Text Processing on the Web

Week 3

Probabilistic IR and Language Modeling

The material for these slides are borrowed heavily from the precursor of this course by Tat-Seng Chua as well as slides from the accompanying recommended texts Baldi et al. and Manning et al.

Recap

- What is Information Retrieval?
- The Vector Space Model
  - Representation of documents and queries as vectors
  - Calculate cosine of angle between vectors
  - Weighting dimensions
    - Term Frequency
    - Inverse Document (as opposed to Collection) Frequency
  - Relevance Feedback
  - Evaluation Metrics

Outline

- Probabilistic IR
- Homework #1 description
- Language Model-Based IR – (may not finish)

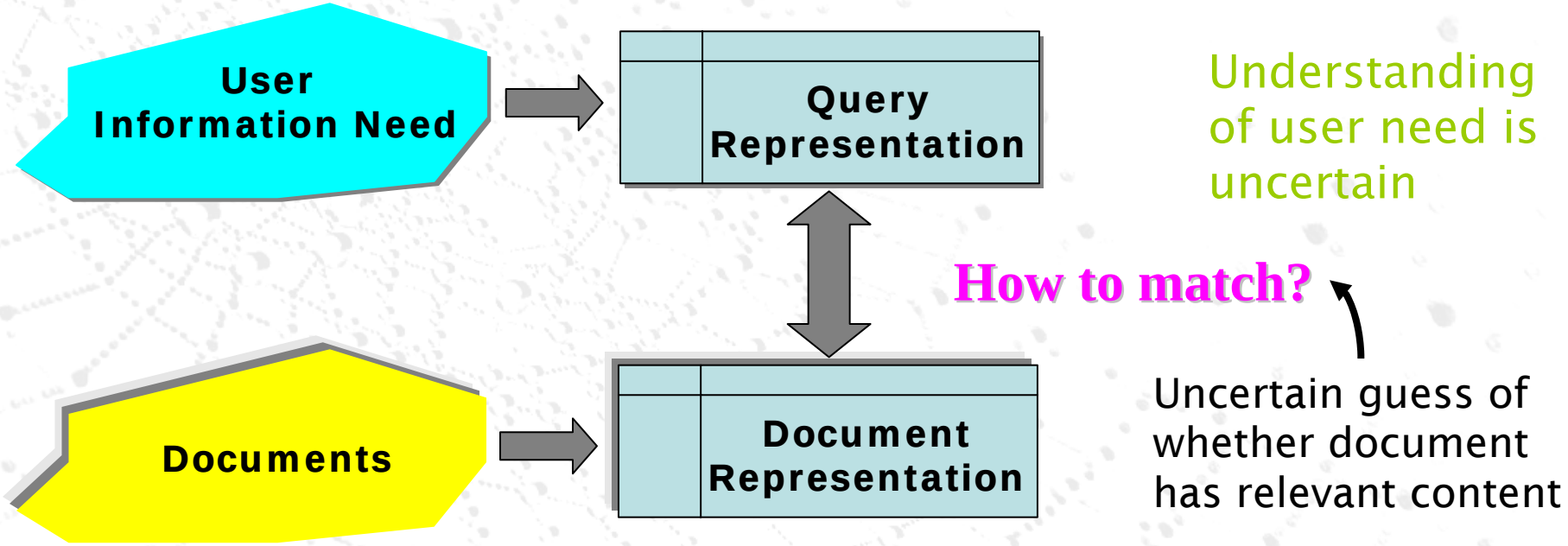
Extensions to Relevance Feedback not covered

Probabilistic relevance feedback

- Rather than reweighting in a vector space…
- If user has told us some relevant and some irrelevant documents, then we can proceed to build a probabilistic classifier:
  - $P(xt|R) = |\mathbf{D}_{rt}| / |\mathbf{D}_{r}|$
  - $P(xt|\underline{R}) = |\mathbf{D}_{\underline{r}t}| / |\mathbf{D}_{\underline{r}}|$

  x_t is a term; $\mathbf{D}_r$ is the set of known relevant documents; $\mathbf{D}_{rt}$ is the subset that contain x_t; $\mathbf{D}_{\underline{r}}$ is the set of known irrelevant documents; $\mathbf{D}_{\underline{r}t}$ is the subset that contain x_t.

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 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 - return to this later
- Bayesian networks for text retrieval
- Language model approach to IR - later today

- 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.

Document ranking

- 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 with respect to information need
  - P(relevant | $document_i$, query)

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)

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

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 $\underline{R}$ represent **non-relevance (NR).**

R={0,1} vs. $\underline{R}$/R

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

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

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

p(*R*),p($\underline{R}$) - prior probability of retrieving a (non) relevant document

$$p(R \mid x) + p(\underline{R} \mid x) = 1$$

p(*x*|*R*), p(*x*|$\underline{R}$) - 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 $p(R|x) > p(\underline{R}|x)$
- 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

Probability Ranking Principle

- How do we compute all those probabilities?
  - Do not know exact probabilities, have to use estimates
  - Binary Independence Retrieval (BIR): cover today
- Questionable assumptions, what are they?
  - "Relevance" of each document is independent of relevance of other documents.
    - In practice, it's bad to keep on returning duplicates
  - Boolean model of relevance
  - Users have just a single step information need
    - Don't think about the user's context in handling a query
    - E.g., Relevance Feedback

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 formula (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

"Binary": documents are represented as binary incidence vectors (instead of weighted by tf.idf)

- $x = (x_1, x_2, \ldots, x_n)$
- $x_t = 1$ iff term t is present in document x.

Note: Baldi et al. use ω_j for x_i

- **"Independence"**: terms occur in documents independently

- Some different documents may end up collapsed to the same vector. Why?

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
- Will use odds and Bayes' Rule:

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

Binary Independence Model

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

Constant for a given query, throw out

Needs estimation

Using **Independence** Assumption:

$$\frac{p(\vec{x} \mid R, q)}{p(\vec{x} \mid \underline{R}, q)} = \prod_{t=1}^{n} \frac{p(x_t \mid R, q)}{p(x_t \mid \underline{R}, q)}$$

So : $$O(R \mid q, d) = O(R \mid q) \cdot \prod_{t=1}^{n} \frac{p(x_t \mid R, q)}{p(x_t \mid \underline{R}, q)}$$

Binary Independence Model

- Copied over: $O(R|q,d) = O(R|q) \cdot \prod_{t=1}^{n} \frac{p(x_t|R,q)}{p(x_t|\underline{R},q)}$

- Since x_i is either 0 or 1:

$$O(R|q,d) = O(R|q) \cdot \prod_{x_t=1} \frac{p(x_t=1|R,q)}{p(x_t=1|\underline{R},q)} \cdot \prod_{x_t=0} \frac{p(x_t=0|R,q)}{p(x_t=0|\underline{R},q)}$$

- Let $p_t = p(x_t=1|R,q);$ $u_t = p(x_t=1|\underline{R},q);$

Prob of x_t in relevant doc

Prob of x_t in non-relevant doc

- Assume, for all terms not in the query ($q_t=0$), $p_t = u_t$

  But this is not always true (e.g., in relevance feedback)

Binary Independence Model

$$O(R \mid q, \vec{x}) = O(R \mid q) \cdot \prod_{x_t = q_t = 1} \frac{p_t}{u_t} \cdot \prod_{\substack{x_t = 0 \\ q_t = 1}} \frac{1 - p_t}{1 - u_t}$$

All matching terms

Non-matching query terms

$$= O(R \mid q) \cdot \prod_{x_t = q_t = 1} \frac{p_t (1 - u_t)}{u_t (1 - p_t)} \cdot \prod_{q_t = 1} \frac{1 - p_t}{1 - u_t}$$

All matching terms

All query terms

Binary Independence Model

$$O(R \mid q, \vec{x}) = O(R \mid q) \cdot \prod_{x_t = q_t = 1} \frac{p_t(1-u_t)}{u_t(1-p_t)} \cdot \prod_{q_t = 1} \frac{1-p_t}{1-u_t}$$

Constant

Constant

- Retrieval Status Value:

$$RSV = \log \prod_{x_t = q_t = 1} \frac{p_t(1-u_t)}{u_t(1-p_t)} = \sum_{x_t = q_t = 1} \log \frac{p_t(1-u_t)}{u_t(1-p_t)} = \sum c_t$$

Binary Independence Model

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

Documents	Relevant	Non-Relevant
$X_t=1$		
$X_t=0$		

- Estimate: $p_t \approx \frac{s}{S} \quad u_t \approx \frac{(n-s)}{(N-S)}$

$$c_t \approx \log \frac{(s\ \ )/(S-s\ \ )}{(df_t-s\ \ )/(N-df_t-S+s\ \ )}$$

For smoothing

Estimation – key challenge

- If non-relevant documents are approximated by the whole collection, then u_i (prob. of occurrence in non-relevant documents for query) is df_t/N and
  - $\log (1- u_t)/u_t = \log (N- df_t)/df_t \approx \log N/df_t$ = IDF!
- p_t (probability of occurrence in relevant documents) can be estimated in various ways:
  - From relevant documents if we know some
    - Relevance weighting can be used in feedback loop
  - Constant

Summary: PRP and BIR

- Getting reasonable approximations of probabilities is possible

Fix these in Okapi BM 25, next

- Requires restrictive assumptions:
  - term independence
  - terms not in query don't affect the outcome
  - boolean representation of documents/queries/relevance
  - does not account for different document lengths
  - document relevance values are independent

Okapi BM 25

$$RSV_d = \sum_{t \in q} \log \frac{N}{df_t}$$

$$= \sum_{t \in q} \log \left[\frac{N}{df_t} \right] \cdot \frac{(k_1 + 1) tf_{td}}{k_1((1-b) + b \times (L_d / L_{ave})) + tf_{td}}$$

Notes:

- k_1 = tuning parameter for term frequency (when k_1=0 then you get BIM)
- b = document length parameter (when b=0 no normalization for length)

Quick question: why does k_1 appear outside of b?

Food for thought

- Think through the differences between standard tf.idf and the probabilistic retrieval model in the first iteration

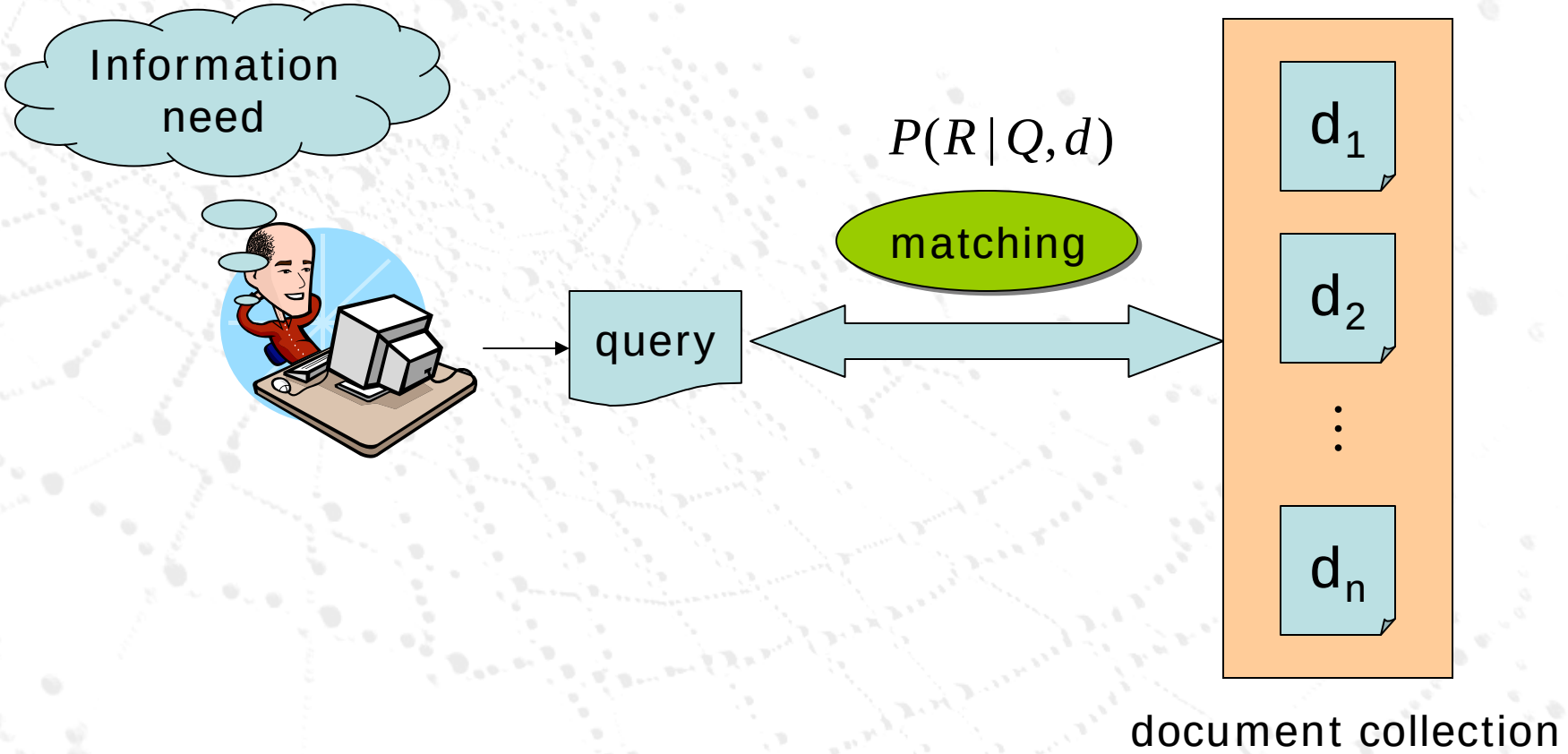
- The BM 25 still seems a bit heuristic in needing those pesky tuning parameters. Can you think of ways to get (around) them?
  - Hint: Check the 11.4.3 section in the Manning et al. text.

Resources

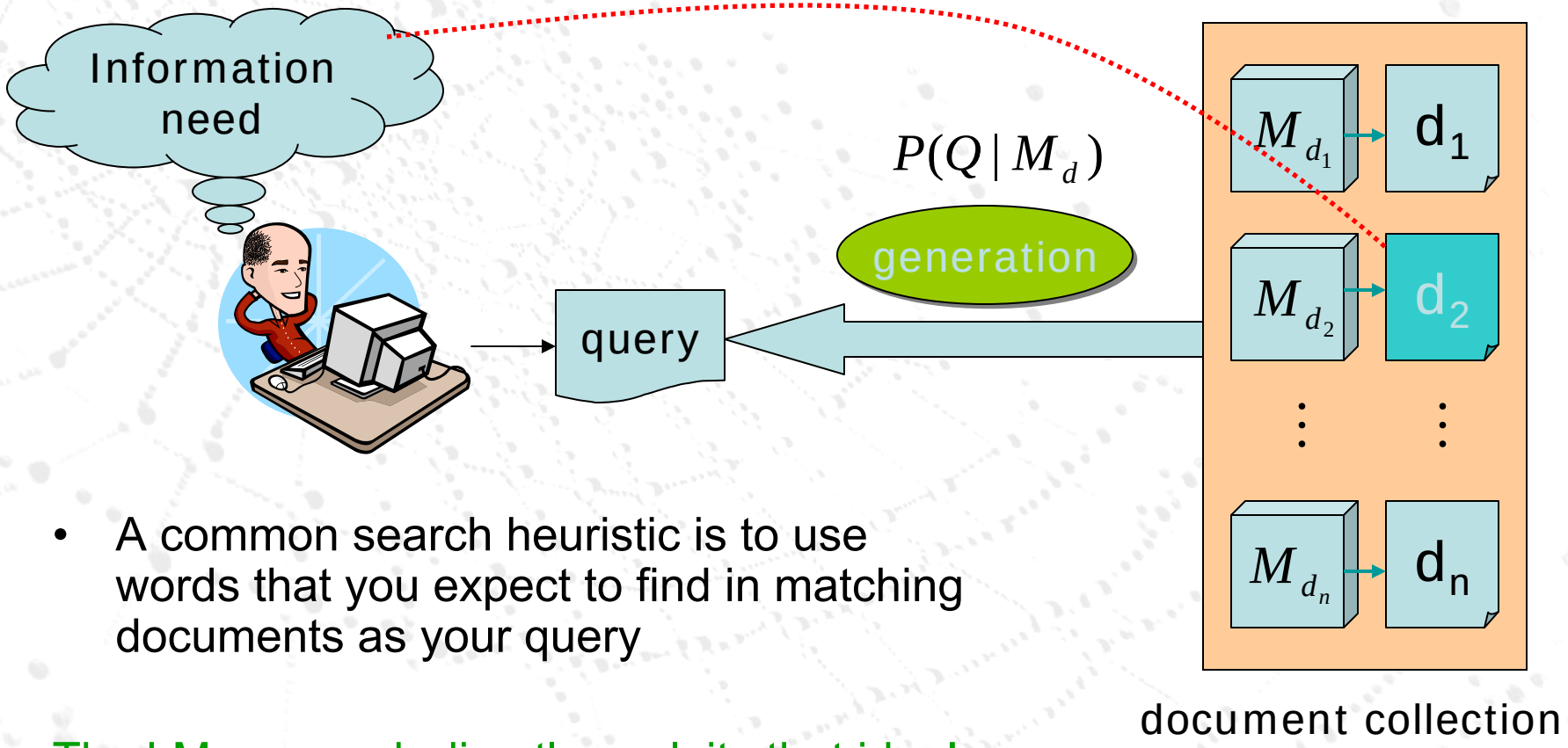
- S. E. Robertson and K. Spärck Jones. 1976. Relevance Weighting of Search Terms. Journal of the American Society for Information Sciences 27(3): 129–146.
- C. J. van Rijsbergen. 1979. Information Retrieval. 2nd ed. London: Butterworths, chapter 6. [Most details of math] http://www.dcs.gla.ac.uk/Keith/Preface.html
- 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/
- R. K. Belew. 2001. Finding Out About: A Cognitive Perspective on Search Engine Technology and the WWW. Cambridge UP 2001.
- Modern Information Retrieval, 2.5.4, 2.8

Language Models for IR

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

The LM approach directly exploits that idea!

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

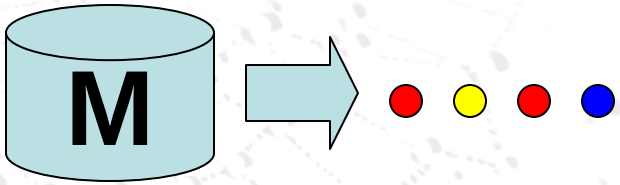
Model M2	
0.2	the
0.0001	class
0.03	sayst
0.02	pleaseth
0.1	yon
0.01	maiden
0.0001	woman

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

P (● ○ ● ●)

= P (●) P (○ | ●) P (● | ● ○) P (● | ● ○ ●)

- Unigram Language Models

  P (●) P (○) P (●) P (●)

  Easy. Effective!

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

  P (●) P (○ | ●) P (● | ○) P (● | ●)

- Other Language Models
  - Probably too complex for current IR

Using Language Models in IR

1. Treat each document as the basis for a model (e.g., unigram statistics)
2. Rank document d based on P(d | q)

- P(d | q) = P(q | d) 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**

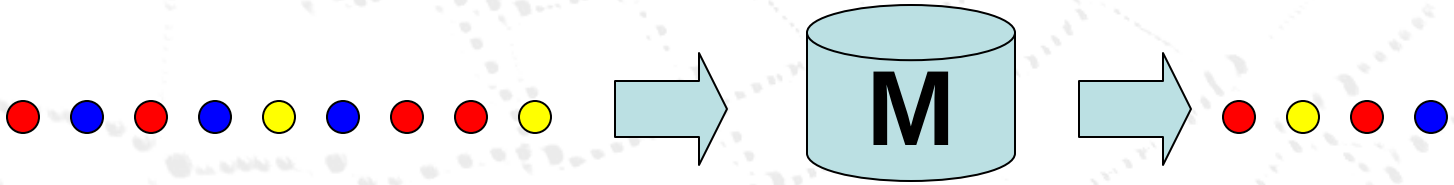
- Very general formal approach

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
  - Model the query generation process
  - Rank documents by the prob that a query would be observed as a random sample from the respective document model

    - Multinomial approach

$$P(q \mid M_d) = \prod_w P(w \mid M_d)^{q_w}$$

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

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.
      - But in practice, there's usually some empirically set parameters

Query generation probability

- 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 term frequency of term *t* in document *d*

dl_d : total number of tokens in document *d* (document length)

Insufficient data

- Zero probabilities spell disaster $p(t \mid M_d) = 0$
  - We need to smooth probabilities
    - Otherwise, gives conjunction semantics
    - Give some probability mass to unseen things

- Many approaches to smoothing probability distributions: adding 1, ½ or ε to counts, Dirichlet priors, discounting, and interpolation
- A simple idea: use a mixture between the document multinomial and the collection multinomial distribution

Mixture model

$$P(w|d) = \lambda P_{mle}(w|M_d) + (1 - \lambda)P_{mle}(w|M_c)$$

- 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

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: $d_1 > d_2$

LM vs. Prob. Model

- The main difference is whether “relevance” figures explicitly in the model or not
  - LM approach does away with modeling relevance
- LM approach assumes that documents and queries are of the same type
- Computationally tractable, intuitively appealing
- Problems of basic LM approach
  - Assumed equivalence of document and information problem representation is unrealistic
  - Very simple models of language - can’t scale to large n-gram
  - No notion of relevance so relevance feedback is difficult to integrate

LM vs. VSM

- 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

Summary

- Last Week: VSM
  - Empirical for the most part; success measured by results
  - Few properties provable (!)

- Probabilistic Model
  - +: Based on a firm theoretical foundation; justified optimal ranking
  - -: Binary word-in-doc weights (not using term frequencies)
    Independence of terms (can be alleviated)
    Has never worked convincingly better in practice

- Language Model
  - Accounts for term frequency and document length within model
  - But based in probability so accounting is different
  - Like VSM, puts queries and documents as same types of objects

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.