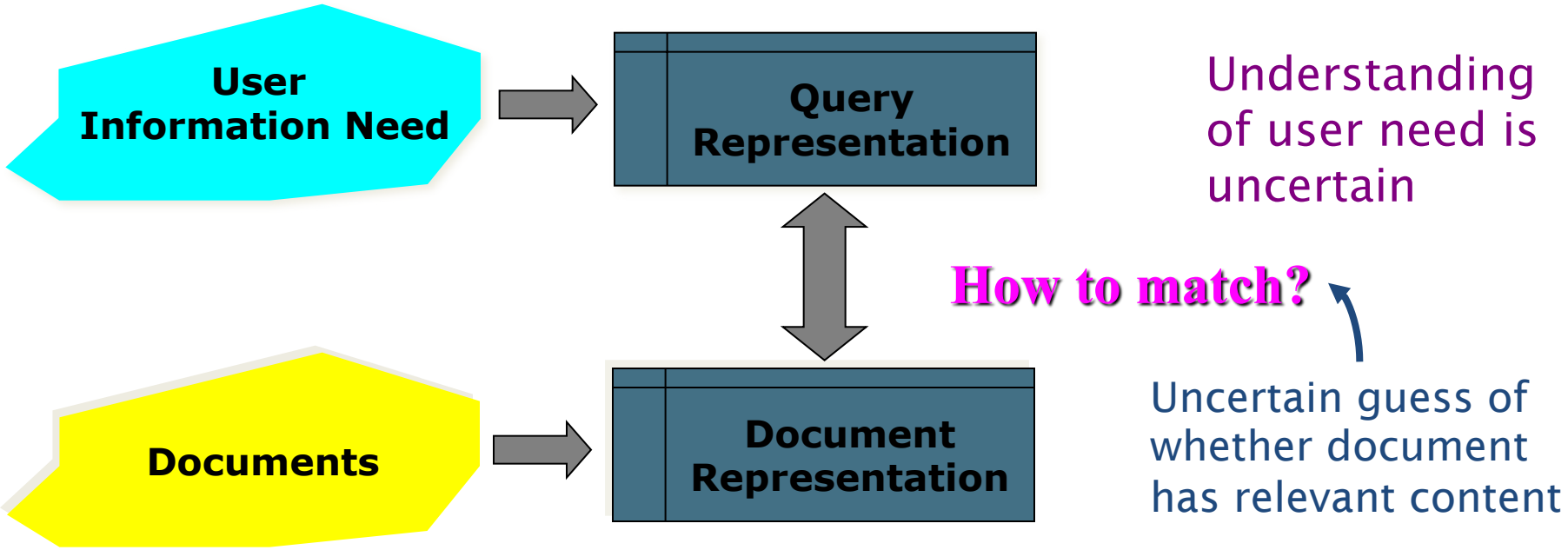
Introduction to

Information Retrieval

Lecture 9: Probabilistic Model & Language Model

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?*

Probabilistic IR topics

- Classical probabilistic retrieval model
  - Probability ranking principle, etc.
- (Naïve) Bayesian Text Categorization
- Bayesian networks for text retrieval
- Language model approach to IR
  - An important emphasis in recent work

- *Probabilistic methods are one of the oldest but also one of the currently hottest topics in IR.*
  - *Traditionally: neat ideas, but they've never won on performance. It may be different now.*

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 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

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

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

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

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

R={0,1} vs. NR/R

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

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

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

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

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

$p(x|R)$, $p(x|NR)$ - probability that if a relevant (non-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
- ***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

- More complex case: retrieval costs.
  - Let *d* be a document
  - *C* - cost of retrieval of relevant document
  - *C'* - cost of retrieval of non-relevant 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**

- **We won't further consider loss/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 Retrieval (BIR) – which we discuss later today – 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

$$MMR \stackrel{\text{def}}{=} Arg \max_{D_i \in R \setminus S} \left[\lambda (Sim_1(D_i, Q) - (1-\lambda) \max_{D_j \in S} Sim_2(D_i, D_j)) \right]$$

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

Probabilistic Ranking

Basic concept:

"For a given query, ***if*** we know some documents that are relevant, terms that occur in those documents should be given greater weighting in searching for other relevant documents.

By making assumptions about the distribution of terms and applying Bayes Theorem, it is possible to derive weights theoretically."

Van Rijsbergen

Binary Independence Model

- 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

- 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

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

Constant for a given 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 Model

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

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

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

Then...

This can be changed (e.g., in relevance feedback)

Binary Independence Model

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

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

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

Constant for each query

Only quantity to be estimated for rankings

- Retrieval Status Value:

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

$$= \sum_{x_i = q_i = 1} \left(\log(odds(p_i)) - \log(odds(r_i)) \right) = \sum_{x_i = q_i = 1} \left(\text{logit}(p_i) - \text{logit}(r_i) \right)$$

Binary Independence Model

- All boils down to computing RSV.

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

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

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

Binary Independence Model

- Estimating RSV coefficients.
- 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: $p_i \approx \frac{s}{S} \qquad r_i \approx \frac{(n-s)}{(N-S)}$

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

However, these estimates could be 0.

Add ½ Smoothing

- Add ½ to each of the center four cells.

Documents	Relevant	Non-Relevant	Total
$X_i = 1$	s+½	n-s+½	n+1
$X_i = 0$	S-s+½	N-n-S+s+½	N-n+1
Total	S+1	N-S+1	N+2

$$c_i \approx K(N,n,S,s) = \log \frac{(s+1/2)/(S-s+1/2)}{(n-s+1/2)/(N-n-S+s+1/2)}$$

Example /1

- Query = $\{x_1, x_2\}$
- $O(R=1|D_3, q)$

Doc	Judgment	x_1	x_2	x_3
D_1	R	1	1	1
D_2	R	0	0	1
D_3	R	1	0	0
D_4	NR	1	0	1
D_5	NR	0	1	1

Example /2

- Estimate p_i and r_i

Doc	Judgment	x_1	x_2	x_3
D_1	R	1	1	1
D_2	R	0	0	1
D_3	R	1	0	0
D_4	NR	1	0	1
D_5	NR	0	1	1

in fact, tf-idf can be deemed as the cross-entropy

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 – 0.5) – then just get idf weighting of terms
  - proportional to prob. of occurrence in collection
    - more accurately, to log of this (Greiff, SIGIR 1998)

optional

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

optional

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*.

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

Okapi BM25

- Heuristically extend the BIR to include information of term frequencies, document length, etc.

caps the contribution of *tf*

$$RSVd = \sum_{t \in q} \left(\log \frac{N}{df_t} \right) \cdot \frac{(k_1+1)tf_{t,d}}{k_1\left((1-b)+b \cdot \frac{L_d}{L_{ave}}\right)+tf_{t,d}} \cdot \frac{(k_3+1)tf_{t,q}}{k_3+tf_{t,q}}$$

idf

Normalized term freq (doc)

Normalized term freq (query)

- Typically, $k_1, k_3 \in [1.2, 2.0], b = 0.75$

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

Resources

S. E. Robertson and K. Spärck Jones. 1976. Relevance Weighting of Search Terms. *Journal of the American Society for Information Sciences* 27(3): 129–146.

C. J. van Rijsbergen. 1979. *Information Retrieval.* 2nd ed. London: Butterworths, chapter 6. [Most details of math] http://www.dcs.gla.ac.uk/Keith/Preface.html

N. Fuhr. 1992. Probabilistic Models in Information Retrieval. *The Computer Journal*, 35(3),243–255. [Easiest read, with BNs]

F. Crestani, M. Lalmas, C. J. van Rijsbergen, and I. Campbell. 1998. Is This Document Relevant? ... Probably: A Survey of Probabilistic Models in Information Retrieval. *ACM Computing Surveys* 30(4): 528–552.

http://www.acm.org/pubs/citations/journals/surveys/1998-30-4/p528-crestani/

[Adds very little material that isn't in van Rijsbergen or Fuhr]

Resources

H.R. Turtle and W.B. Croft. 1990. Inference Networks for Document Retrieval. *Proc. ACM SIGIR*: 1-24.

E. Charniak. Bayesian nets without tears. *AI Magazine* 12(4): 50-63 (1991). http://www.aaai.org/Library/Magazine/Vol12/12-04/vol12-04.html

D. Heckerman. 1995. A Tutorial on Learning with Bayesian Networks. Microsoft Technical Report MSR-TR-95-06 http://www.research.microsoft.com/~heckerman/

N. Fuhr. 2000. Probabilistic Datalog: Implementing Logical Information Retrieval for Advanced Applications. *Journal of the American Society for Information Science* 51(2): 95–110.

R. K. Belew. 2001. *Finding Out About: A Cognitive Perspective on Search Engine Technology and the WWW*. Cambridge UP 2001.

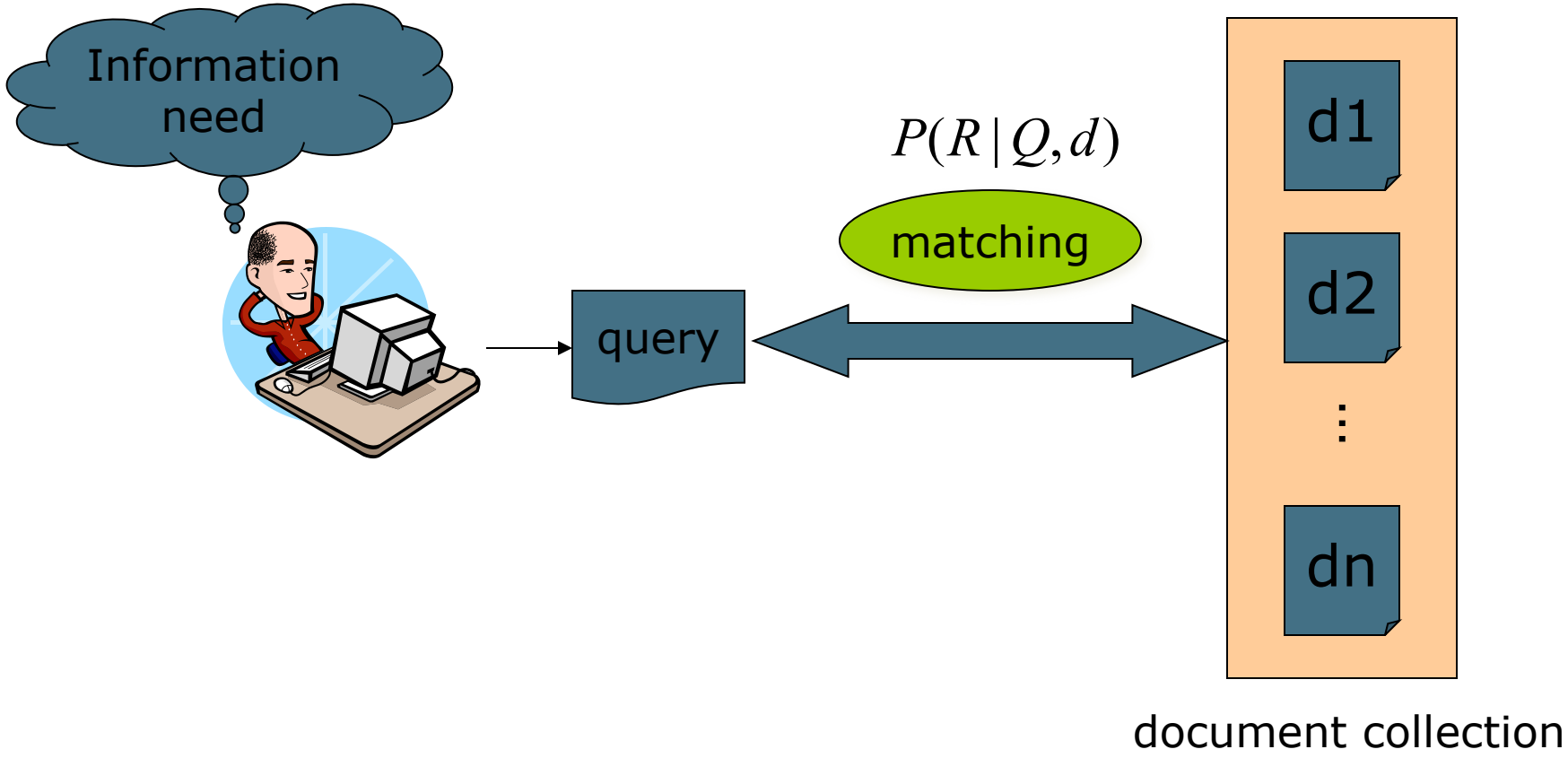
MIR 2.5.4, 2.8

LANGUAGE MODEL

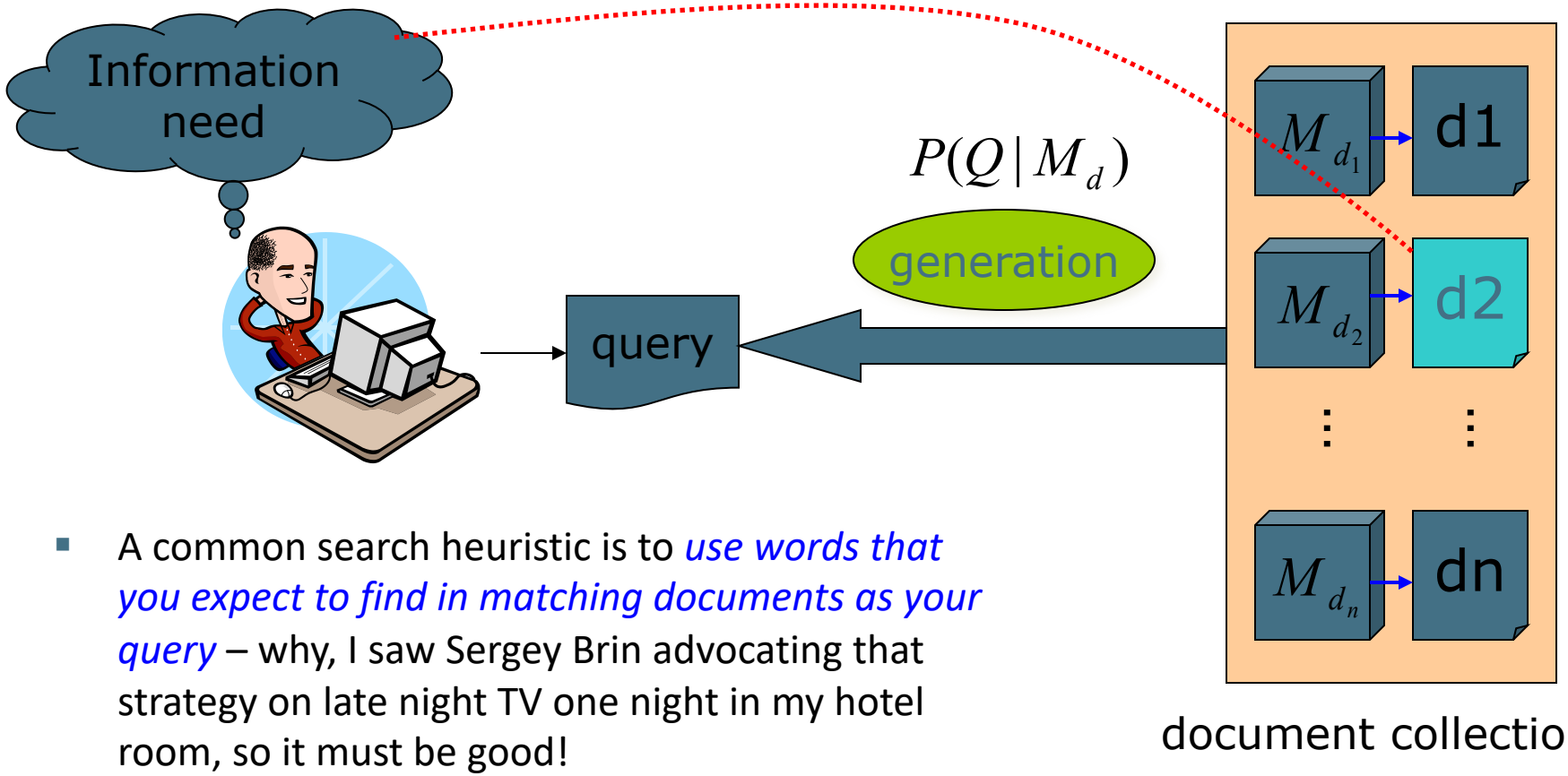
Today

- The Language Model Approach to IR
  - Basic query generation model
  - Alternative models

Standard Probabilistic IR



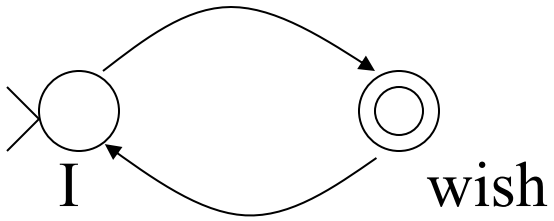
IR based on Language Model (LM)



- A common search heuristic is to *use words that you expect to find in matching documents as your query* – why, I saw Sergey Brin advocating that strategy on late night TV one night in my hotel room, so it must be good!
- The LM approach directly exploits that idea!
- See later slides for a more formal justification

Formal Language (Model)

- Traditional generative model: generates strings
  - Finite state machines or regular grammars, etc.
- Example:



I wish
I wish I wish
I wish I wish I wish
I wish I wish I wish I wish
…

Stochastic Language Models

- Models *probability* of generating strings in the language (commonly all strings over alphabet ∑)

Model M

0.2	the
0.1	a
0.01	man
0.01	woman
0.03	said
0.02	likes
…	…

s:	the	man	likes	the	woman
	0.2	0.01	0.02	0.2	0.01

multiply

$P(s \mid M) = 0.00000008$

Stochastic Language Models

- Model *probability* of generating any string

Model M1	
0.2	the
0.01	class
0.0001	sayst
0.0001	pleaseth
0.0001	yon
0.0005	maiden
0.01	woman

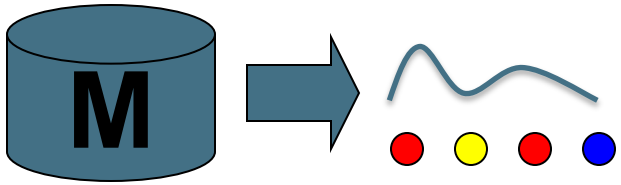
Model M2	
0.2	the
0.0001	class
0.03	sayst
0.02	pleaseth
0.1	yon
0.01	maiden
0.0001	woman

the	class	pleaseth	yon	maiden
0.2	0.01	0.0001	0.0001	0.0005
0.2	0.0001	0.02	0.1	0.01

$$P(s|M2) > P(s|M1)$$

Stochastic Language Models

- A statistical model for generating text
  - Probability distribution over strings in a given language



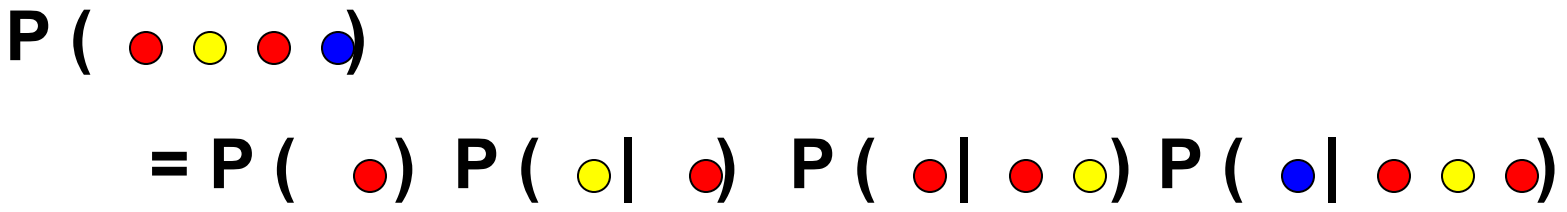
P(● ○ ● ● | M) = P(● | M)

P(○ | M, ●)

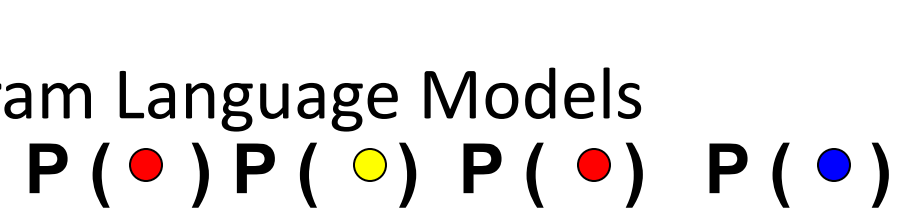
P(● | M, ● ○)

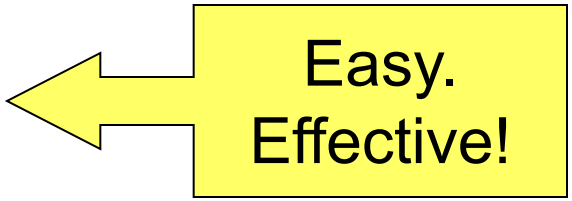
P(● | M, ● ○ ●)

Unigram and higher-order models

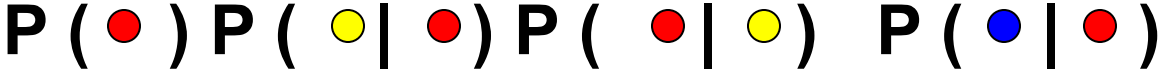


- Unigram Language Models





- Bigram (generally, *n*-gram) Language Models



- Other Language Models
  - Grammar-based models (PCFGs), etc.
    - Probably not the first thing to try in IR

Using Language Models in IR

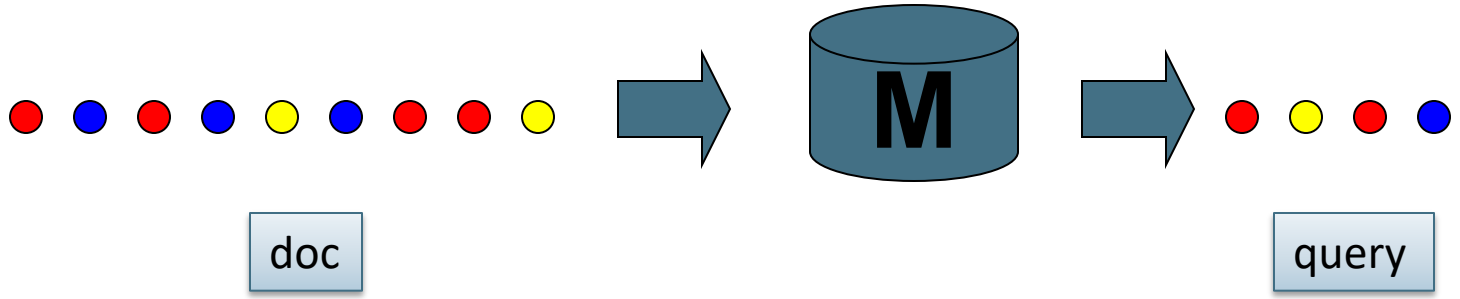
- Treat each document as the basis for a model (e.g., unigram sufficient statistics)
- Rank document d based on P(d | q)
- P(d | q) = P(q | d) x P(d) / P(q)
  - P(q) is the same for all documents, so ignore
  - P(d) [the prior] is often treated as the same for all d
    - But we could use criteria like authority, length, genre
  - P(q | d) is the probability of q given d's model
- Very general formal approach

The fundamental problem of LMs

- Usually we don't know the model **M**
  - But have a sample of text representative of that model

P (● ● ● ● | M (● ● ● ● ● ● ● ● ●))

- Estimate a language model from a sample
- Then compute the observation probability



Language Models for IR

- Language Modeling Approaches
  - Attempt to **model query generation process**
  - Documents are ranked by **the probability that a query would be observed as a random sample from the respective document model**
    - Multinomial approach

$$P(Q|M_D) = \prod_w P(w|M_D)^{q_w}$$

Retrieval based on probabilistic LM

- Treat the generation of queries as a random process.
- Approach
  - Infer a language model for each document.
  - Estimate the probability of generating the query according to each of these models.
  - Rank the documents according to these probabilities.
  - Usually a unigram estimate of words is used
    - Some work on bigrams, paralleling van Rijsbergen

Retrieval based on probabilistic LM

- Intuition
  - Users …
    - Have a reasonable idea of terms that are likely to occur in documents of interest.
    - They will choose query terms that distinguish these documents from others in the collection.
- Collection statistics …
  - Are integral parts of the language model.
  - Are not used heuristically as in many other approaches.
    - In theory. In practice, there's usually some wiggle room for empirically set parameters

Query generation probability (1)

- Ranking formula

$$p(Q,d) = p(d)\,p(Q \mid d)$$

$$\approx p(d)\,p(Q \mid M_d)$$

- The probability of producing the query given the language model of document d using MLE is:

$$\hat{p}(Q \mid M_d) = \prod_{t \in Q} \hat{p}_{ml}(t \mid M_d)$$

$$= \prod_{t \in Q} \frac{tf_{(t,d)}}{dl_d}$$

Unigram assumption:
Given a particular language model, the query terms occur independently

M_d : language model of document d

$tf_{(t,d)}$: raw tf of term t in document d

dl_d : total number of tokens in document d

Insufficient data

- Zero probability $p(t \mid M_d) = 0$
  - May not wish to assign a probability of zero to a document that is missing one or more of the query terms [gives conjunction semantics]
- LM-based smoothing approach
  - A non-occurring term is possible, but no more likely than would be expected by chance in the collection.
  - Naïve Idea: if $tf_{(t,d)} = 0$ then $p(t \mid M_d) \stackrel{?}{=} \frac{cf_t}{cs}$
    - Need to work on the maths so that $\sum_{t \in V} p(t \mid M_d) = 1$

cf_t : raw count of term t in the collection

cs : raw collection size(total number of tokens in the collection)

Insufficient data

- Zero probabilities spell disaster
  - We need to smooth probabilities
    - Discount nonzero probabilities
    - Give some probability mass to unseen things
- There's a wide space of approaches to smoothing probability distributions to deal with this problem, such as adding 1, ½ or ε to counts, Dirichlet priors, discounting, and interpolation
  - [See FSNLP ch. 6 or CS224N if you want more]
- A simple idea that works well in practice is to use a mixture between the document multinomial and the collection multinomial distribution

Mixture model

- Jelinek-Mercer method
  - $P(w|d) = \lambda P_{mle}(w|M_d) + (1 - \lambda)P_{mle}(w|M_c)$
- Mixes the probability from the document with the general collection frequency of the word.
- Correctly setting λ is very important
- A high value of lambda makes the search "conjunctive-like" – suitable for short queries
- A low value is more suitable for long queries
- Can tune λ to optimize performance
  - Perhaps make it dependent on document size (cf. Dirichlet prior or Witten-Bell smoothing)

Basic mixture model summary

- General formulation of the LM for IR

$$p(Q,d) = p(d)\prod_{t\in Q}((1-\lambda)p(t)+\lambda p(t \mid M_d))$$

collection/background language model

individual-document model

- The user has a document in mind, and generates the query from this document.
- The equation represents the probability that the document that the user had in mind was in fact this one.

n: # terms in Q

Note here (i.e., [CMS09]) λ is multiplied to the background model.

Relationship to idf

$$\log P(Q|D) = \sum_{i=1}^{n} \log((1-\lambda)\frac{f_{q_i,D}}{|D|} + \lambda\frac{c_{q_i}}{|C|})$$

$f_{qi,D}=0$ ➔ query word that does not occur in the doc

$$= \sum_{i:f_{q_i,D}>0} \log((1-\lambda)\frac{f_{q_i,D}}{|D|} + \lambda\frac{c_{q_i}}{|C|}) + \sum_{i:f_{q_i,D}=0} \log(\lambda\frac{c_{q_i}}{|C|})$$

$$= \sum_{i:f_{q_i,D}>0} \log \frac{((1-\lambda)\frac{f_{q_i,D}}{|D|} + \lambda\frac{c_{q_i}}{|C|})}{\lambda\frac{c_{q_i}}{|C|}} + \sum_{i=1}^{n} \log(\lambda\frac{c_{q_i}}{|C|})$$

$$\overset{rank}{=} \sum_{i:f_{q_i,D}>0} \log\left(\frac{((1-\lambda)\frac{f_{q_i,D}}{|D|}}{\lambda\frac{c_{q_i}}{|C|}} + 1\right)$$

Add contributions from $i:f_{qi,D}>0$

Becomes a constant | Q, C

proportional to the tf, inversely proportional to the cf

Example

- Document collection (2 documents)
  - d_1: Xerox reports a profit but revenue is down
  - d_2: Lucent narrows quarter loss but revenue decreases further
- Model: MLE unigram from documents; $\lambda = ½$
- Query: *revenue down*
  - $P(Q|d_1) = [(1/8 + 2/16)/2] \times [(1/8 + 1/16)/2]$
    $= 1/8 \times 3/32 = 3/256$
  - $P(Q|d_2) = [(1/8 + 2/16)/2] \times [(0 + 1/16)/2]$
    $= 1/8 \times 1/32 = 1/256$
- Ranking: $d_1 > d_2$

Ponte and Croft Experiments

- Data
  - TREC topics 202-250 on TREC disks 2 and 3
    - Natural language queries consisting of one sentence each
  - TREC topics 51-100 on TREC disk 3 using the concept fields
    - Lists of good terms

<num>Number: 054
<dom>Domain: International Economics
<title>Topic: Satellite Launch Contracts
<desc>Description:
… </desc>

<con>Concept(s):
1. Contract, agreement
2. Launch vehicle, rocket, payload, satellite
3. Launch services, … </con>

Precision/recall results 202-250

	tf.idf	LM	%chg	I/D	Sign	Wilc.
Rel:	6501	6501				
Rret.:	3201	3364	+5.09	36/43	0.0000*	0.0002*
Prec.						
0.00	0.7439	0.7590	+2.0	10/22	0.7383	0.5709
0.10	0.4521	0.4910	+8.6	24/42	0.2204	0.0761
0.20	0.3514	0.4045	+15.1	27/44	0.0871	0.0081*
0.30	0.2761	0.3342	+21.0	28/43	0.0330*	0.0054*
0.40	0.2093	0.2572	+22.9	25/39	0.0541	0.0158*
0.50	0.1558	0.2061	+32.3	24/35	0.0205*	0.0018*
0.60	0.1024	0.1405	+37.1	22/27	0.0008*	0.0027*
0.70	0.0451	0.0760	+68.7	13/15	0.0037*	0.0062*
0.80	0.0160	0.0432	+169.6	9/10	0.0107*	0.0035*
0.90	0.0033	0.0063	+89.3	2/3	0.5000	undef
1.00	0.0028	0.0050	+76.9	2/3	0.5000	undef
Avg:	0.1868	0.2233	+19.55	32/49	0.0222*	0.0003*
Prec.						
5	0.4939	0.5020	+1.7	10/21	0.6682	0.4106
10	0.4449	0.4898	+10.1	22/30	0.0081*	0.0154*
15	0.3932	0.4435	+12.8	19/26	0.0145*	0.0038*
20	0.3643	0.4051	+11.2	22/34	0.0607	0.0218*
30	0.3313	0.3707	+11.9	28/41	0.0138*	0.0070*
100	0.2157	0.2500	+15.9	32/42	0.0005*	0.0003*
200	0.1655	0.1903	+15.0	35/44	0.0001*	0.0000*
500	0.1004	0.1119	+11.4	36/44	0.0000*	0.0000*
1000	0.0653	0.0687	+5.1	36/43	0.0000*	0.0002*
RPr	0.2473	0.2876	+16.32	34/43	0.0001*	0.0000*

Precision/recall results 51-100

	tf.idf	LM	%chg	I/D	Sign	Wilc.
Rel:	10485	10485				
Rret.:	5818	6105	+4.93	32/42	0.0005⋆	0.0003⋆
Prec.						
0.00	0.7274	0.7805	+7.3	10/22	0.7383	0.2961
0.10	0.4861	0.5002	+2.9	26/44	0.1456	0.1017
0.20	0.3898	0.4088	+4.9	24/45	0.3830	0.1405
0.30	0.3352	0.3626	+8.2	28/47	0.1215	0.0277⋆
0.40	0.2826	0.3064	+8.4	25/45	0.2757	0.0286⋆
0.50	0.2163	0.2512	+16.2	26/40	0.0403⋆	0.0007⋆
0.60	0.1561	0.1798	+15.2	20/30	0.0494⋆	0.0025⋆
0.70	0.0913	0.1109	+21.5	14/22	0.1431	0.0288⋆
0.80	0.0510	0.0529	+3.7	8/13	0.2905	0.2108
0.90	0.0179	0.0152	−14.9	1/4	0.3125	undef
1.00	0.0005	0.0004	−11.9	1/2	0.7500	undef
Avg:	0.2286	0.2486	+8.74	32/50	0.0325⋆	0.0015⋆
Prec.						
5	0.5320	0.5960	+12.0	15/21	0.0392⋆	0.0125⋆
10	0.5080	0.5260	+3.5	14/30	0.7077	0.1938
15	0.4933	0.5053	+2.4	14/28	0.5747	0.3002
20	0.4670	0.4890	+4.7	16/34	0.6962	0.1260
30	0.4293	0.4593	+7.0	20/32	0.1077	0.0095⋆
100	0.3344	0.3562	+6.5	29/45	0.0362⋆	0.0076⋆
200	0.2670	0.2852	+6.8	29/44	0.0244⋆	0.0009⋆
500	0.1797	0.1881	+4.7	30/42	0.0040⋆	0.0011⋆
1000	0.1164	0.1221	+4.9	32/42	0.0005⋆	0.0003⋆
RPr	0.2836	0.3013	+6.24	30/43	0.0069⋆	0.0052⋆

Language models: pro & con

- Novel way of looking at the problem of text retrieval based on probabilistic language modeling
  - Conceptually simple and explanatory
  - Formal mathematical model
  - Natural use of collection statistics, not heuristics (almost…)
- LMs provide effective retrieval and can be improved to the extent that the following conditions can be met
  - Our language models are accurate representations of the data.
  - Users have some sense of term distribution.*
    - *Or we get more sophisticated with translation model

Comparison With Vector Space

- There's some relation to traditional tf.idf models:
  - (unscaled) term frequency is directly in model
  - the probabilities do length normalization of term frequencies
  - the effect of doing a mixture with overall collection frequencies is a little like idf: terms rare in the general collection but common in some documents will have a greater influence on the ranking

Comparison With Vector Space

- Similar in some ways
  - Term weights based on frequency
  - Terms often used as if they were independent
  - Inverse document/collection frequency used
  - Some form of length normalization useful

- Different in others
  - Based on probability rather than similarity
    - Intuitions are probabilistic rather than geometric
  - Details of use of document length and term, document, and collection frequency differ

Resources

J.M. Ponte and W.B. Croft. 1998. A language modelling approach to information retrieval. In *SIGIR 21.*

D. Hiemstra. 1998. A linguistically motivated probabilistic model of information retrieval. *ECDL 2*, pp. 569–584.

A. Berger and J. Lafferty. 1999. Information retrieval as statistical translation. *SIGIR 22*, pp. 222–229.

D.R.H. Miller, T. Leek, and R.M. Schwartz. 1999. A hidden Markov model information retrieval system. *SIGIR 22*, pp. 214–221.

[Several relevant newer papers at *SIGIR 23–25,* 2000–2002.]

Workshop on Language Modeling and Information Retrieval, CMU 2001. http://la.lti.cs.cmu.edu/callan/Workshops/lmir01/ .

The Lemur Toolkit for Language Modeling and Information Retrieval. http://www-2.cs.cmu.edu/~lemur/ . CMU/Umass LM and IR system in C(++), currently actively developed.