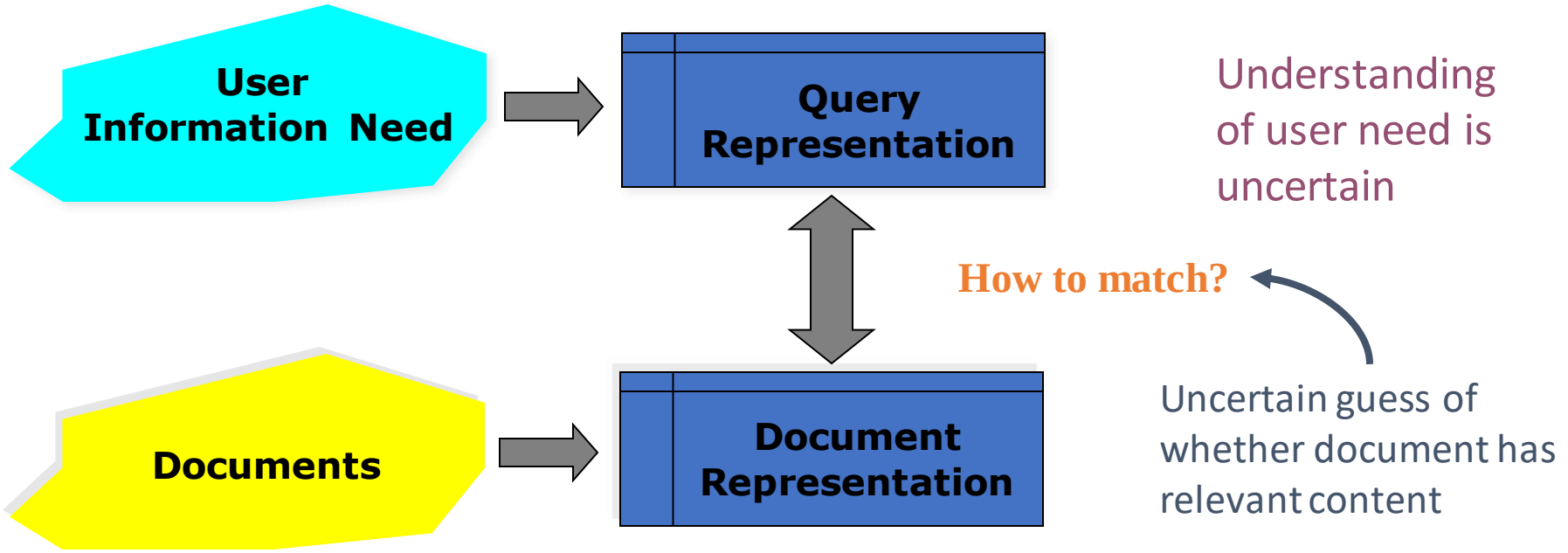
# Probabilistic IR

# 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?*

# vector space vs. probabilistic

- Vector space model: rank 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 non-relevant.*
- Probability theory is arguably a cleaner formalization of what we really want an IR system to do: give relevant documents to the user.

# The document ranking problem

- We have a collection of documents
- User issues a query
- A list of documents needs to be returned
- **Ranking method is the 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**
  - $P(R=1|\text{document}_i, \text{query})$

# Recall a few probability basics

- For events *A* and *B:*
- Bayes' Rule

$$p(A,B) = p(A \cap B) = p(A \mid B)\,p(B) = p(B \mid A)\,p(A)$$

$$p(A \mid B) = \frac{p(B \mid A)\,p(A)}{p(B)} = \frac{p(B \mid A)\,p(A)}{\sum_{X=A,\bar{A}} p(B \mid X)\,p(X)}$$

Prior

Posterior

- Odds:

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

# The Probability Ranking Principle

"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."

- [1960s/1970s] S. Robertson, W.S. Cooper, M.E. Maron; van Rijsbergen (1979:113); Manning & Schütze (1999:538)

# Probability Ranking Principle

Let $x$ represent a document in the collection.
Let $R$ represent **relevance** of a document w.r.t. given (fixed) query and let **R=1** represent relevant and **R=0** not relevant**.**

Need to find $p(R=1|x)$ - probability that a document $x$ is **relevant.**

$$p(R=1|x)=\frac{p(x|R=1)p(R=1)}{p(x)}$$

$$p(R=0|x)=\frac{p(x|R=0)p(R=0)}{p(x)}$$

$$p(R=0|x)+p(R=1|x)=1$$

$p(R=1)$,$p(R=0)$ - prior probability of retrieving a relevant or non-relevant document

$p(x|R=1)$, $p(x|R=0)$ - probability that if a relevant (not relevant) document is retrieved, it is $x$.

# Probability Ranking Principle (PRP)

- Simple case: no selection costs or other utility concerns that would differentially weight errors

- PRP in action: Rank all documents by $p(R=1|x)$

# Probability Ranking Principle

- More complex case: retrieval costs.
  - Let *d* be a document
  - *C* – cost of not retrieving a relevant document
  - *C'* – cost of retrieving a non-relevant document
- Probability Ranking Principle: if

$$C\times p(R=0\mid d)-C\times p(R=1\mid d)\le C\times p(R=0\mid d')-C\times p(R=1\mid d')$$

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

- **We won't further consider cost/utility from now on**

# Probability Ranking Principle

- How do we compute all those probabilities?
  - Do not know exact probabilities, have to use estimates
  - Binary Independence Model (BIM) – which we discuss next – 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

# Binary Independence Model

- Traditionally used in conjunction with PRP
- **"Binary" = Boolean**: documents are represented as binary incidence 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 the same vector

# Binary Independence Model

- 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:

$$O(R \mid q, \vec{x}) = \frac{p(R=1 \mid q, \vec{x})}{p(R=0 \mid q, \vec{x})} = \frac{\dfrac{p(R=1 \mid q)p(\vec{x} \mid R=1, q)}{p(\vec{x} \mid q)}}{\dfrac{p(R=0 \mid q)p(\vec{x} \mid R=0, q)}{p(\vec{x} \mid q)}}$$

# Binary Independence Model

$$O(R \mid q, \vec{x}) = \frac{p(R=1 \mid q, \vec{x})}{p(R=0 \mid q, \vec{x})} = \frac{p(R=1 \mid q)}{p(R=0 \mid q)} \times \frac{p(\vec{x} \mid R=1, q)}{p(\vec{x} \mid R=0, q)}$$

Constant for a given query

Needs estimation

- Using **Independence** Assumption:

$$\frac{p(\vec{x} \mid R=1, q)}{p(\vec{x} \mid R=0, q)} = \prod_{i=1}^{n} \frac{p(x_i \mid R=1, q)}{p(x_i \mid R=0, q)}$$

$$O(R \mid q, \vec{x}) = O(R \mid q) \times \prod_{i=1}^{n} \frac{p(x_i \mid R=1, q)}{p(x_i \mid R=0, q)}$$

# Binary Independence Model

$$O(R \mid q, \vec{x}) = O(R \mid q) \times \prod_{i=1}^{n} \frac{p(x_i \mid R=1, q)}{p(x_i \mid R=0, q)}$$

- Since $x_i$ is either $0$ or $1$:

$$O(R \mid q, \vec{x}) = O(R \mid q) \times \prod_{x_i=1} \frac{p(x_i=1 \mid R=1, q)}{p(x_i=1 \mid R=0, q)} \times \prod_{x_i=0} \frac{p(x_i=0 \mid R=1, q)}{p(x_i=0 \mid R=0, q)}$$

- Let $p_i = p(x_i=1 \mid R=1, q);\ r_i = p(x_i=1 \mid R=0, q);$

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

A term not occurring in the query is equally likely to occur in relevant and non-relevant documents.

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

| | document | relevant (R=1) | not relevant (R=0) |
|---|---|---|---|
| term present | $x_i = 1$ | $p_i$ | $r_i$ |
| term absent | $x_i = 0$ | $(1 - p_i)$ | $(1 - r_i)$ |

# Binary Independence Model

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

All matching terms

$x_i = q_i = 1$

$x_i = 0$, $q_i = 1$

Non-matching query terms

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

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

All matching terms

$x_i = q_i = 1$

$q_i = 1$

All query terms

# Binary Independence Model

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

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

All boils down to computing RSV.

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

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

The $c_i$ are log odds ratios
They function as the term weights in this model

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

# bim retrieval status value (rsv)

Equivalent: rank documents using the log odds ratios for the terms in the query $c_t$:

$$c_t = \log \frac{p_t(1 - r_t)}{r_t(1 - p_t)} = \log \frac{p_t}{(1 - p_t)} - \log \frac{r_t}{1 - r_t}$$

- The odds ratio is the ratio of two odds: (i) the odds of the term appearing if the document is relevant ($p_t/(1 - p_t)$), and (ii) the odds of the term appearing if the document is nonrelevant ($r_t/(1 - r_t)$)
- $c_t = 0$: term has equal odds of appearing in relevant and nonrelevant docs
- $c_t$ positive: higher odds to appear in relevant documents
- $c_t$ negative: higher odds to appear in nonrelevant documents

# avoiding zeros

- If any of the counts is a zero, then the term weight is not well-defined.
- Maximum likelihood estimates do not work for rare events.
- To avoid zeros: add 0.5 to each count or use a different type of smoothing

$$p_i = p(x_i = 1 \mid R = 1, q); \quad r_i = p(x_i = 1 \mid R = 0, q);$$

# Binary Independence Model

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

| Documents | 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:

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

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

# 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

$$\log\frac{1-r_i}{r_i} = \log\frac{N-n-S+s}{n-s} \approx \log\frac{N-n}{n} \approx \log\frac{N}{n} = IDF!$$

# Estimation – key challenge

- $p_i$ (probability of occurrence in relevant documents) cannot be approximated as easily
- $p_i$ can be estimated in various ways:
  - Use the frequency of term occurrence in known relevant documents if we know some
    - Relevance weighting can be used in a feedback loop
  - constant (Croft and Harper combination match) – then just get idf weighting of terms (with $p_i=0.5$)

$$RSV = \sum_{x_i=q_i=1} \log \frac{N}{n_i}$$

  - proportional to prob. of occurrence in collection
    - Estimate $p_i = \frac{1}{3} + \frac{2}{3}\frac{df_i}{N}$
  - Pseudo-relevance feedback

# Probabilistic Relevance Feedback

1. Guess a preliminary probabilistic description of *R=1* documents and use it to retrieve a first set of documents
2. Interact with the user to refine the description: learn some definite members with R=1 and R=0
3. Reestimate $p_i$ and $r_i$ on the basis of these
   - Or can combine new information with original guess (use Bayesian prior):

$$p_i^{(2)} = \frac{|V_i| + \kappa p_i^{(1)}}{|V| + \kappa}$$

$\kappa$ is prior weight

4. Repeat, thus generating a succession of approximations to relevant documents

# Iteratively estimating $p_i$ *and* $r_i$ (= Pseudo-relevance feedback)

1. Assume that $p_i$ *is* constant over all $x_i$ in query and $r_i$ as before
   - $p_i$ = 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
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

# Thanks