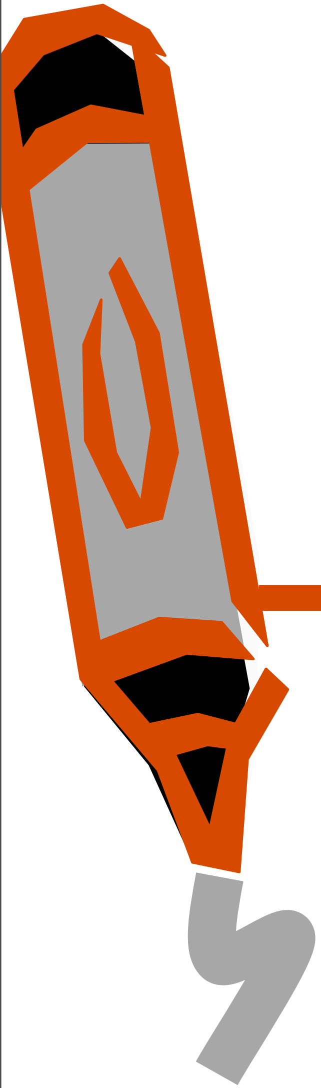
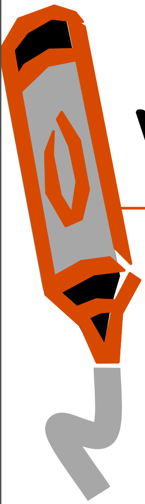


language models for retrieval

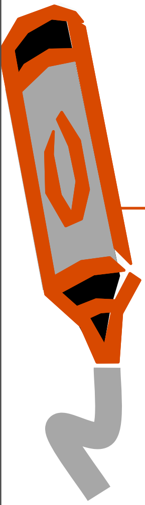
many slides courtesy James Allan@umass Amherst
some slides courtesy Christopher Manning and Prabhakar Raghavan @ Stanford



what is a retrieval model?

- Model is an idealization or abstraction of an actual process
- Mathematical models are used to study the properties of the process, draw conclusions, make predictions
- Conclusions derived from a model depend on whether the model is a good approximation of the actual situation
- Statistical models represent repetitive processes, make predictions about frequencies of interesting events
- Retrieval models can describe the computational process
  - e.g. how documents are ranked
  - Note that how documents or indexes are *stored* is implementation
- Retrieval models can attempt to describe the human process
  - e.g. the information need, interaction
  - Few do so meaningfully
- Retrieval models have an explicit or implicit definition of relevance

retrieval models

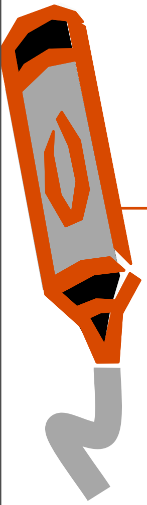


- boolean
- vector space
- latent semnatic indexing
- statistical language
- inference network
- hyperlink based

today

language models not in MIR

1. J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. Proceedings of ACM-SIGIR 1998, pages 275-281.
2. J. M. Ponte. A language modeling approach to information retrieval. Phd dissertation, University of Massachusets, Amherst, MA, September 1998.
3. D. Hiemstra. Using Language Models for Information Retrieval. PhD dissertation, University of Twente, Enschede, The Netherlands, January 2001.
4. D. R. H. Miller, T. Leek, and R. M. Schwartz. A hidden Markov model information retrieval system. Proceedings of ACM-SIGIR 1999, pages 214-221.
5. F. Song and W. B. Croft. A general language model for information retrieval. In Proceedings of Eighth International Conference on Information and Knowledge Management (CIKM 1999)
6. S. F. Chen and J. T. Goodman. An empirical study of smoothing techniques for language modeling. In Proceedings of the 34th Annual Meeting of the ACL, 1996.
7. C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. Proceedings of the ACM-SIGIR 2001, pages 334-342.
8. V. Lavrenko and W. B. Croft. Relevance-based language models. Proceedings of the ACM SIGIR 2001, pages 120-127.
9. V. Lavrenko and W. B. Croft, Relevance Models in Information Retrieval, in Language Modeling for Information Retrieval, W. Bruce Croft and John Lafferty, ed., Kluwer Academic Publishers, chapter 2.

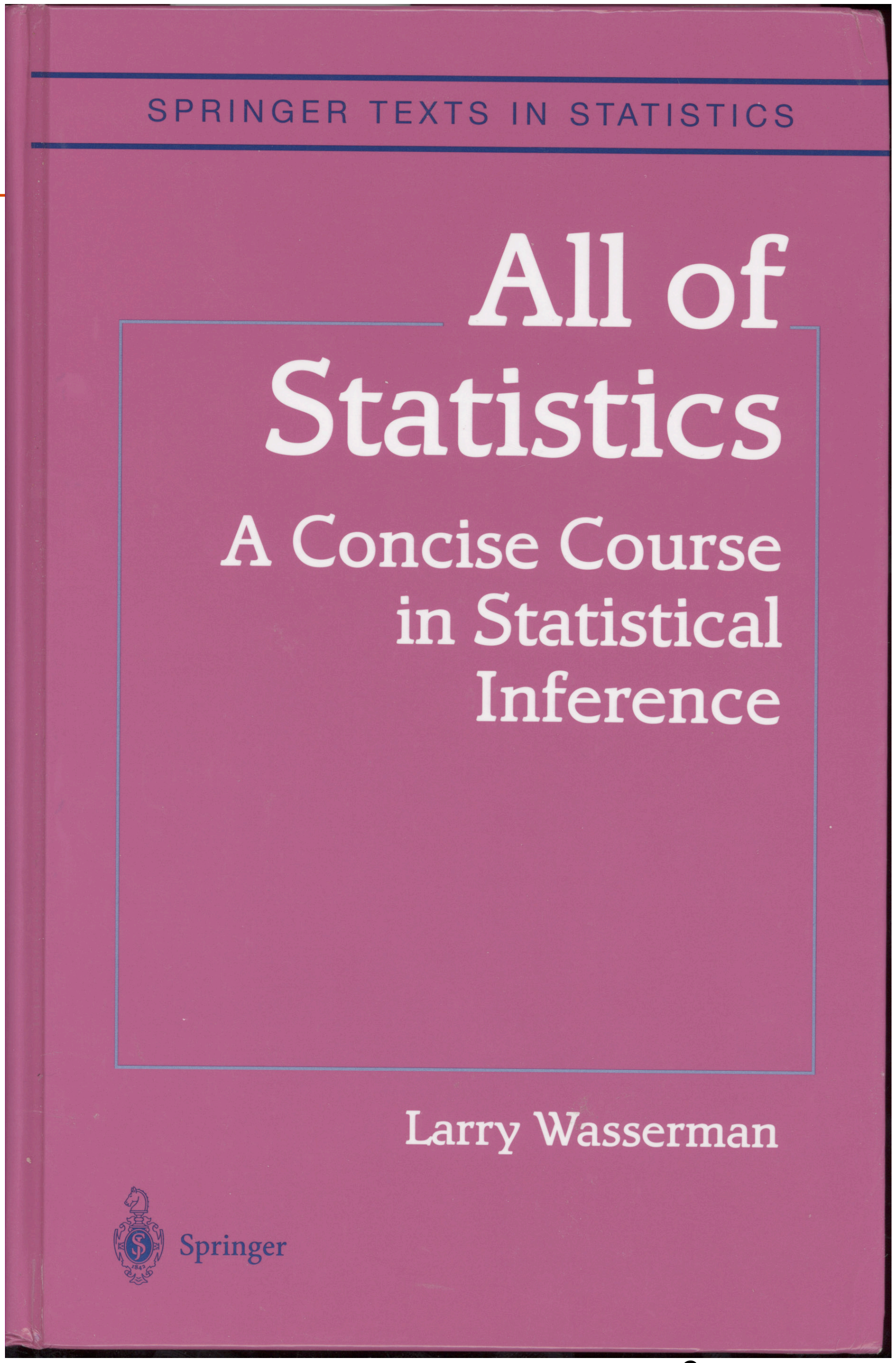


outline

- review: probabilities
- language model
- similarity, ranking in LM
- probability estimation
- smoothing methods
- examples

probabilities

- sample space
- probability
- independent events
- cond. probability
- Bayes theorem
- distributions



SPRINGER TEXTS IN STATISTICS

All of Statistics

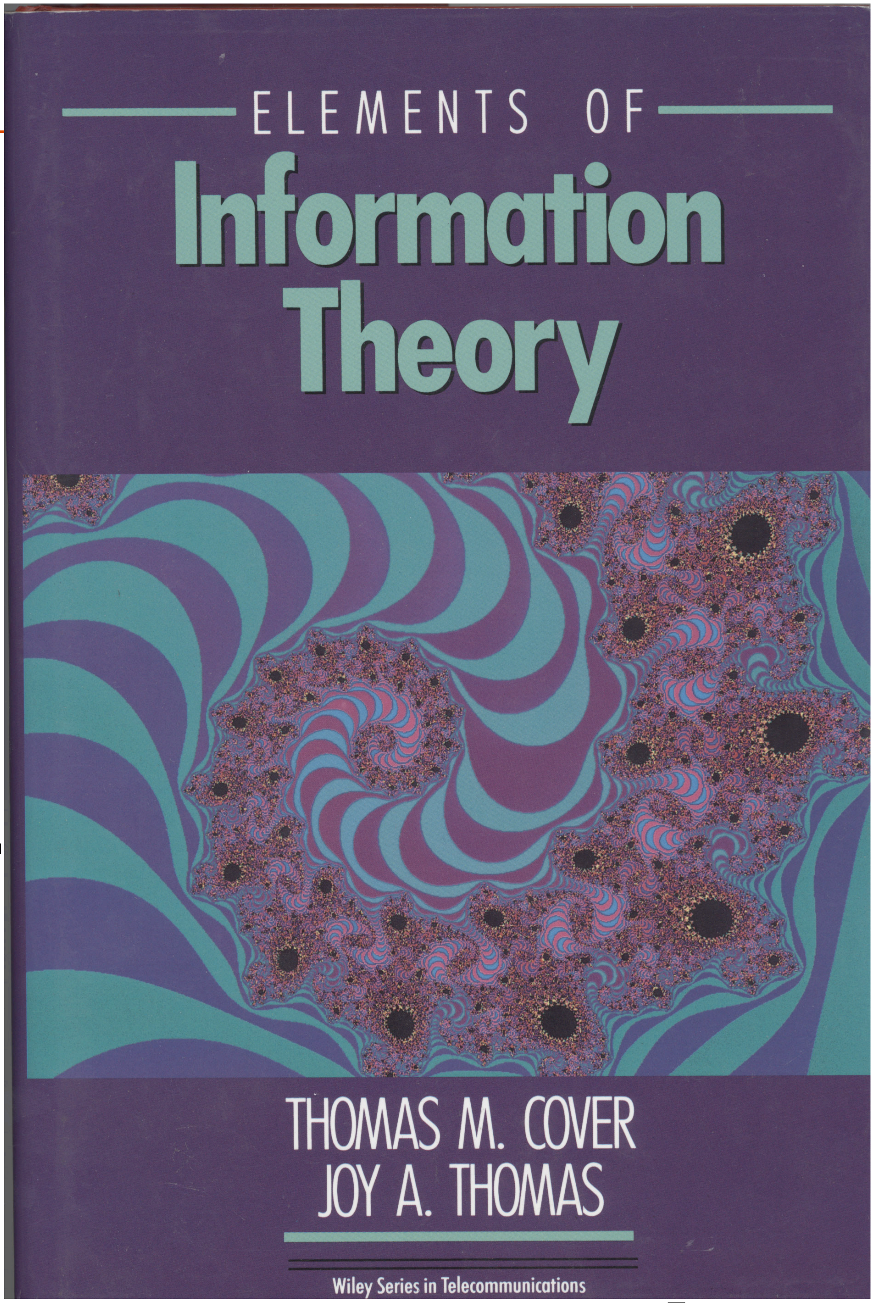
A Concise Course in Statistical Inference

Larry Wasserman

Springer

information theory, coding

- entropy
- joint entropy
- cond. entropy
- relative entropy
- convexity, Jensen ineq.
- optimal coding
- Fano’s ineq.



outline

- review: probabilities
- language model
- similarity, ranking in LM
- probability estimation
- smoothing methods
- examples

what is a language model ?

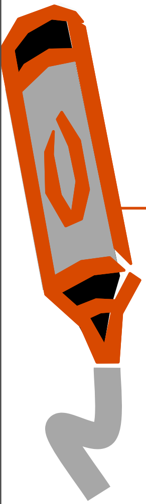
- Probability distribution over strings of text
  - how likely is a given string (observation) in a given "language"
  - for example, consider probability for the following four strings

p1 = P("a quick brown dog")
p2 = P("dog quick a brown")
p3 = P("быстрая brown dog")
p4 = P("быстрая собака")

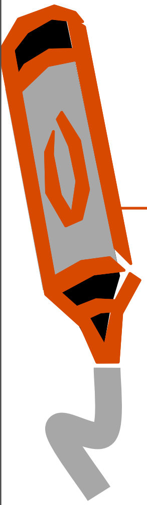
- English: p1 > p2 > p3 > p4

- … depends on what "language" we are modeling
  - In most of IR, assume that p1 == p2

lang modeling for IR



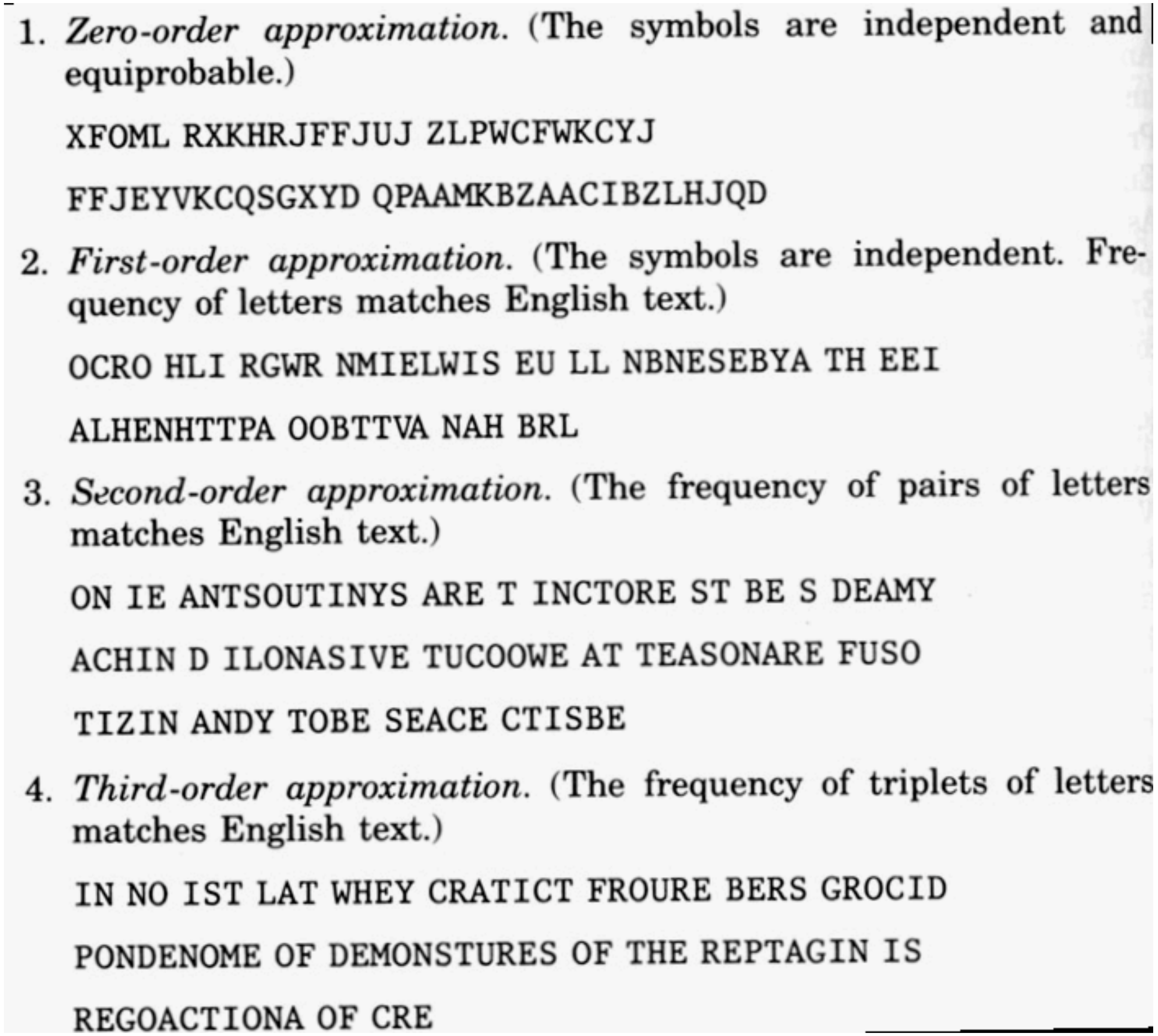
- Every document in a collection defines a “language”
  - consider all possible sentences (strings) that author could have written down when creating some given document
  - some are perhaps more likely to occur than others
- subject to topic, writing style, language …
  - $P(s|MD) =$ probability that author would write down string “s”
- think of writing a billion variations of a document and counting how many time we get “s”
- Now suppose “Q” is the user’s query
  - what is the probability that author would write down “q” ?
- Rank documents D in the collection by $P(Q|MD)$
  - probability of observing “Q” during random sampling from the language model of document D



language models

- estimate probabilities of certain ”events” in the text
- based on these probabilities, use likelihood as similarity
- language model based on
  - letters?
  - words?
  - phrases?

statistical text generation



1. *Zero-order approximation.* (The symbols are independent and equiprobable.)

   XFOML RXKHRJFFJUJ ZLPWCFWKCYJ

   FFJEYVKCQSGXYD QPAAMKBZAACIBZLHJQD

2. *First-order approximation.* (The symbols are independent. Frequency of letters matches English text.)

   OCRO HLI RGWR NMIELWIS EU LL NBNESEBYA TH EEI

   ALHENHTTPA OOBTTVA NAH BRL

3. *Second-order approximation.* (The frequency of pairs of letters matches English text.)

   ON IE ANTSOUTINYS ARE T INCTORE ST BE S DEAMY

   ACHIN D ILONASIVE TUCOOWE AT TEASONARE FUSO

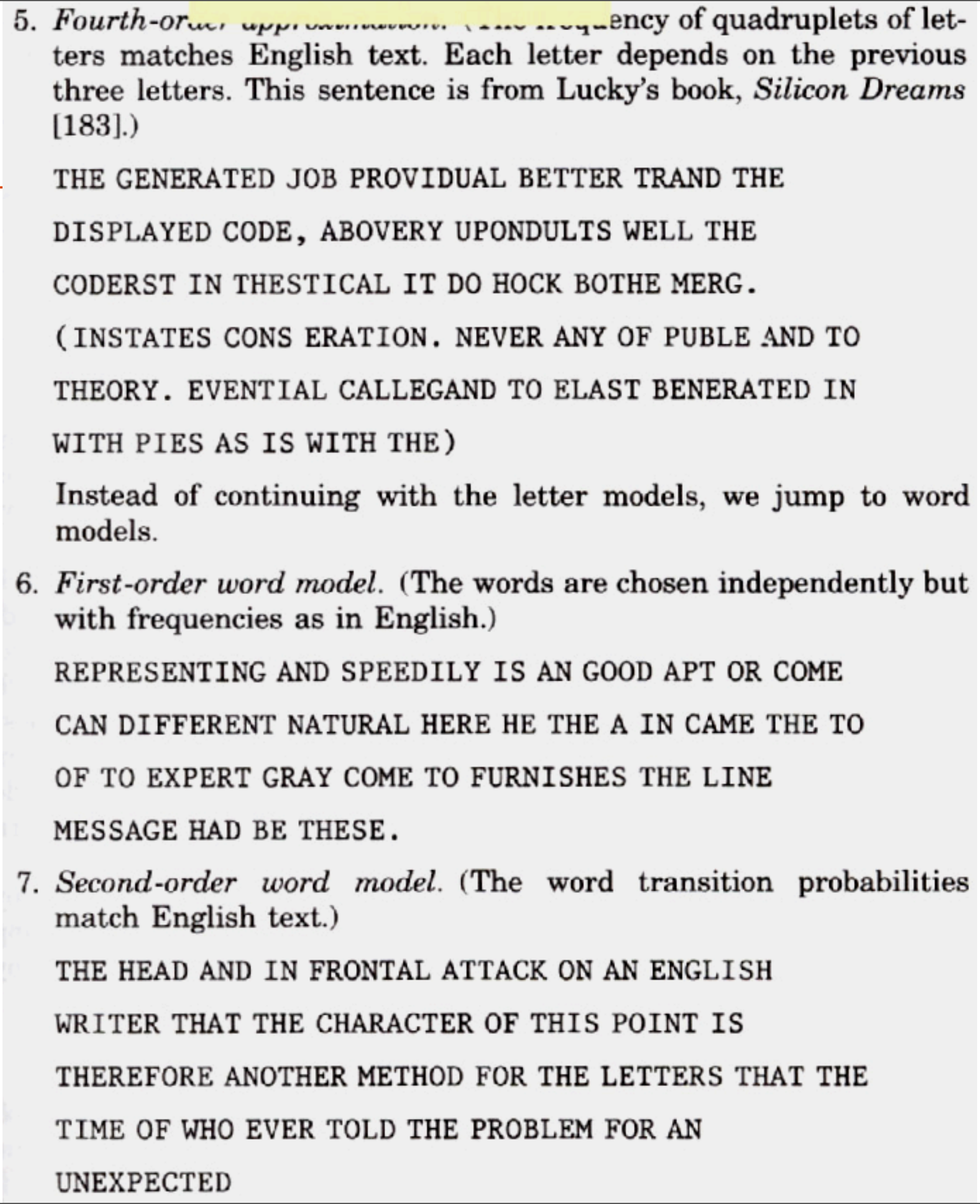
   TIZIN ANDY TOBE SEACE CTISBE

4. *Third-order approximation.* (The frequency of triplets of letters matches English text.)

   IN NO IST LAT WHEY CRATICT FROURE BERS GROCID

   PONDENOME OF DEMONSTURES OF THE REPTAGIN IS

   REGOACTIONA OF CRE



5. *Fourth-order approximation.* (The frequency of quadruplets of letters matches English text. Each letter depends on the previous three letters. This sentence is from Lucky's book, *Silicon Dreams* [183].)

```
THE GENERATED JOB PROVIDUAL BETTER TRAND THE
DISPLAYED CODE, ABOVERY UPONDULTS WELL THE
CODERST IN THESTICAL IT DO HOCK BOTHE MERG.
(INSTATES CONS ERATION. NEVER ANY OF PUBLE AND TO
THEORY. EVENTIAL CALLEGAND TO ELAST BENERATED IN
WITH PIES AS IS WITH THE)
```

Instead of continuing with the letter models, we jump to word models.

6. *First-order word model.* (The words are chosen independently but with frequencies as in English.)

```
REPRESENTING AND SPEEDILY IS AN GOOD APT OR COME
CAN DIFFERENT NATURAL HERE HE THE A IN CAME THE TO
OF TO EXPERT GRAY COME TO FURNISHES THE LINE
MESSAGE HAD BE THESE.
```

7. *Second-order word model.* (The word transition probabilities match English text.)

```
THE HEAD AND IN FRONTAL ATTACK ON AN ENGLISH
WRITER THAT THE CHARACTER OF THIS POINT IS
THEREFORE ANOTHER METHOD FOR THE LETTERS THAT THE
TIME OF WHO EVER TOLD THE PROBLEM FOR AN
UNEXPECTED
```

LM choices

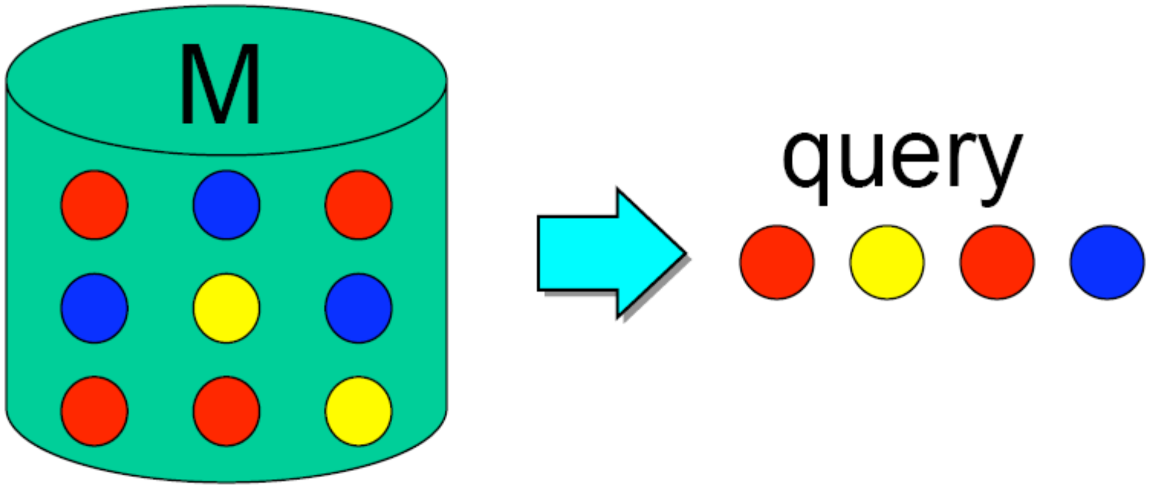
- What kind of language model should we use?
  - Unigram or higher-order models?
  - Multinomial or multiple-Bernoulli?

- How can we estimate model parameters?
  - Basic models
  - Translation models
  - Aspect models
  - non-parametric models

- How can we use the model for ranking?
  - Query-likelihood
  - Document-likelihood
  - Likelihood Ratio
  - Divergence of query and document models

unigram LM

- words are sampled independently, with replacement
- order of the words is lost (no phrases)

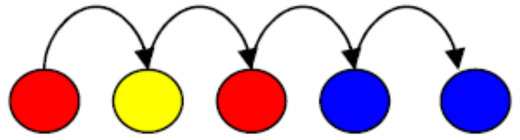


P (● ● ● ●) = P (●) P (●) P (●) P (●)

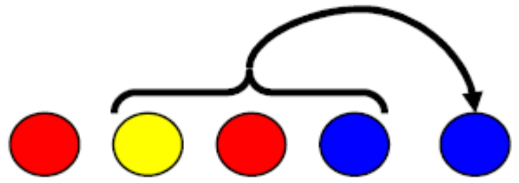
= 4 / 9 * 2 / 9 * 4 / 9 * 3 / 9

higher-order LM

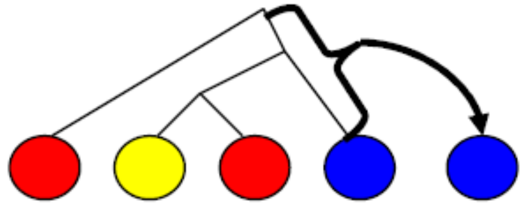
- Unigram model assumes word independence
  - cannot capture surface form: P(“brown dog”) == P(“dog brown”)
- Higher-order models
  - n-gram: condition on preceding words



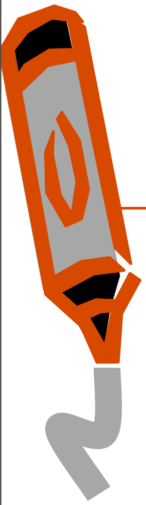
  - cache: condition on a window (cache)



  - grammar: condition on parse tree



- Are they useful?
  - no improvements from n-gram, grammar-based models
  - some research on cache-like models (proximity, passages, etc.)
  - parameter estimation is prohibitively expensive



outline

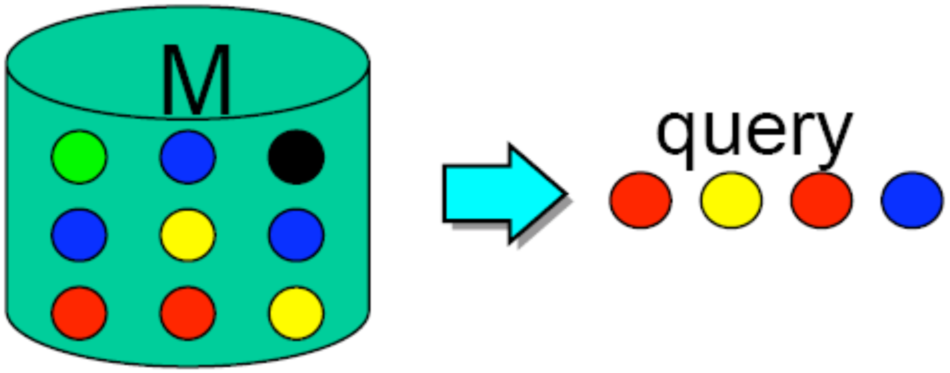
- review: probabilities
- language model
- similarity, ranking in LM
- probability estimation
- smoothing methods
- examples

multinomial similarity

- Predominant model

- Fundamental event:
what is the identity of the i'th query token?

- observation is a sequence of events, one for each query token



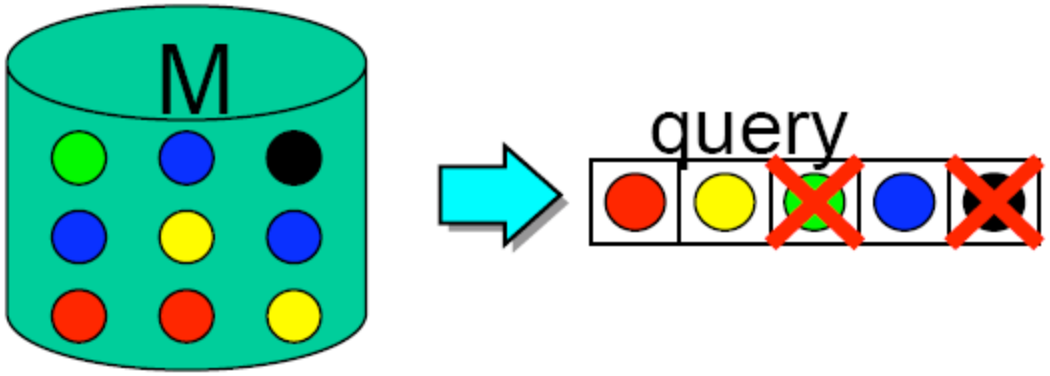
$$P(q_1 \ldots q_k \mid M) = \prod_{i=1}^{k} P(q_i \mid M)$$

multiple-Bernoulli similarity

- Original model

- fundamental event: *does the word w occur in the query?*

- observation is a vector of binary events, one for each possible word



$$P(q_1 \dots q_k \mid M) = \prod_{w \in q_1 \dots q_k} P(w \mid M) \prod_{w \notin q_1 \dots q_k} [1 - P(w \mid M)]$$

score, ranking in LM

- what is the probability to generate the given query, given a language model?
- what is the probability to generate the given document, given a language model?
- how "close" are 2 statistical models?

score: query likelihood

- Standard approach: query-likelihood
  - estimate a language model MD for every document D in the collection
  - rank docs by the probability of “generating” the query

$$P(q_1 \ldots q_k \mid M_D) = \prod_{i=1}^{k} P(q_i \mid M_D)$$

- |
  - no notion of relevance in the model: everything is random sampling
  - user feedback / query expansion not part of the model
  -examples of relevant documents cannot help us improve the language model MD
  - does not directly allow weighted or structured queries

score: document likelihood

- Flip the direction of the query-likelihood approach
  - estimate a language model MQ for the query Q
  - rank docs D by the likelihood of being a random sample from MQ
  - MQ expected to "predict" a typical relevant document

- Problems:
  - different doc lengths, probabilities not comparable
  - favors documents that contain frequent (low content) words

$$P(D \mid M_Q) = \prod_{w \in D} P(w \mid M_Q)$$

score: likelihood ratio

- Try to fix document likelihood:
  - Bayes' likelihood that M_q was the source, given that we observed D
  - related to Probability Ranking Principle: P(D|R) / P (D|N)
  - allows relevance feedback, query expansion, etc.
  - can benefit from complex estimation of the query model MQ

$$P(M_Q \mid D) = \frac{P(M_Q)P(D \mid M_Q)}{P(D)} \approx \frac{c\prod_{w \in D} P(w \mid M_Q)}{\prod_{w \in D} P(w \mid GE)}$$

score: model comparison

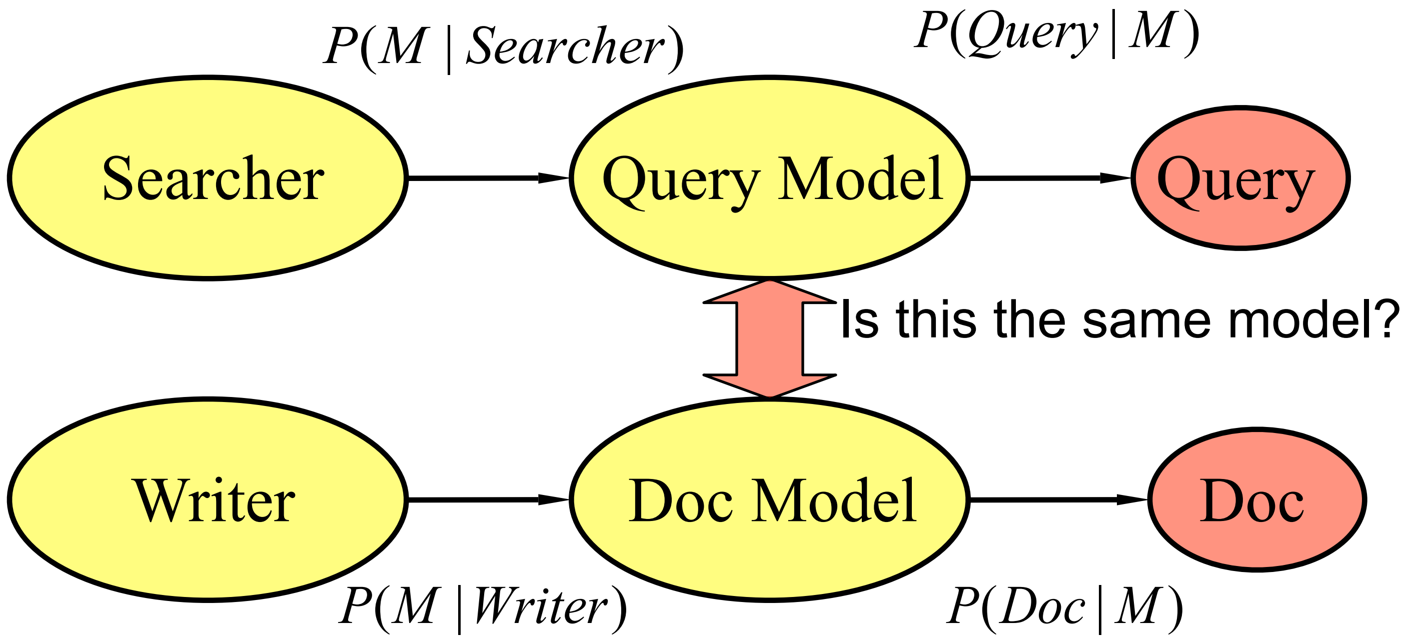
- Combine advantages of two ranking methods
  - estimate a model of both the query MQ and the document MD
  - directly compare similarity of the two models
  - natural measure of similarity is cross-entropy (others exist):

$$H(M_Q \,||\, M_D) = -\sum_w P(w \mid M_Q) \log P(w \mid M_D)$$

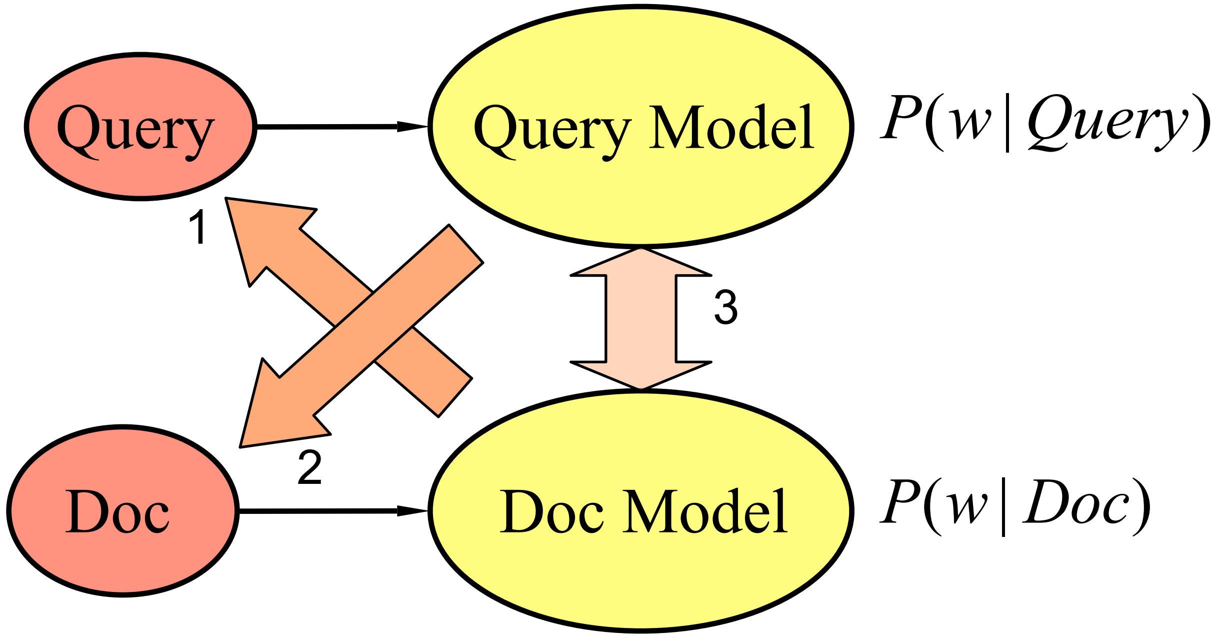
  - number of bits we would need to “encode” MQ using MD
  - equivalent to Kullback-Leibler divergence
  - equivalent to query-likelihood if MQ is simply counts of words in Q

- Cross-entropy is not symmetric: use H (MQ || MD)
  - reverse works consistently worse, favors different document
  - use reverse if ranking multiple queries w.r.t. one document

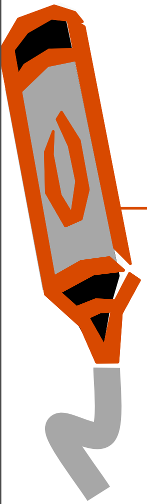
Models of Text Generation



Retrieval with Language Models



Retrieval:	Query likelihood (1)
	Document likelihood (2)
	Model comparison (3)



LM: popular choices

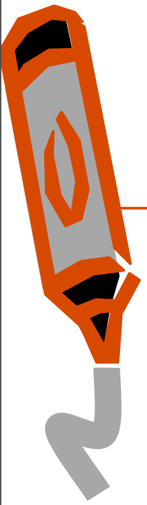
- Use Unigram models
  - no consistent benefit from using higher order models
  - estimation is much more complex (e.g. bi-gram from a 3-word query)
- Use Multinomial models
  - well-studied, consistent with other fields that use LMs
  - extend multiple-Bernoulli model to non-binary events?
- Use Model Comparison for ranking
  - allows feedback, expansion, etc. through estimation of MQ and MD
  - use KL(MQ || MD) for ranking multiple documents against a query
- Estimation of MQ and MD is a crucial step
  - very significant impact on performance (more than other choices)
  - key to cross-language, cross-media and other applications

outline

- review: probabilities
- language model
- similarity, ranking in LM
- probability estimation
- smoothing methods
- examples

estimation

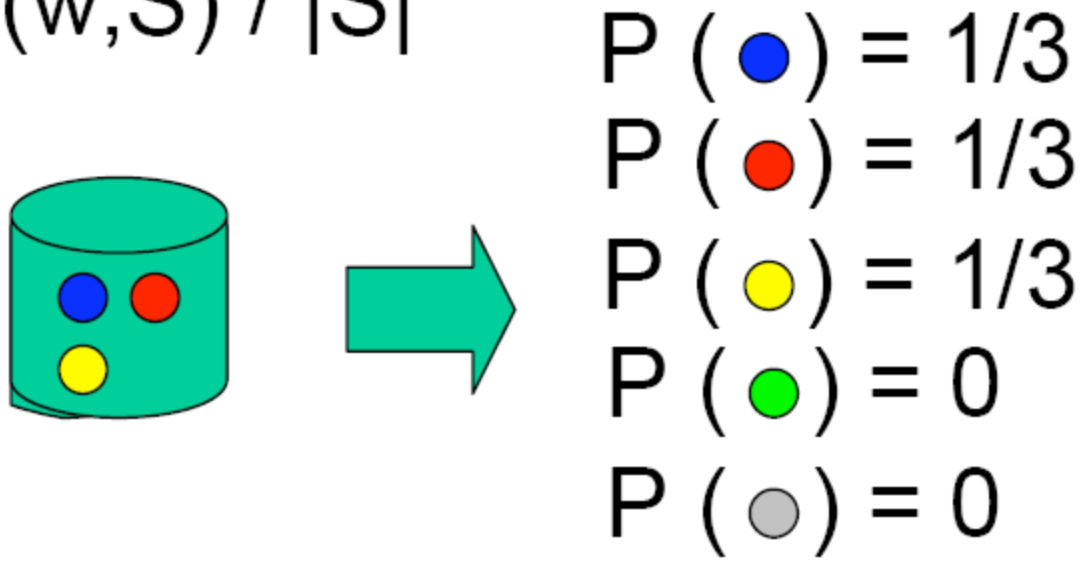
- Want to estimate MQ and/or MD from Q and/or D
- General problem:
  - given a string of text S (= Q or D), estimate its language model MS
  - S is commonly assumed to be an i.i.d. random sample from MS
    - Independent and identically distributed
- Basic Language Models
  - maximum-likelihood estimator and the zero frequency problem
  - discounting, interpolation techniques
  - Bayesian estimation



maximum likelihood

- count relative frequencies of words in S
- maximum-likelihood property:
  - assigns highest possible likelihood to the observation
- unbiased estimator:
  - if we repeat estimation an infinite number of times with different starting points S, we will get correct probabilities (on average)
  - this is not very useful…

$$P_{ml}(w|M_S) = \#(w,S) \;/\; |S|$$



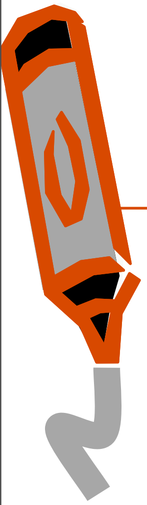
P (●) = 1/3

P (●) = 1/3

P (●) = 1/3

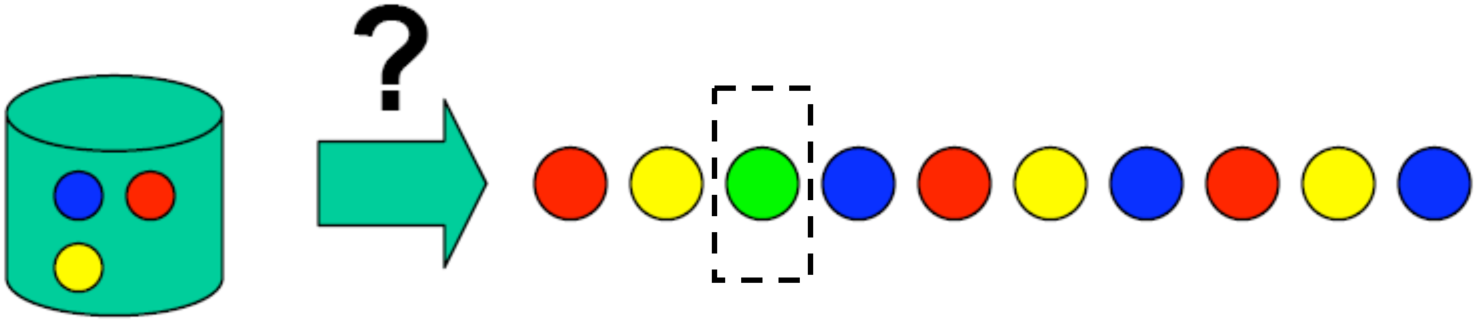
P (●) = 0

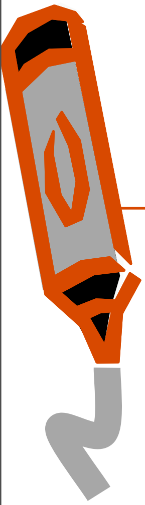
P (●) = 0



zero-frequency problem

- Suppose some event not in our observation S
  - Model will assign zero probability to that event
  - And to any set of events involving the unseen event
- Happens very frequently with language
- It is incorrect to infer zero probabilities
  - especially when creating a model from short samples





Laplace smoothing

- count events in observed data
- add 1 to every count
- renormalize to obtain probabilities
- it corresponds to uniform priors

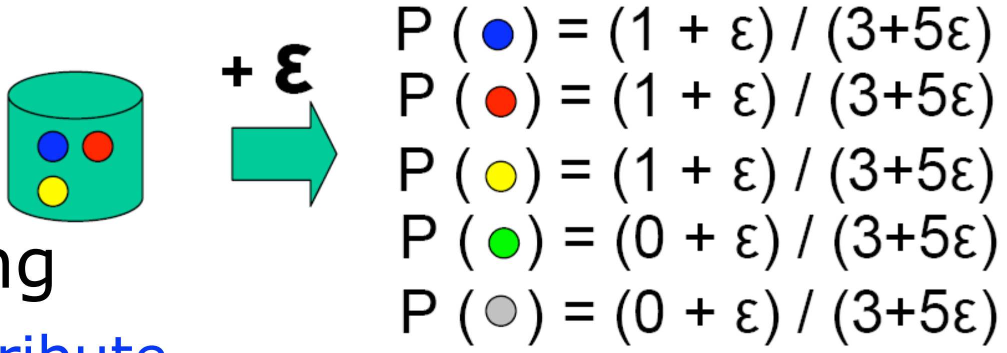
- if event counts are $(m_1, m_2, ..., m_k)$ with $\sum_i m_i = N$ then

max lielihood estimates are $(\frac{m_1}{N}, \frac{m_2}{N}, ..., \frac{m_k}{N})$

laplace estimates are $(\frac{m_1+1}{N+k}, \frac{m_2+1}{N+k}, ..., \frac{m_k+1}{N+k})$

discounting methods

- Laplace smoothing
- Lindstone correction
  - add ε to all count, renormalize
- absolute discounting
  - substract ε , redistribute probab mass



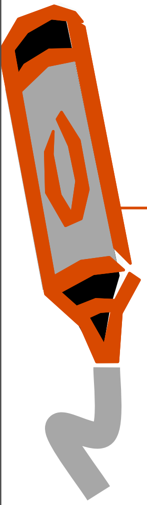
$P(●) = (1 + ε) / (3+5ε)$
$P(●) = (1 + ε) / (3+5ε)$
$P(●) = (1 + ε) / (3+5ε)$
$P(●) = (0 + ε) / (3+5ε)$
$P(○) = (0 + ε) / (3+5ε)$

discounting methods

- Held-out estimation
  - Divide data into training and held-out sections
  - In training data, count Nr, the number of words occurring r times
  - In held-out data, count Tr, the number of times those words occur
  - r* = Tr/Nr is adjusted count (equals r if training matches held-out)
  - Use r*/N as estimate for words that occur r times
- Deleted estimation (cross-validation)
  - Same idea, but break data into K sections
  - Use each in turn as held-out data, to calculate Tr(k) and Nr(k)
  - Estimate for words that occur r times is average of each
- Good-Turing estimation
  - From previous, P(w|M) = r* / N if word w occurs r times in sample
  - In Good-Turing, steal total probability mass from next most frequent word
  - Provides probability mass for words that occur r=0 times
  - Take what's leftover from r>0 to ensure adds to one

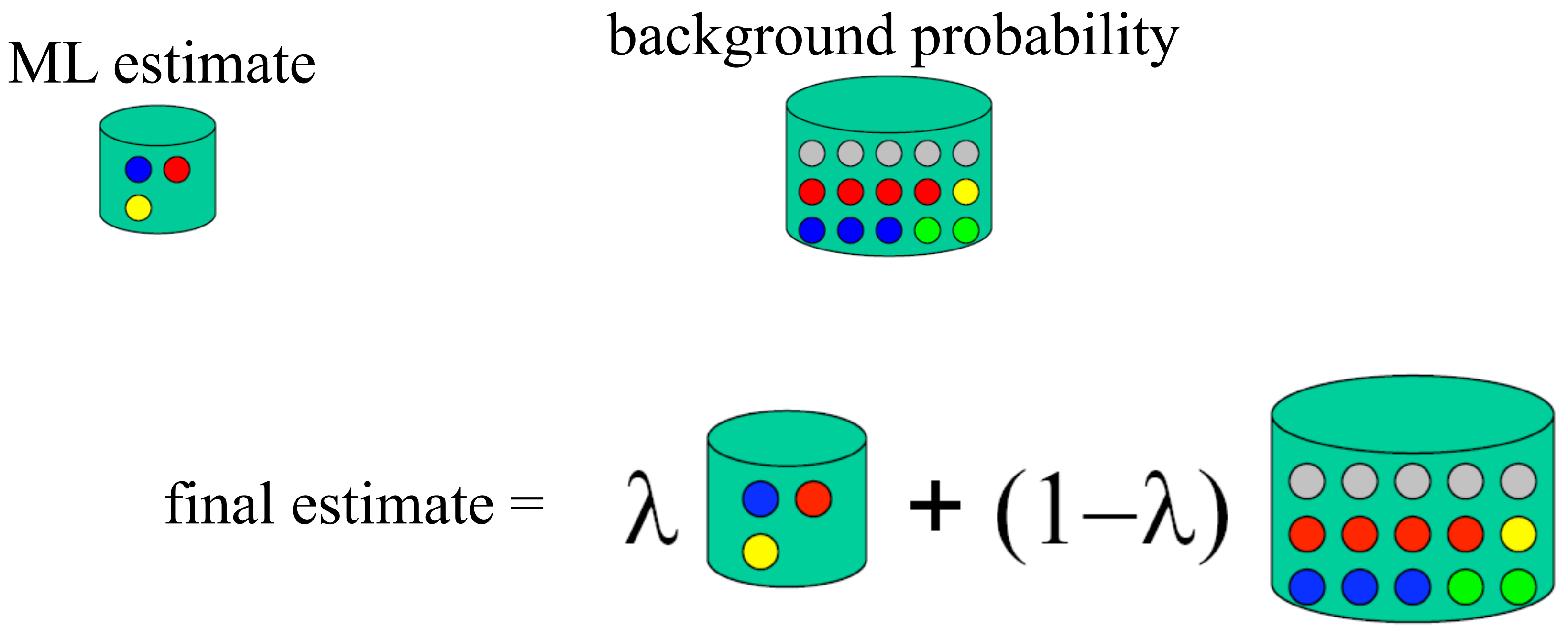
$$TPM(r+1) = N_{r+1} \cdot \frac{r+1}{N}$$

$$\begin{aligned} P(w_r|M) &= TPM(r+1)/N_r \\ &= \frac{N_{r+1}}{N_r} \cdot \frac{r+1}{N} \end{aligned}$$



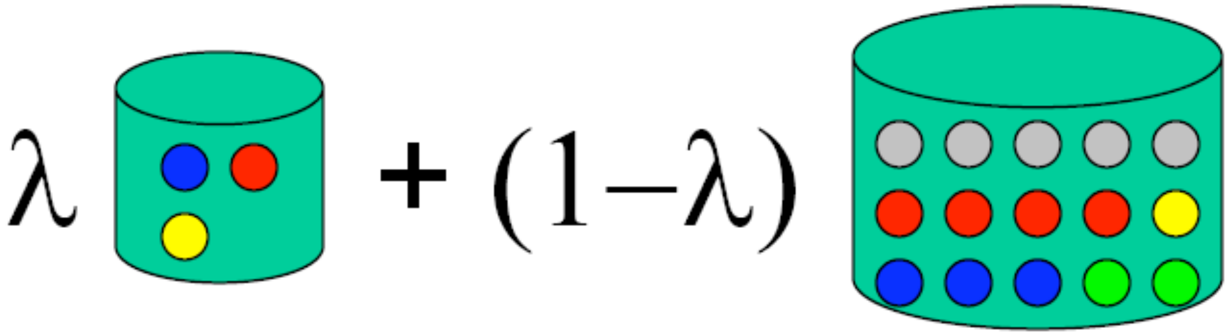
interpolation methods

- Problem with all discounting methods:
  - discounting treats unseen words equally (add or subtract ε)
  - some words are more frequent than others
- Idea: use background probabilities
  - "interpolate" ML estimates with General English expectations (computed as relative frequency of a word in a large collection)
  - reflects expected frequency of events



Jelinek Mercer smoothing

- Correctly setting **λ** is very important
- Start simple
  - set **λ** to be a constant, independent of document, query
- Tune to optimize retrieval performance
  - optimal value of **λ** varies with different databases, query sets, etc.



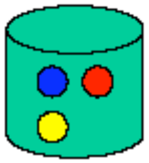
Dirichlet smoothing

- Problem with Jelinek-Mercer:
  - longer documents provide better estimates
  - could get by with less smoothing

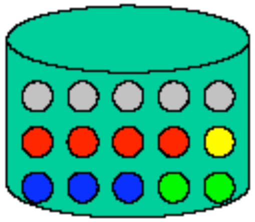
- Make smoothing depend on sample size

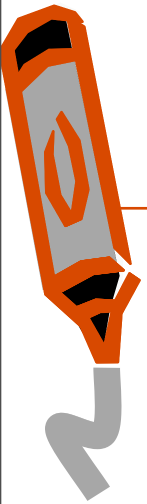
- N is length of sample = document length
- μ is a constant

$$\underbrace{N / (N + \mu)}_{\lambda}$$



$$+ \quad \underbrace{\mu / (N + \mu)}_{(1-\lambda)}$$

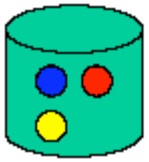
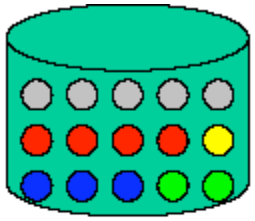




Witten-Bell smoothing

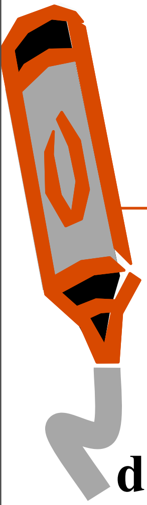
- A step further:
  - condition smoothing on “redundancy” of the example
  - long, redundant example requires little smoothing
  - short, sparse example requires a lot of smoothing

- Derived by considering the proportion of new events as we walk through example
  - N is total number of events = document length
  - V is number of unique events = number of unique terms in doc

$\underbrace{N / (N + V)}_{\lambda}$  $+$ $\underbrace{V / (N + V)}_{(1-\lambda)}$ 

interpolation vs back-off

- Two possible approaches to smoothing
- Interpolation:
  - Adjust probabilities for all events, both seen and unseen
- Back-off:
  - Adjust probabilities only for unseen events
  - Leave non-zero probabilities as they are
  - Rescale everything to sum to one: rescales “seen” probabilities by a constant
- Interpolation tends to work better
  - And has a cleaner probabilistic interpretation



Two-stage smoothing

Query = “the		algorithms	for	data	mining”
d1:	0.04	0.001	0.02	0.002	0.003
d2:	0.02	0.001	0.01	0.003	0.004

Two-stage smoothing

Query =	"the	algorithms	for	data	mining"
d1:	0.04	0.001	0.02	0.002	0.003
d2:	0.02	0.001	0.01	0.003	0.004

p("algorithms"|d1) = p("algorithm"|d2)
p("data"|d1) < p("data"|d2)
p("mining"|d1) < p("mining"|d2)

But p(q|d1)>p(q|d2)!

Two-stage smoothing

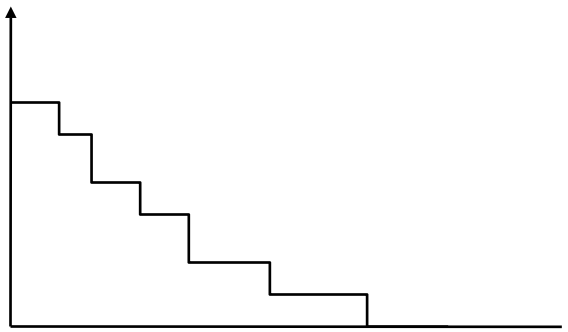
Query =	"the	algorithms	for	data	mining"
d1:	0.04	0.001	0.02	0.002	0.003
d2:	0.02	0.001	0.01	0.003	0.004

p("algorithms"|d1) = p("algorithm"|d2)
p("data"|d1) < p("data"|d2)
p("mining"|d1) < p("mining"|d2)

But p(q|d1)>p(q|d2)!

We should make p("the") and p("for") less different for all docs.

Two-stage smoothing

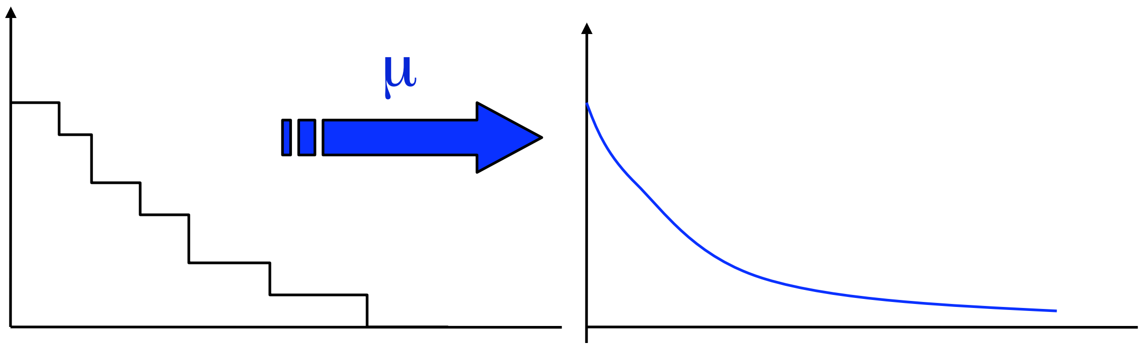


$$P(w|d) = \frac{c(w,d)}{|d|}$$

Two-stage smoothing

Stage-1

-Explain unseen words
-Dirichlet prior(Bayesian)



$$P(w|d) = \frac{c(w,d) + \mu p(w|C)}{|d| + \mu}$$

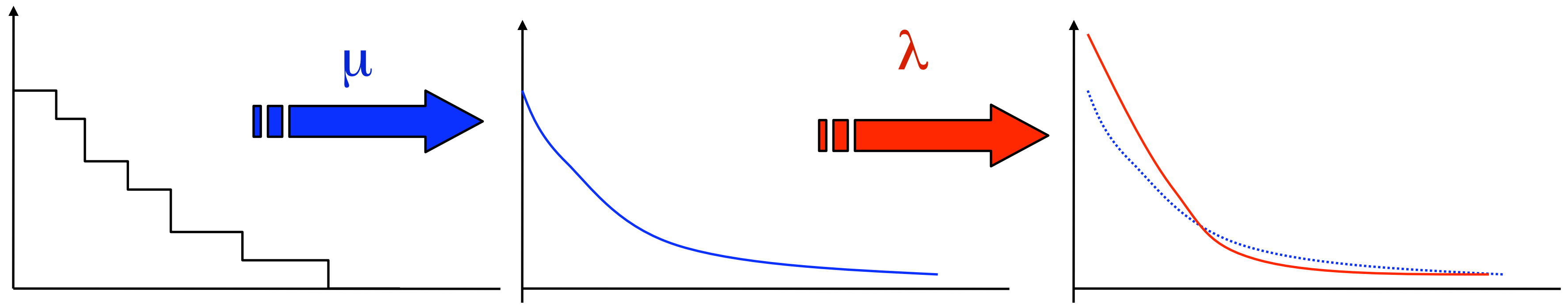
Two-stage smoothing

Stage-1

-Explain unseen words
-Dirichlet prior(Bayesian)

Stage-2

-Explain noise in query
-2-component mixture



$$P(w|d) = (1-\lambda)\frac{c(w,d) + \mu p(w|C)}{|d| + \mu} + \lambda p(w|U)$$

Translation model (Berger and Lafferty)

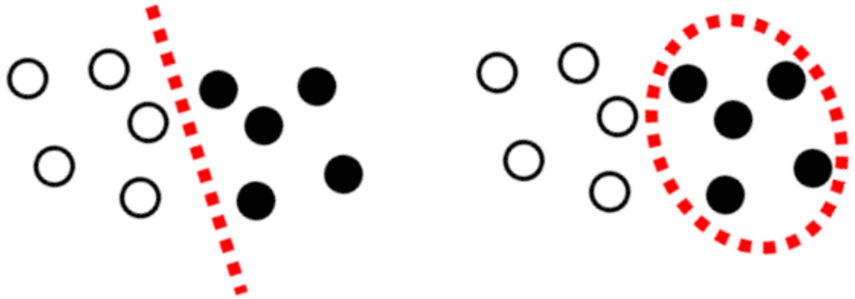
- Basic LMs do not address issues of synonymy.
  - Or any deviation in expression of information need from language of documents
- A translation model lets you generate query words not in document via “translation” to synonyms etc.
  - Or to do cross-language IR, or multimedia IR

$$P(\vec{q}\,|\,M) = \prod_i \sum_{v \in Lexicon} \underbrace{P(v\,|M)}_{\text{Basic LM}}\underbrace{T(q_i\,|\,v)}_{\text{Translation}}$$

  - Need to learn a translation model (using a dictionary or via statistical machine translation)

LM are generative techniques

- How do we determine if a given model is a LM?
- LM is generative
  - at some level, a language model can be used to generate text
  - explicitly computes probability of observing a string of text
  - Ex: probability of observing a query string from a document model
    probability of observing an answer from a question model
  - model an entire population
- Discriminative approaches
  - model just the decision boundary
  - Ex: is this document relevant?
    does it belong to class X or Y



- have a lot of advantages,
- but these are not generative approaches

LM: summary

- Goal: estimate a model M from a sample text S
- Use maximum-likelihood estimator
  - count the number of times each word occurs in S, divide by length
- Smoothing to avoid zero frequencies
  - discounting methods: add or subtract a constant, redistribute mass
  - better: interpolate with background probability of a word
  - smoothing has a role similar to IDF in classical models
- Smoothing parameters very important
  - Dirichlet works well for short queries (need to tune the parameter)
  - Jelinek-Mercer works well for longer queries (also needs tuning)
  - Lots of other ideas being worked on

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.

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