(x)$
- Bayes' theorem $p(a|b) \cdot p(b) = p(b|a) \cdot p(a)$
- Odds $O(a) = \frac{p(a)}{p(\bar{a})} = \frac{p(a)}{1-p(a)}$

Probabilistic model: Prob. Ranking Principle

- Probabilistic model captures IR problem in a probabilistic framework

- The probability ranking principle (PRP)
  - “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, …, *the overall effectiveness of the system to its user will be the best that is obtainable*…” [Robertson and Jones, 1976]

- Classic probabilistic models are also known as binary independence retrieval (BIR) models

Probabilistic model: Prob. Ranking Principle

- Let d be a document in the collection
- Let R represent relevance of a document w.r.t a given (fixed) query q and let NR represent non-relevance
- Need to find $p(R|q, d_j)$: probability that a document d_j is relevant given the query q
- According to PRP, rank documents in descending order of $p(R|q, d_j)$

Probabilistic model: BIM

- Model hypothesis: Binary Independence Model
  - Distribution of terms in relevant documents is different from of terns in non relevant documents
  - Terms that occur in many relevant documents, and are absent in many irrelevant documents, should be given greater importance
- Binary = Boolean: documents are represented as binary incidence vectors of terms (remember the Boolean model?)
  - $w_{i,j} \in \{0,1\}$
    - $w_{i,j} = 1$ if term t_i is present in document d_j
    - otherwise $w_{i,j} = 0$
- Independence: terms occur in documents independently one from each other
- Remark: different documents can be modelled by same vector

Probabilistic model: BIM

- Documents are binary incidence vectors (as for Boolean model)
  - $d_j \rightarrow \boldsymbol{d}_j$
- Also queries are binary incidence vectors (remember that in Boolean models queries were logical expressions)
  - $q \rightarrow \boldsymbol{q}$

- Given a query q
  - For each document d_j need to compute $p(R|q, d_j)$
  - Replace with computing $p(R|\boldsymbol{q}, \boldsymbol{d}_j)$, where $\boldsymbol{d}_i$ is the binary incidence vector representing d_j and $\boldsymbol{q}$ is the binary incidence vector representing q

Probabilistic model: BIM

- Prior probabilities (independent from a specific document)
  - $p(R|q)$ probability of retrieving a relevant document
  - $p(NR|q)$ probability of retrieving a non-relevant document

- $p(d_j|q, R)$ probability that if a relevant document is retrieved, it is document d_j.

- $p(d_j|q, NR)$ probability that if a non-relevant document is retrieved, it is document d_j.

Probabilistic model: BIM

- Need to find the posterior probability for a specific document. By Bayes' theorem:
  - $p(R|q,d_j) = p(d_j|q,R) \cdot \frac{p(R|q)}{p(d_j|q)}$
  - $p(NR|q,d_j) = p(d_j|q,NR) \cdot \frac{p(NR|q)}{p(d_j|q)}$
  - $p(R|q,d_j) + p(NR|q,d_j) = 1$

- Given a query q
  - Estimate how terms contribute to relevance
  - Compute the probability of each document d_j to be relevant with respect to the query $q \rightarrow p(R|q,d_j)$
  - Order documents by decreasing probability $p(R|q,d_j)$

Probabilistic model: BIM

- Starting from the odds

$$\boxed{O(R|\boldsymbol{q},\boldsymbol{d}_j)} = \frac{p(R|\boldsymbol{q},\boldsymbol{d}_j)}{p(NR|\boldsymbol{q},\boldsymbol{d}_j)} = \frac{\dfrac{p(\boldsymbol{d}_j|\boldsymbol{q},R)p(R|\boldsymbol{q})}{p(\boldsymbol{d}_j|\boldsymbol{q})}}{\dfrac{p(\boldsymbol{d}_j|\boldsymbol{q},NR)p(NR|\boldsymbol{q})}{p(\boldsymbol{d}_j|\boldsymbol{q})}}$$

$$= \frac{p(\boldsymbol{d}_j|\boldsymbol{q},R)p(R|\boldsymbol{q})}{p(\boldsymbol{d}_j|\boldsymbol{q},NR)p(NR|\boldsymbol{q})} = \boxed{\frac{p(R|\boldsymbol{q})}{p(NR|\boldsymbol{q})}}\ \boxed{\frac{p(\boldsymbol{d}_j|\boldsymbol{q},R)}{p(\boldsymbol{d}_j|\boldsymbol{q},NR)}}$$

  - constant for a given query
    - $O(R|\boldsymbol{q}) = \frac{p(R|\boldsymbol{q})}{p(NR|\boldsymbol{q})}$
  - needs to be estimated for each document

Probabilistic model: BIM

- Using the independence assumption
  - Consider a dictionary of M terms

$$p(\boldsymbol{d}_j|\boldsymbol{q},R) = \prod_{i=1}^{M} p(w_{i,j}|\boldsymbol{q},R)$$

$$p(\boldsymbol{d}_j|\boldsymbol{q},NR) = \prod_{i=1}^{M} p(w_{i,j}|\boldsymbol{q},NR)$$

$$\boxed{\frac{p(\boldsymbol{d}_j|\boldsymbol{q},R)}{p(\boldsymbol{d}_j|\boldsymbol{q},NR)}} = \prod_{i=1}^{M} \frac{p(w_{i,j}|\boldsymbol{q},R)}{p(w_{i,j}|\boldsymbol{q},NR)}$$

$$\boxed{O(R|\boldsymbol{q},\boldsymbol{d}_j)} = \boxed{O(R|\boldsymbol{q})} \boxed{\prod_{i=1}^{M} \frac{p(w_{i,j}|\boldsymbol{q},R)}{p(w_{i,j}|\boldsymbol{q},NR)}}$$

Probabilistic model: BIM

$$\boxed{O(R|\boldsymbol{q},\boldsymbol{d}_j)} = \boxed{O(R|\boldsymbol{q})} \boxed{\prod_{i=1}^{M} \frac{p(w_{i,j}|\boldsymbol{q},R)}{p(w_{i,j}|\boldsymbol{q},NR)}}$$

- Given that $w_{i,j} \in \{0,1\}$

$$\prod_{i=1}^{M} p(w_{i,j}|\boldsymbol{q},R) = \prod_{i|w_{i,j}=1} p(w_i = 1|\boldsymbol{q},R) \prod_{i|w_{i,j}=0} p(w_i = 0|\boldsymbol{q},R)$$

$$\prod_{i=1}^{M} p(w_{i,j}|\boldsymbol{q},NR) = \prod_{i|w_{i,j}=1} p(w_i = 1|\boldsymbol{q},NR) \prod_{i|w_{i,j}=0} p(w_i = 0|\boldsymbol{q},NR)$$

- $p_i \triangleq p(w_i = 1|\boldsymbol{q},R) \rightarrow$ probability of term t_i appearing in a document relevant to the query
- $u_i \triangleq p(w_i = 1|\boldsymbol{q},NR) \rightarrow$ probability of term t_i appearing in a document non relevant to the query
- Assuming that for all the terms not occurring in the query

$$p_i = u_i$$

Probabilistic model: BIM

- Continues…

$$\boxed{O(R|\boldsymbol{q},\boldsymbol{d}_j)} = \boxed{O(R|\boldsymbol{q})} \boxed{\prod_{i=1}^{M} \frac{p(w_{i,j}|\boldsymbol{q},R)}{p(w_{i,j}|\boldsymbol{q},NR)}}$$

$$= \boxed{O(R|\boldsymbol{q})} \prod_{i|w_{i,j}=1} \frac{p(w_i = 1|\boldsymbol{q},R)}{p(w_i = 1|\boldsymbol{q},NR)} \prod_{i|w_{i,j}=0} \frac{p(w_i = 0|\boldsymbol{q},R)}{p(w_i = 0|\boldsymbol{q},NR)}$$

$$= \boxed{O(R|\boldsymbol{q})} \prod_{i|w_{i,j}=1} \frac{p_i}{u_i} \prod_{i|w_{i,j}=0} \frac{1-p_i}{1-u_i}$$

- Since for the terms not occurring in the query $p_i = u_i$
  we consider only the terms t_i occurring in the query ($w_{i,q} = 1$)

$$\boxed{O(R|\boldsymbol{q},\boldsymbol{d}_j)} = \boxed{O(R|\boldsymbol{q})} \boxed{\prod_{i \left| \substack{w_{i,j}=1 \\ w_{i,q}=1} \right.} \frac{p_i}{u_i}} \boxed{\prod_{i \left| \substack{w_{i,j}=0 \\ w_{i,q}=1} \right.} \frac{1-p_i}{1-u_i}}$$

  - Terms occurring both in the query and in the document
  - Terms occurring only in the query

Probabilistic model: BIM

- Continues…

$$\boxed{\prod_{i \,\middle|\, \substack{w_{i,j}=0 \\ w_{i,q}=1}} \frac{1-p_i}{1-u_i}} = \frac{\boxed{\prod_{i|w_{i,q}=1} \frac{1-p_i}{1-u_i}}}{\boxed{\prod_{i \,\middle|\, \substack{w_{i,j}=1 \\ w_{i,q}=1}} \frac{1-p_i}{1-u_i}}} = \boxed{\prod_{i \,\middle|\, \substack{w_{i,j}=1 \\ w_{i,q}=1}} \frac{1-u_i}{1-p_i}} \boxed{\prod_{i|w_{i,q}=1} \frac{1-p_i}{1-u_i}}$$

- Terms occurring both in the query and in the document
- All the terms occurring in the query: document independent

$$\boxed{O\left(R \middle| \boldsymbol{q}, \boldsymbol{d}_j\right)} = \boxed{O(R|\boldsymbol{q})} \boxed{\prod_{i \,\middle|\, \substack{w_{i,j}=1 \\ w_{i,q}=1}} \frac{p_i}{u_i} \boxed{\frac{(1-u_i)}{(1-p_i)}}} \boxed{\prod_{i|w_{i,q}=1} \frac{1-p_i}{1-u_i}}$$

Probabilistic model: BIM

$$O(R|\boldsymbol{q},\boldsymbol{d}_j) = \boxed{O(R|\boldsymbol{q})} \boxed{\prod_{i \left| \substack{w_{i,j}=1 \\ w_{i,q}=1} \right.} \frac{p_i\ (1-u_i)}{u_i\ \ (1-p_i)}} \boxed{\prod_{i|w_{i,q}=1} \frac{1-p_i}{1-u_i}}$$

- Constant for each query
- Only quantity to be estimated for ranking

$$SC(q,d_j) \triangleq \log_2 \prod_{i \left| \substack{w_{i,j}=1 \\ w_{i,q}=1} \right.} \frac{p_i\ (1-u_i)}{u_i\ \ (1-p_i)} = \sum_{i \left| \substack{w_{i,j}=1 \\ w_{i,q}=1} \right.} \log_2 \frac{p_i\ (1-u_i)}{u_i\ \ (1-p_i)}$$

- How do we estimate p_i and u_i from data?

Probabilistic model: BIM

- For each term t_i consider the following table of document counts

	Relevant	Non-Relevant	Total
documents containing $\boldsymbol{t_i}$	s_i	$n_i - s_i$	n_i
documents not containing $\boldsymbol{t_i}$	$S - s_i$	$(N - S) - (n_i - s_i)$	$N - n_i$
Total	S	$N - S$	N

- Estimates
  - $p_i = p(w_i = 1|\boldsymbol{q}, R) = \frac{s_i}{S}$
  - $u_i = p(w_i = 1|\boldsymbol{q}, NR) = \frac{n_i - s_i}{N - S}$

Probabilistic model: BIM

- u_i is initialized as $\frac{n_i}{N}$, as non relevant documents are approximated by the whole documents collection
- p_i can be initialized in various ways
  - From relevant documents if known some (e.g. relevance feedback, prior knowledge)
  - Constant (e.g. $p_i = 0.5$ (even odds) for any given document)
  - Proportional to probability of occurrence in collection

- An iterative procedure is used to refine p_i and u_i
  1. Determine a guess of relevant document set
     – Top-S ranked documents or user's relevance feedback
  2. Update the estimates for p_i and u_i
     – $p_i = \frac{s_i}{S}$ $\quad u_i = \frac{n_i - s_i}{N - S}$
  - Go to 1 until convergence then return ranking

Probabilistic model: example

- *Given*
  - Query incidence vector
    - $\boldsymbol{q} = [0\ 1\ 0\ 0\ 1\ 1]$
  - Documents incidence vectors
    - $\boldsymbol{d_1} = [1\ 0\ 0\ 1\ 0\ 1]$
    - $\boldsymbol{d_2} = [1\ 0\ 0\ 1\ 0\ 0]$
    - $\boldsymbol{d_3} = [0\ 0\ 1\ 1\ 1\ 0]$
    - $\boldsymbol{d_4} = [1\ 1\ 0\ 0\ 1\ 0]$
  - Initialize
    - $p_i = 0.5$
  - Consider as relevant the top-2 documents ($S = 2$)

Probabilistic model: example

- *Determine the ranking of the documents collection with respect to the given query using a Probabilistic Model under the Binary Independence assumption.*
- $N = 4 \rightarrow$ number of documents in the collection
- Reduce the documents incidence vectors considering only the terms that appear in the query
  - $\boldsymbol{d_1} = [0\ 0\ 1]$
  - $\boldsymbol{d_2} = [0\ 0\ 0]$
  - $\boldsymbol{d_3} = [0\ 1\ 0]$
  - $\boldsymbol{d_4} = [1\ 1\ 0]$
- Initialize u_i for each term t_i

	t_1	t_2	t_3
n_i	1	2	1
u_i	0.25	0.50	0.25

- Recall that $p_i = 0.5\ \forall\ t_i$

Probabilistic model: example

- 1st iteration
  - Determine the SC for each document
    - $SC(d_1, q) = \log_2 \frac{p_3}{1-p_3} + \log_2 \frac{1-u_3}{u_3} = 1.59$
    - $SC(d_2, q) = -\infty$ (d_2 contains no query terms)
    - $SC(d_3, q) = \log_2 \frac{p_2}{1-p_2} + \log_2 \frac{1-u_2}{u_2} = 0$
    - $SC(d_4, q) = \log_2 \frac{p_1}{1-p_1} + \log_2 \frac{1-u_1}{u_1} + \log_2 \frac{p_2}{1-p_2} + \log_2 \frac{1-u_2}{u_2} = 1.59$

  - Rank the documents and consider the first top-k as relevant (in case of tie keep documents ordering
    - $d_1 > d_4 > d_3 > d_2$
    - Relevant documents = $\{d_1, d_4\}$

Probabilistic model: example

- 1st iteration (continues…)
  - Calculate s_i and update p_i and u_i
    - $s_1 = 1$
    - $s_2 = 1$
    - $s_3 = 1$
    - $p_1 = \frac{s_1}{S} = \frac{1}{2} = 0.5$
    - $p_2 = \frac{s_2}{S} = \frac{1}{2} = 0.5$
    - $p_3 = \frac{s_3}{S} = \frac{1}{2} = 0.5$
    - $u_1 = \frac{n_1 - s_1}{N - S} = \frac{0}{2} = 0$
    - $u_2 = \frac{n_2 - s_2}{N - S} = \frac{1}{2} = 0.5$
    - $u_3 = \frac{n_3 - s_3}{N - S} = \frac{0}{2} = 0$

Probabilistic model: example

- 2nd iteration
  - Determine the SC for each document
    - $SC(d_1, q) = \log_2 \frac{p_3}{1-p_3} + \log_2 \frac{1-u_3}{u_3} = \log_2 \frac{1-\epsilon}{\epsilon}$
    - $SC(d_2, q) = -\infty$
    - $SC(d_3, q) = \log_2 \frac{p_2}{1-p_2} + \log_2 \frac{1-u_2}{u_2} = 0$
    - $SC(d_4, q) = \log_2 \frac{p_1}{1-p_1} + \log_2 \frac{1-u_1}{u_1} + \log_2 \frac{p_2}{1-p_2} + \log_2 \frac{1-u_2}{u_2} = \log_2 \frac{1-\epsilon}{\epsilon}$
  - Rank the documents and consider the first top-k as relevant (in case of tie keep documents ordering
    - $d_1 > d_4 > d_3 > d_2$
    - Relevant documents = $\{d_1, d_4\} \rightarrow$ Convergence reached

Probabilistic model: conclusions

- Pros
  - Documents are ranked in decreasing order of probability of being relevant
  - It includes a mechanism for relevance feedback

- Cons
  - The need to guess the initial separation of documents into relevant and irrelevant
  - It does not take into account the frequency with which a term occurs inside a document
  - The assumption of independence of index terms