The document ranking problem

- Have a collection of documents
- User issues a query
- A list of documents needs to be returned
- **Ranking method is core of an IR system:**
  - **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(relevant | document$_i$, query)

Recall a few probability basics

- For events *a* and *b:*
- Bayes' Rule

$$p(a,b) = p(a \cap b) = p(a \mid b)\,p(b) = p(b \mid a)\,p(a)$$

$$p(\bar{a} \mid b)\,p(b) = p(b \mid \bar{a})\,p(\bar{a})$$

$$\underbrace{p(a \mid b)}_{\text{Posterior}} = \frac{p(b \mid a)\,p(a)}{p(b)} = \frac{p(b \mid a)\,\overbrace{p(a)}^{\text{Prior}}}{\sum_{x=a,\bar{a}} p(b \mid x)\,p(x)}$$

- Odds:

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

Probability Ranking Principle (PRP)

Probability Ranking Principle (PRP)

Let *x* be a document in the collection.
Let *R* represent **relevance** of a document w.r.t. given (fixed) query
Let *NR* represent **non-relevance.** R={0,1} vs. NR/R

Need to find **p(*R*|*x*)** ---- probability that a document *x* is **relevant.**

$$p(R \mid x) = \frac{p(x \mid R)\,p(R)}{p(x)}$$

$$p(NR \mid x) = \frac{p(x \mid NR)\,p(NR)}{p(x)}$$

$$p(R \mid x) + p(NR \mid x) = 1$$

p(*R*),p(*NR*)
prior probability of retrieving a (non) relevant document

p(*x*|*R*), p(*x*|*NR*)
probability that if a relevant (non-relevant) document is retrieved, it is *x*.

Probability Ranking Principle (续)

- Simple case: no selection costs or other utility concerns that would differentially weight errors
- ***Bayes' Optimal Decision Rule***
  - ***x*** is **relevant** iff $p(R|x) > p(NR|x)$
- **PRP** in action: **Rank** all documents by $p(R|x)$
- 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 (续)

- How do we compute all those probabilities?
  - Do not know exact probabilities, have to use estimates
  - Binary Independence Retrieval (**BIR**) 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 things like tf, df, and length influence your judgments about document relevance?
    - One answer is the Okapi formulae (S. Robertson)
- Combine to find document relevance probability
- Order documents by decreasing probability

Binary Independence Model (BIM)

Binary Independence Model (BIM)

- Traditionally used in conjunction with PRP
- **"Binary" = Boolean**:
  - documents are represented as binary incidence vectors of terms (cf. lecture 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 same vector
- Bernoulli Naive Bayes model (cf. text categorization!)

Binary Independence Model (续)

- 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:

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

Binary Independence Model (续)

$$O(R \mid q, \vec{x}) = \frac{p(R \mid q, \vec{x})}{p(NR \mid q, \vec{x})} = \underbrace{\frac{p(R \mid q)}{p(NR \mid q)}}_{\text{Constant for a given query}} \cdot \underbrace{\frac{p(\vec{x} \mid R, q)}{p(\vec{x} \mid NR, q)}}_{\text{Needs estimation}}$$

- Using **Independence** Assumption:

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

- So : $O(R \mid q, d) = O(R \mid q) \cdot \prod_{i=1}^{n} \frac{p(x_i \mid R, q)}{p(x_i \mid NR, q)}$

Binary Independence Model (续)

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

- Since x_i is either 0 or 1:

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

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

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

Binary Independence Model (续)

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

All matching terms

Non-matching query terms

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

All matching terms

All query terms

Binary Independence Model (续)

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

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 (续)

- All boils down to computing RSV.

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

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

Binary Independence Model (续)

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

Documens	Relevant	Non-Relevant	Total
Xi=1	*s*	*n-s*	*n*
Xi=0	*S-s*	*N-n-S+s*	*N-n*
Total	*S*	*N-S*	*N*

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

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

For now, assume no zero terms. More next lecture.

PRP and BIR

- 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 only can derive somewhat inferior term weights

Probabilistic Relevance Feedback

1. Guess a preliminary probabilistic description of ***R*** and use it to retrieve a first set of documents **V**, as above.
2. Interact with the user to refine the description : learn some definite members of R and NR
3. Reestimate p_i and r_i on the basis of these
   - 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 ***R***.

Estimation – key challenge

If non-relevant documents are approximated by the whole collection, then

- r_i = n_i / N
  - r_i : prob. of occurrence in non-relevant documents for query
  - **log (1– r_i)/r_i = log (N– n_i)/ n_i ≈ log N/n_i** = IDF
- p_i can be estimated in various ways:
  - p_i : probability of occurrence in relevant documents
  - from relevant documents if know some
    - Relevance weighting can be used in feedback loop
  - constant (Croft and Harper combination match)
    - then just get idf weighting of terms
  - proportional to prob. of occurrence in collection
    - more accurately, to log of this (Greiff, SIGIR 1998)

Iteratively estimating p_i

1. Assume that p_i constant over all x_i in query
   - 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
     - note: now a bit like **tf.idf**
3. 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

Probabilistic Model (summary)

- Definition
  - The index term weight : all binary , $w_{ij} \in \{0,1\}$, $w_{iq} \in \{0,1\}$
  - Query **q** : a subset of index terms
  - R: the set of relevant documents known (or initially guessed)
  - $\overline{R}$: the complement of **R** (the set of non-relevant documents)
  - $P(R|d_j)$: the probability that d_j is relevant to **q**
  - $P(\overline{R}|d_j)$: the probability that d_j is non-relevant to **q**
  - The similarity ***sim(d$_j$,q)*** of d_j to **q** is defined as the ratio:

$$sim(d_j, q) \sim \sum_{i=1}^{t} w_{iq} \times w_{ij} \times \left(\log \frac{P(x_i \mid R)}{1 - P(x_i \mid R)} + \log \frac{1 - P(x_i \mid \overline{R})}{P(x_i \mid \overline{R})}\right)$$

Assumptions

$$P(x_i \mid R) = 0.5, P(x_i \mid \overline{R}) = \frac{n_i}{N}$$

$$P(x_i \mid R) = \frac{V_i}{V}, P(x_i \mid \overline{R}) = \frac{n_i - V_i}{N - V}$$

$$P(x_i \mid R) = \frac{V_i + 0.5}{V + 1}, P(kx_i \mid \overline{R}) = \frac{n_i - V + 0.5_i}{N - V + 1}$$

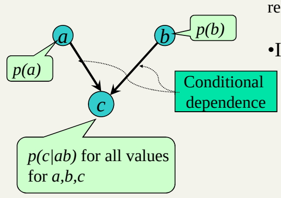
$$P(x_i \mid R) = \frac{V_i + \frac{n_i}{N}}{V + 1}, P(x_i \mid \overline{R}) = \frac{n_i - V_i + \frac{n_i}{N}}{N - V + 1}$$

Bayesian Networks for Text Retrieval
(Turtle and Croft 1990)

- Standard probabilistic model assumes
  - can't estimate P(R|D,Q)
  - Instead assume independence and use P(D|R)
- But maybe can with a Bayesian network
- What is a Bayesian network?
  - A directed acyclic graph
  - Nodes
    - Events or Variables
      - Assume values.
      - For our purposes, all Boolean
  - Links
    - model direct dependencies between nodes

Bayesian Networks

a,b,c - propositions (events).

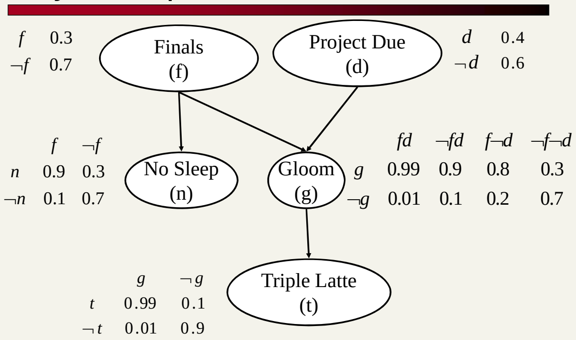


• Bayesian networks model causal relations between events

•Inference in Bayesian Nets:
  •Given probability distributions for roots and conditional probabilities can compute a priori *probability* of any instance
  • Fixing assumptions (e.g., *b* was observed) will cause re-computation of probabilities

For more information see:
R.G. Cowell, A.P. Dawid, S.L. Lauritzen, and D.J. Spiegelhalter. 1999. ***Probabilistic Networks and Expert Systems***. Springer Verlag.
J. Pearl. 1988. ***Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.*** Morgan-Kaufman.

Toy Example



f	0.3
$\neg f$	0.7

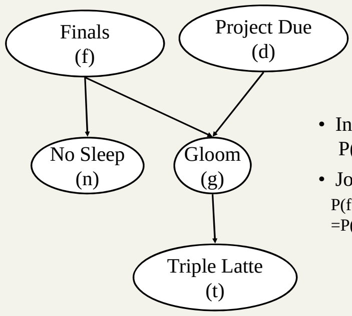
d	0.4
$\neg d$	0.6

	f	$\neg f$
n	0.9	0.3
$\neg n$	0.1	0.7

	fd	$\neg fd$	$f\neg d$	$\neg f\neg d$
g	0.99	0.9	0.8	0.3
$\neg g$	0.01	0.1	0.2	0.7

	g	$\neg g$
t	0.99	0.1
$\neg t$	0.01	0.9

Independence Assumptions

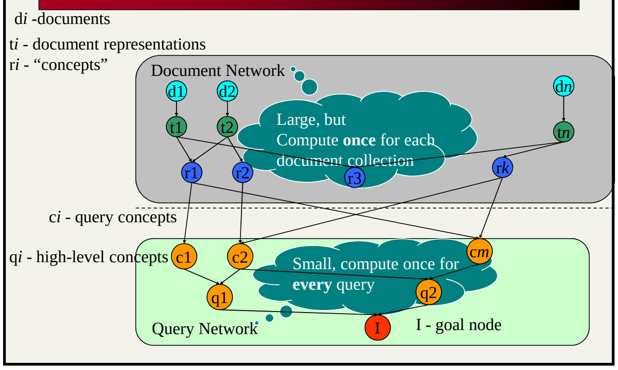


- Independence assumption: P(t|g, f)=P(t|g)
- Joint probability
  P(f d n g t)
  =P(f) P(d) P(n|f) P(g|f d) P(t|g)

Model for Text Retrieval

- Goal
  - given a user's information need (evidence)
  - find probability a doc satisfies need
- Retrieval model
  - Model docs in a *document network*
  - Model information need in a *query network*

Bayesian Nets for IR: Idea

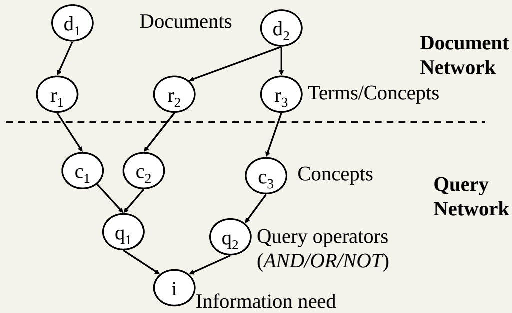


Bayesian Nets for IR

- Construct Document Network (once !)

- For each query
  - Construct best Query Network
  - Attach it to Document Network
  - Find subset of **d_i's** which maximizes the probability value of node **I** (best subset).
  - Retrieve these **d_i's** as the answer to query.

Bayesian nets for text retrieval



Link matrices and probabilities

- Prior doc probability ***P*(*d*) = 1/*n***
- ***P*(*r*|*d*)**
  - within-document term frequency
  - ***tf* × *idf*** - based
- ***P*(*c*|*r*)**
  - 1-to-1
  - thesaurus
- ***P*(*q*|*c*)**
  - canonical forms of query operators
  - always use things like AND and NOT
    - never store a full CPT*

*conditional probability table

Example: *"reason trouble –two"*