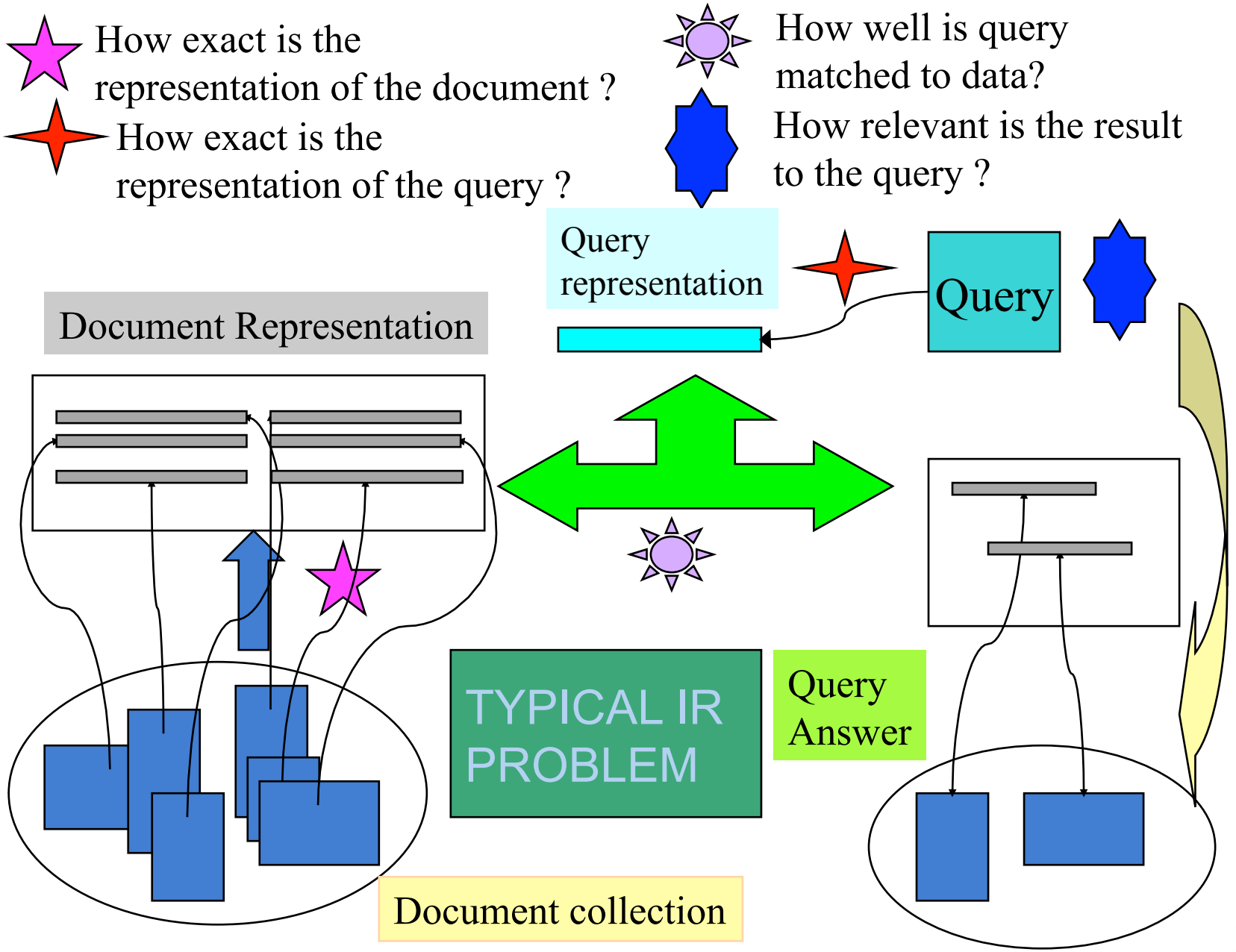
Multimedia Information Extraction and Retrieval

Probabilistic Information Retrieval

Ralf Moeller
Hamburg Univ. of Technology

Acknowledgements

- Slides taken from:
  - Introduction to Information Retrieval
    Christopher Manning and Prabhakar Raghavan


How exact is the representation of the document ?
How exact is the representation of the query ?
How well is query matched to data?
How relevant is the result to the query ?
Query representation
Query
Document Representation
TYPICAL IR PROBLEM
Query Answer
Document collection

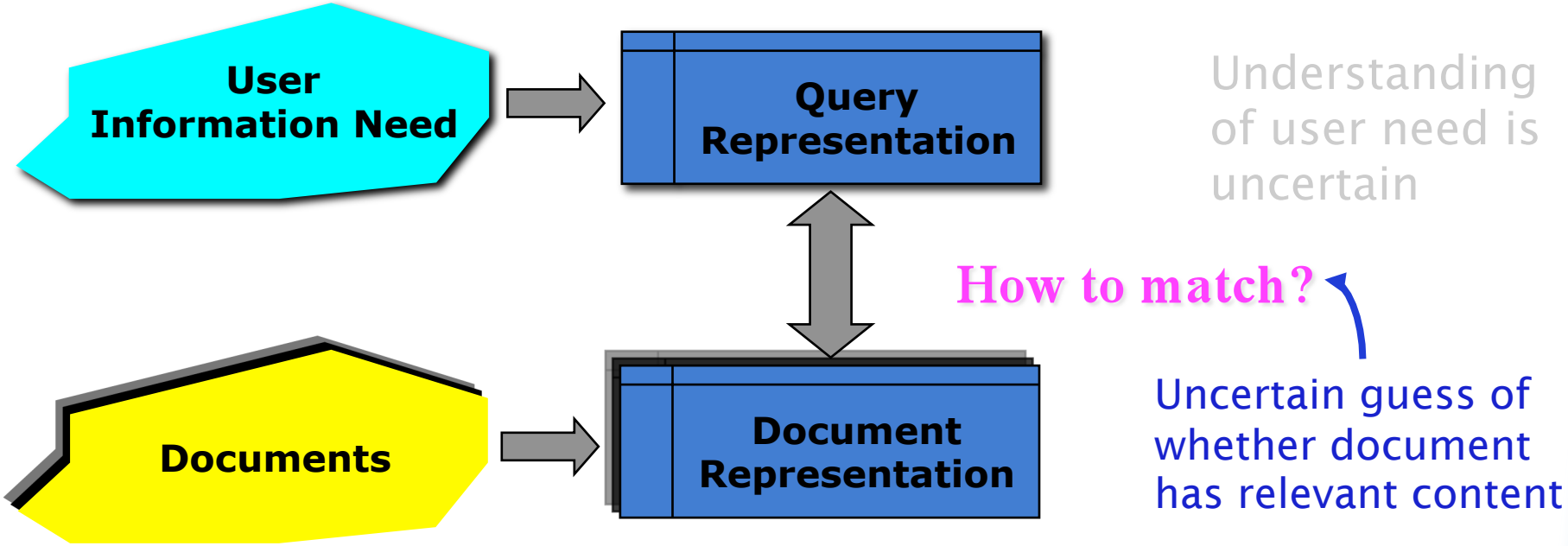
Recall and Precision

$$\text{precision} = \frac{\text{\# relevant documents retrieved}}{\text{\# documents retrieved}}$$

$$\text{recall} = \frac{\text{\# relevant documents retrieved}}{\text{\# relevant documents in collection}}$$

- precision can be directly observed
- recall can only be approximated
  - query expansion
  - pooling of answers from different systems
  - (known item retrieval)

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?

Why use probabilities ?

- Information Retrieval deals with uncertain information
- Probability theory seems to be the most natural way to quantify uncertainty

Probabilistic Approaches to IR

- Probability Ranking Principle (Robertson, 70ies; Maron, Kuhns, 1959)
- Information Retrieval as Probabilistic Inference (van Rijsbergen & co, since 70ies)
- Probabilistic Indexing (Fuhr & Co.,late 80ies-90ies)
- Bayesian Nets in IR (Turtle, Croft, 90ies)
- Probabilistic Logic Programming in IR (Fuhr & co, 90ies)

Success : varied

Next: Probability Ranking Principle

Probability Ranking Principle

- Collection of Documents
- User issues a query
- A set of documents needs to be returned
- **Question: In what order to present documents to user ?**

Probability Ranking Principle

- **Question: In what order to present documents to user ?**
- Intuitively, want the "best" document to be first, second best – second, etc…
- Need a formal way to judge the "goodness" of documents w.r.t. queries.
- **Idea: Probability of relevance of the document w.r.t. query**

Probability Ranking Principle

If a reference retrieval system's **response to each request** is *a ranking of the documents in the collections* in ***order of decreasing probability of usefulness*** to the user who submitted the request ...

... where the probabilities are estimated as accurately as possible on the basis of whatever data made available to the system for this purpose ...

... then the **overall effectiveness** of the system to its users **will be the best** that is obtainable on the basis of that data.
W.S. Cooper

Let us remember Probability Theory

- Bayesian probability formulas

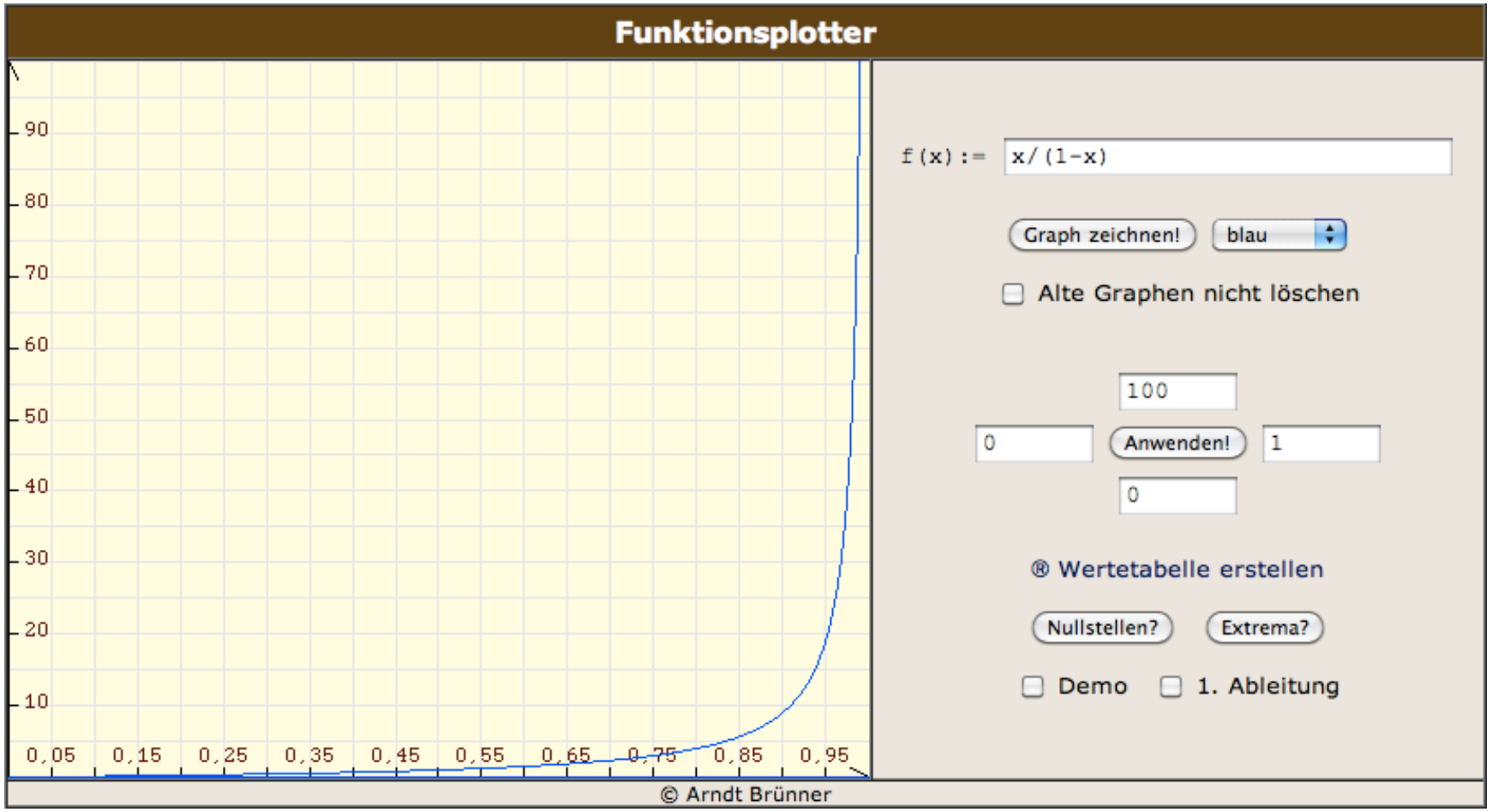
$$p(a \mid b)\,p(b) = p(a \cap b) = p(b \mid a)\,p(a)$$

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

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

- Odds: $O(y) = \dfrac{p(y)}{p(\overline{y})} = \dfrac{p(y)}{1 - p(y)}$

Odds vs. Probabilities



Probability Ranking Principle

Let x be a document in the retrieved collection.
Let R represent **relevance** of a document w.r.t. given (fixed) query and let NR represent **non-relevance.**

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

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

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

$p(R)$,$p(NR)$ - prior probability of retrieving a relevant or non-relevant document, respectively

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

Probability Ranking Principle

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

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

Ranking Principle (Bayes' Decision Rule):

If p(*R*|*x*) > p(*NR*|*x*) then *x* is relevant,
otherwise *x* is not relevant

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

Probability Ranking Principle

Claim: ***PRP minimizes the average probability of error***

$$p(error \mid x) = \begin{cases} p(R \mid x) & \text{If we decide } \textit{NR} \\ p(NR \mid x) & \text{If we decide } \textit{R} \end{cases}$$

$$p(error) = \sum_{x} p(error \mid x)\, p(x)$$

p(error) is minimal when all *p(error|x)* are minimimal.
Bayes' decision rule minimizes each *p(error|x)*.

Probability Ranking Principle

- More complex case: retrieval costs.
  - *C* – cost of retrieval of relevant document
  - *C'* – cost of retrieval of non-relevant document
  - let *d*, be a document
- Probability Ranking Principle: if

$$C \cdot p(R \mid d) + C' \cdot (1 - p(R \mid d)) \leq C \cdot p(R \mid d') + C' \cdot (1 - p(R \mid d'))$$

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

PRP: Issues (Problems?)

- How do we compute all those probabilities?
  - Cannot compute exact probabilities, have to use estimates.
  - **Binary Independence Retrieval (BIR)**
    - See below
- Restrictive assumptions
  - "Relevance" of each document is independent of relevance of other documents.
  - Most applications are for Boolean model.

Next: Probabilistic Inference

Probabilistic Inference

- Represent each document as a collection of sentences (formulas) in some logic.
- Represent each query as a sentence in the same logic.
- Treat Information Retrieval as a **process of inference**: document *D* is relevant for query *Q* if $p(D \rightarrow Q)$ is high in the inference system of selected logic.

Probabilistic Inference: Notes

- $p(D \rightarrow Q)$ is the probability that the description of the document in the logic implies the description of the query.
  - $\rightarrow$ is not necessarily material implication:

$$p(A \rightarrow B) = \frac{p(A \wedge B)}{p(A)} \neq p(\neg A \vee B)$$

- Reasoning to be done in some kind of probabilistic logic

Probabilistic Logic

- Uncertainty: statements are true or false. But, due to lack of knowledge we can only estimate to which probability/possibility/necessity degree they are true or false
  - For instance, a bird flies or does not fly. The probability/possibility/necessity degree that it flies is 0.83
- Usually we have a possible world semantics with a distribution over possible worlds:

$$W = \{I \text{ classical interpretation}\}, \quad I(\varphi) \in \{0,1\}$$

$$\mu\colon W \to [0,1], \quad \mu(I) \in [0,1]$$

$$Pr(\phi) = \sum_{I \models \phi} \mu(I)$$

$$Poss(\phi) = \sup_{I \models \phi} \mu(I)$$

$$Necc(\phi) = \inf_{I \not\models \phi} \mu(I) = 1 - Poss(\neg\phi)$$

From: Th. Lukasievicz, U. Straccia

Probabilistic Inference: Roadmap

- Describe your own probabilistic logic/ inference system
  - document / query representation
  - inference rules
- Given query Q compute $p(D \rightarrow Q)$ for each document D
- Select the "winners"

Probabilistic Inference: Pros/Cons

Pros:

- Flexible: Create-Your-Own-Logic approach
- Possibility for provable properties for PI based IR.
- Another look at the same problem ?

Cons:

- Vague: PI is just a broad framework not a cookbook
- Efficiency:
  - Computing probabilities always hard;
  - Probabilistic Logics are notoriously inefficient (up to being undecidable)

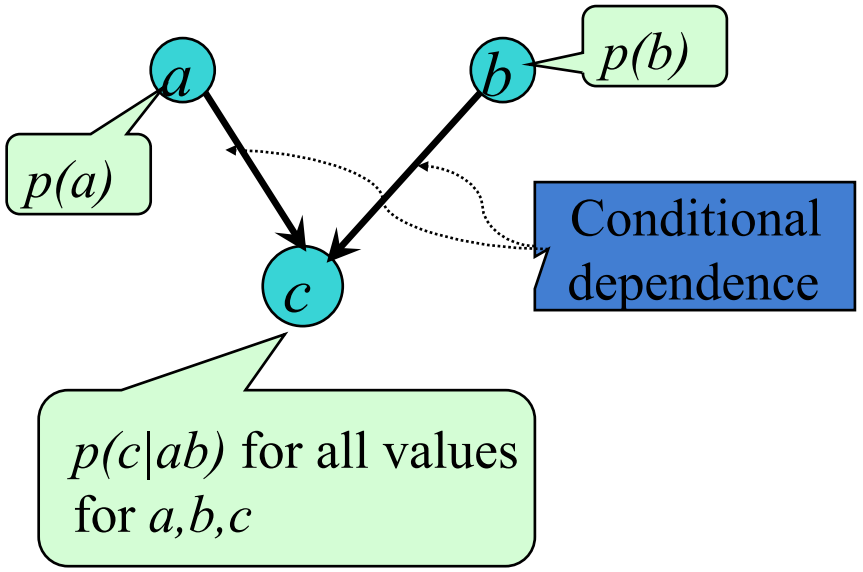
Next: Bayesian Nets in IR

Bayesian Nets in IR

- Bayesian Nets is the most popular way of doing probabilistic inference.
- What is a Bayesian Net ?
- How to use Bayesian Nets in IR?

Bayesian Nets

a,b,c - propositions (events). • Running Bayesian Nets:

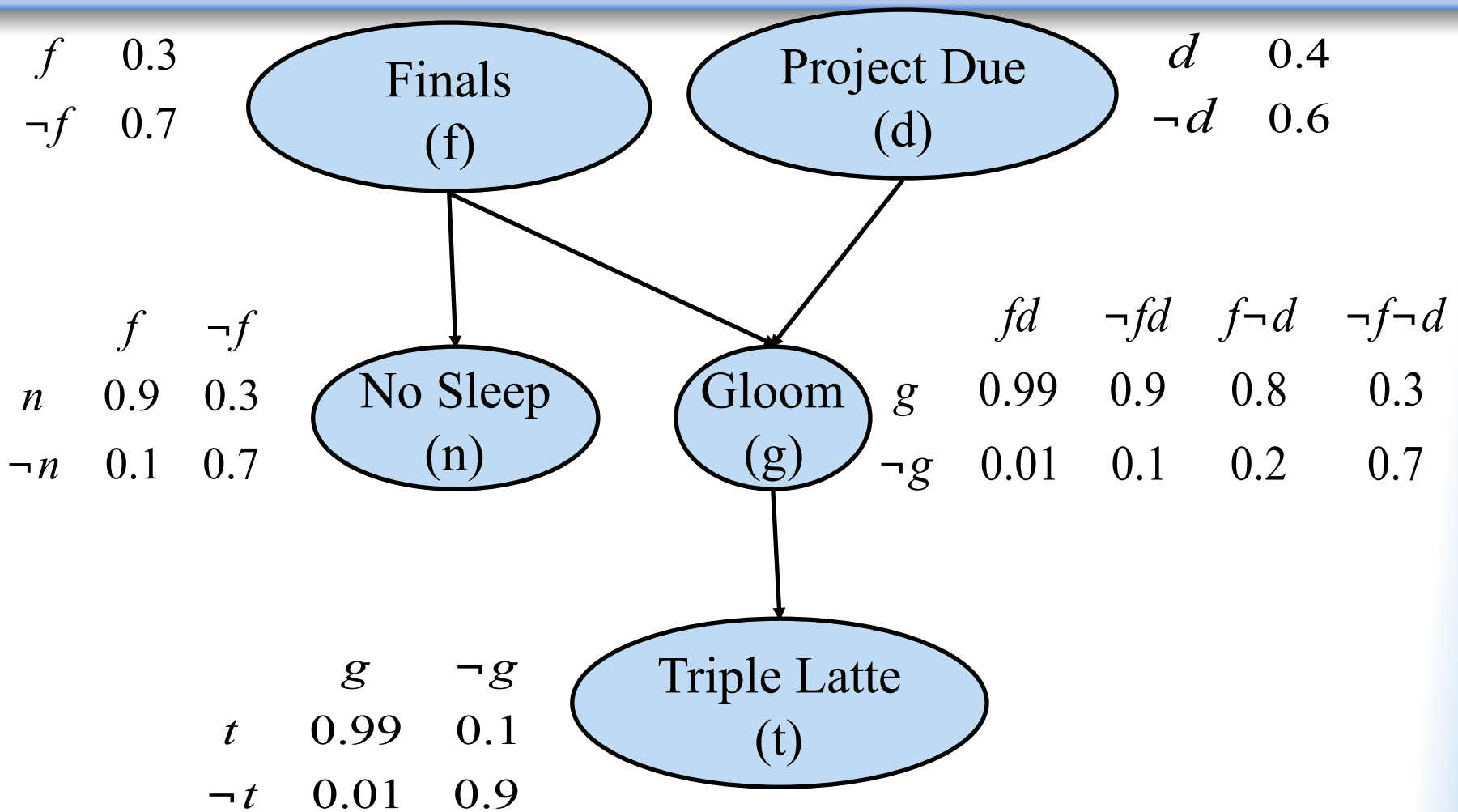


Given probability distributions for roots and conditional probabilities can compute apriori *probability* of any instance

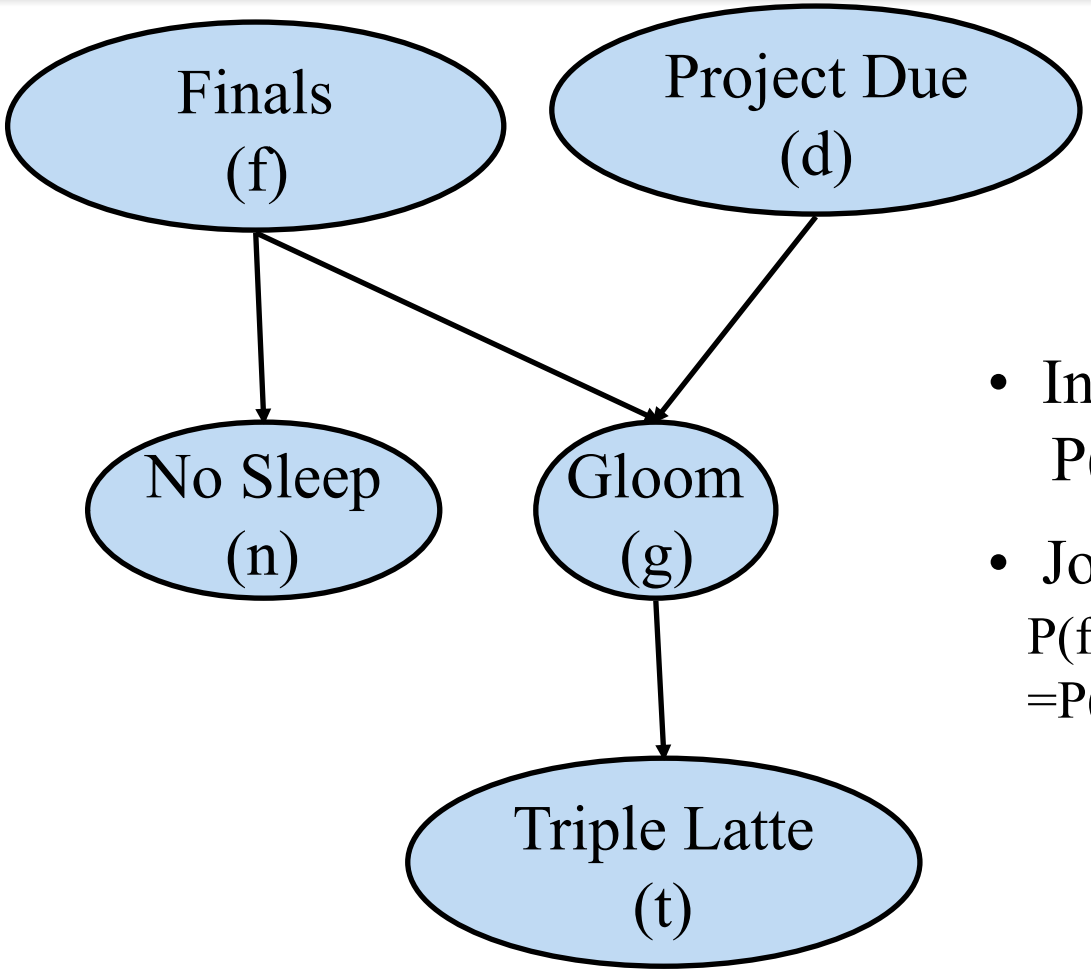
Fixing assumptions (e.g., *b* was observed) will cause recomputation of probabilities

For more information see J. Pearl, "***Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference***", 1988, Morgan-Kaufman.

Toy Example


f 0.3
¬f 0.7
Finals
(f)
Project Due
(d)
d 0.4
¬d 0.6
f ¬f
n 0.9 0.3
¬n 0.1 0.7
No Sleep
(n)
Gloom
(g)
fd ¬fd f¬d ¬f¬d
g 0.99 0.9 0.8 0.3
¬g 0.01 0.1 0.2 0.7
g ¬g
t 0.99 0.1
¬t 0.01 0.9
Triple Latte
(t)

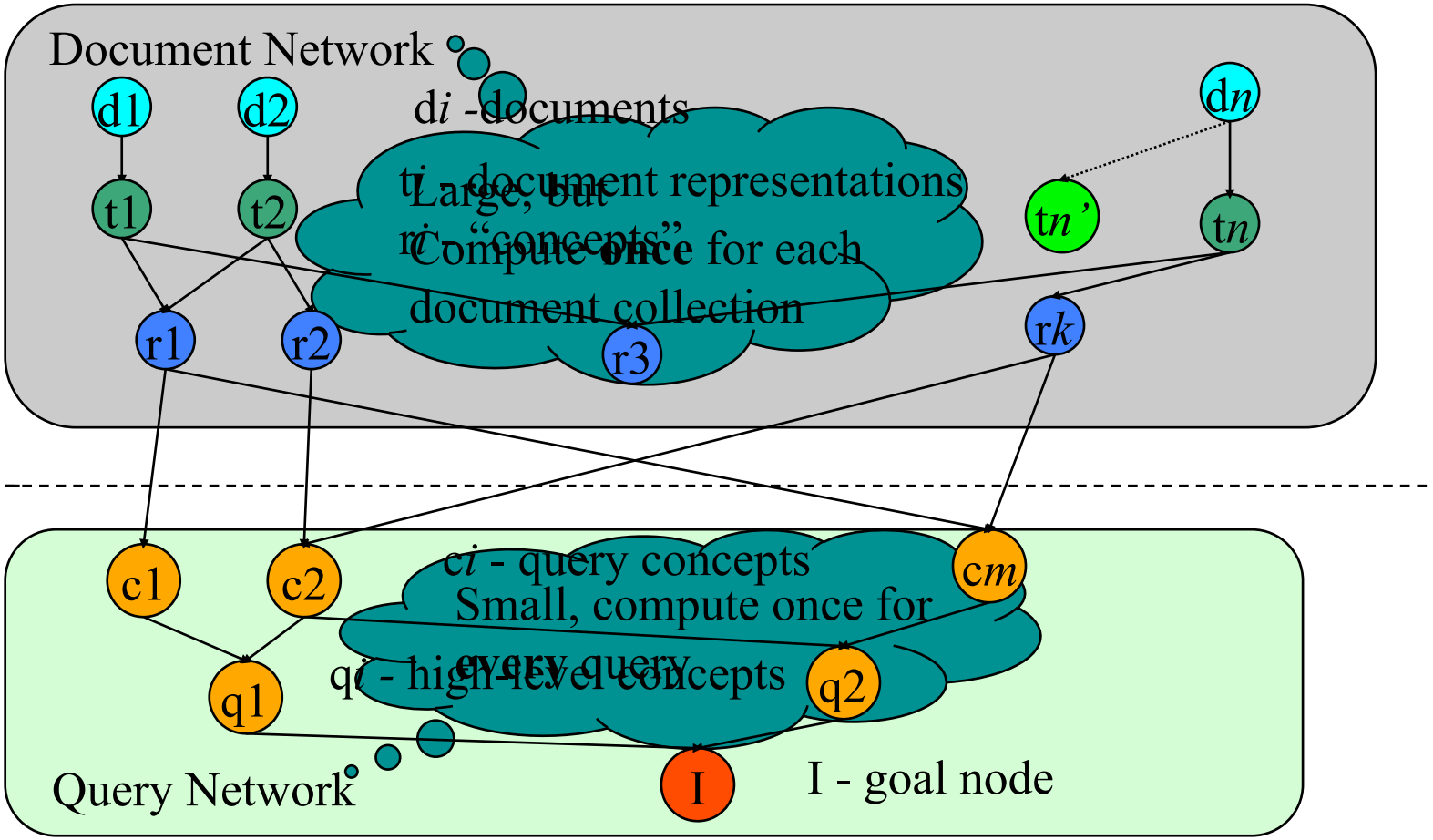
Independence Assumptions



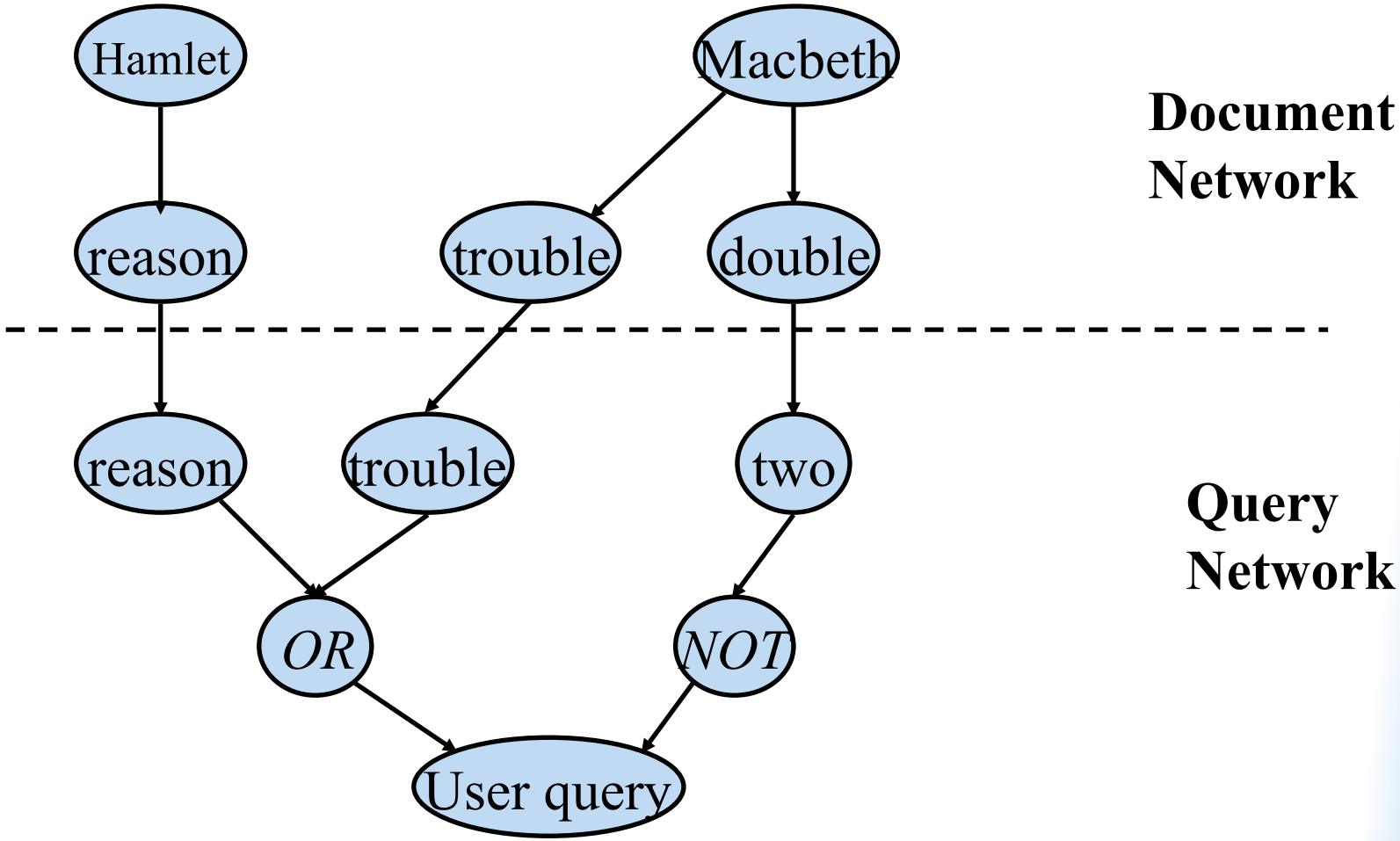
- Independence assumption:
  $P(t|g, f)=P(t|g)$
- Joint probability

  $P(f\ d\ n\ g\ t)$
  $=P(f)\ P(d)\ P(n|f)\ P(g|f\ d)\ P(t|g)$

Bayesian Nets for IR: Idea



Example: *"reason trouble –two"*



Bayesian Nets for IR: Roadmap

- 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 in IR: Pros / Cons

• Pros

- More of a cookbook solution
- Flexible:create-your-own Document (Query) Networks
- Relatively easy to update
- Generalizes other Probabilistic approaches
  - PRP
  - Probabilistic Indexing

• Cons

- Best-Subset computation is NP-hard
  - have to use quick approximations
  - approximated Best Subsets may not contain best documents
- Where Do we get the numbers ?

Next: Probabilistic Logic Programming in IR

Probabilistic LP in IR

- Probabilistic Inference estimates $p(D \rightarrow Q)$ in some probabilistic logic
- Most probabilistic logics are hard
- **Logic Programming:** possible solution
  - logic programming languages are restricted
  - but decidable
- Logic Programs may provide flexibility (write your own IR program)
- Fuhr & Co: Probabilistic Datalog

Probabilistic Datalog: Example

- Sample Program:

```
0.7 term(d1,ir).
0.8 term(d1,db).
0.5 link(d2,d1).
about(D,T):- term(D,T).
about(D,T):- link(D,D1), about(D1,T).
```

- Query/Answer:

```
:- term(X,ir) & term(X,db).
X= 0.56 d1
```

Probabilistic Datalog: Example

- Sample Program:

```
0.7 term(d1,ir).
0.8 term(d1,db).
0.5 link(d2,d1).
about(D,T):- term(D,T).
about(D,T):- link(D,D1), about(D1,T).
```

- Query/Answer:

```
q(X):- term(X,ir).
q(X):- term(X,db).
:-q(X)
X= 0.94 d1
```

Probabilistic Datalog: Example

- Sample Program:

```
0.7 term(d1,ir).
0.8 term(d1,db).
0.5 link(d2,d1).
about(D,T):- term(D,T).
about(D,T):- link(D,D1), about(D1,T).
```

- Query/Answer:

```
:- about(X,db).
X= 0.8 d1;
X= 0.4 d2
```

Probabilistic Datalog: Example

- Sample Program:

```
0.7 term(d1,ir).
0.8 term(d1,db).
0.5 link(d2,d1).
about(D,T):- term(D,T).
about(D,T):- link(D,D1), about(D1,T).
```

- Query/Answer:

```
:- about(X,db)& about(X,ir).
X= 0.56 d1
X= 0.28 d2   # NOT 0.14 = 0.8*0.5*0.7*0.5
```

Probabilistic Datalog: Issues

- Possible Worlds Semantics
- Lots of restrictions (!)
  - all statements are either independent or disjoint
    - not clear how this is distinguished syntactically
  - point probabilities
  - needs to carry a lot of information along to support reasoning because of independence assumption

Next: Relevance Models

Relevance models

- **Given: PRP**
- **Goal: Estimate probability P(R|q,d)**
- Binary Independence Retrieval (BIR):
  - Many documents D – one query q
  - Estimate P(R|q,d) by considering whether d in D is relevant for q
- Binary Independence Indexing (BII):
  - One document d – many queries Q
  - Estimate P(R|q,d) by considering whether a document d is relevant for a query q in Q

Binary Independence Retrieval

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

Binary Independence Retrieval

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

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

Binary Independence Retrieval

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

Constant for each query

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 Retrieval

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

- 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 occuring in the query ($q_i=0$) $p_i = r_i$

Then...

Binary Independence Retrieval

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

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 Retrieval

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

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 Retrieval

- All boils down to computing RSV.

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

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

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

Binary Independence Retrieval

- Estimating RSV coefficients.
- For each term *i* look at the following table:

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

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

Binary Independence Indexing

- "Learning" from queries
  - More queries: better results

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

- ***p(q|x,R)*** *– probability that if document* ***x*** *had been deemed relevant, query* ***q*** *had been asked*
- The rest of the framework is similar to BIR

Binary Independence Indexing vs. Binary Independence Retrieval

•BIR

- Many Documents, One Query
- Bayesian Probability:

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

- Varies: document representation
- Constant: query (representation)

•BII

- One Document, Many Queries
- Bayesian Probability

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

- Varies: query
- Constant: document

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
  - $\log (1- r_i)/r_i = \log (N- n)/n \approx \log N/n$ = IDF!
- p_i (probability of occurrence in relevant documents) can be estimated in various ways:
  - 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)
- We have a nice theoretical foundation of wTD.IDF

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. 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

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

7. Repeat, thus generating a succession of approximations to *R*.

PRP and BIR: The lessons

- Getting reasonable approximations of probabilities is possible.
- Simple methods work only with restrictive assumptions:
  - *term independence*
  - *terms not in query do not affect the outcome*
  - *boolean representation of documents/ queries*
  - *document relevance values are independent*
- Some of these assumptions can be removed

Removing term independence

- In general, index terms aren't independent
- Dependencies can be complex
- van Rijsbergen (1979) proposed model of simple tree dependencies
  - 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

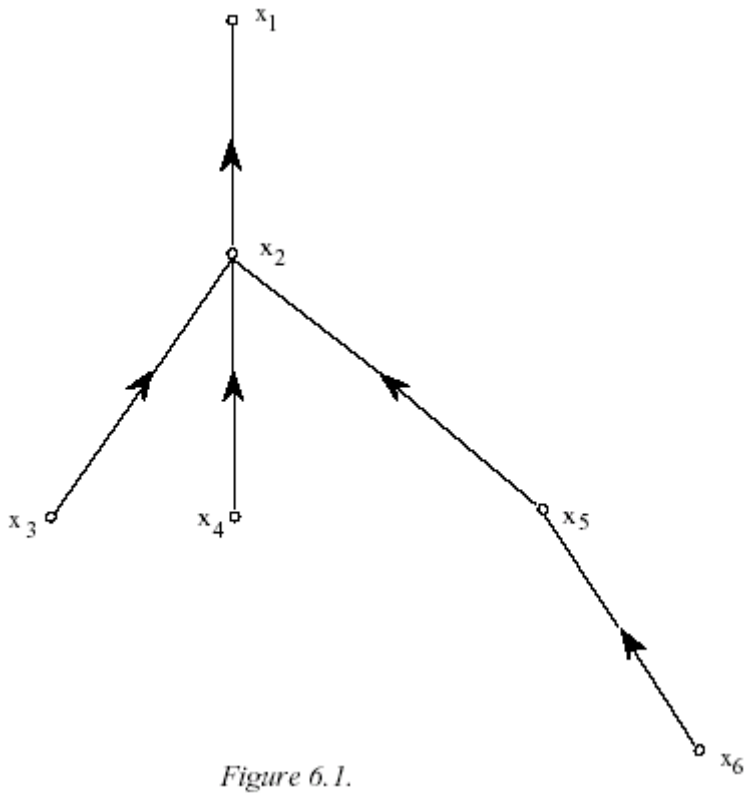


Figure 6.1.

Food for thought

- Think through the differences between standard tf.idf and the probabilistic retrieval model in the first iteration
- Think through the differences between vector space (pseudo) relevance feedback and probabilistic (pseudo) relevance feedback

Good and Bad News

- Standard Vector Space Model
  - Empirical for the most part; success measured by results
  - Few properties provable
- Probabilistic Model Advantages
  - Based on a firm theoretical foundation
  - Theoretically justified optimal ranking scheme
- Disadvantages
  - Making the initial guess to get V
  - Binary word-in-doc weights (not using term frequencies)
  - Independence of terms (can be alleviated)
  - Amount of computation
  - Has never worked convincingly better in practice