estimation

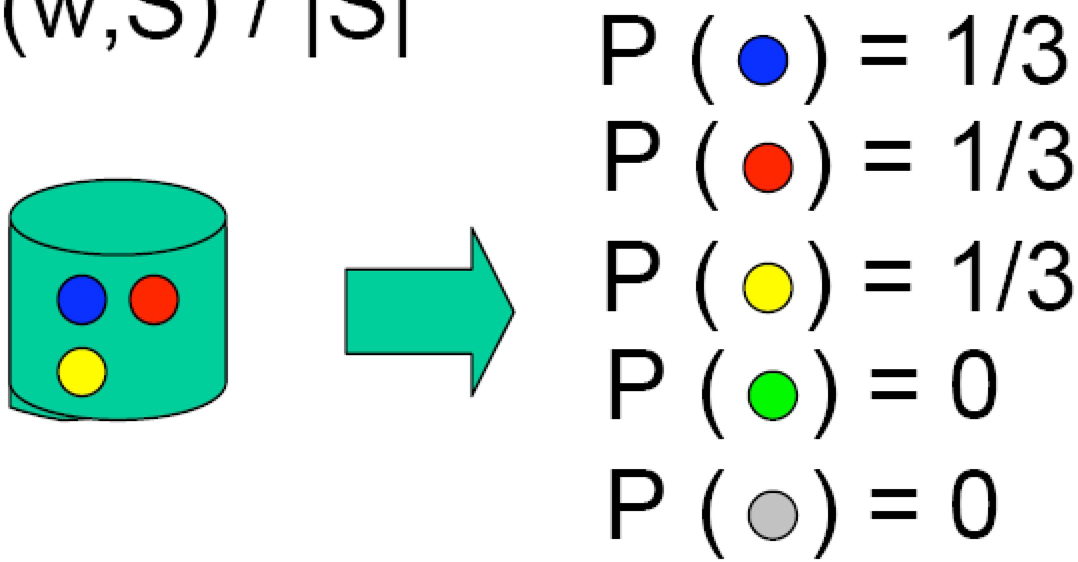
- Want to estimate MQ and/or MD from Q and/or D
- General problem:
  - given a string of text S (= Q or D), estimate its language model MS
  - S is commonly assumed to be an i.i.d. random sample from MS
    - Independent and identically distributed
- Basic Language Models
  - maximum-likelihood estimator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

30

maximum likelihood

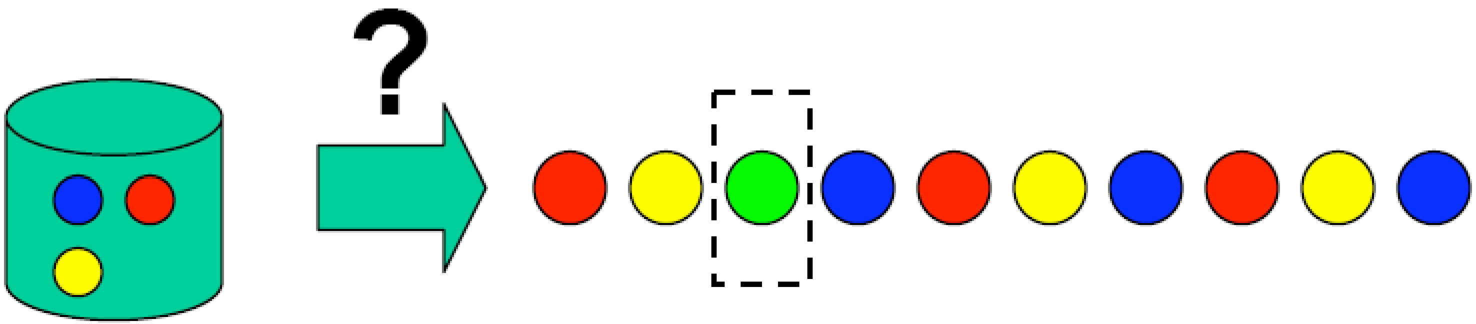
- count relative frequencies of words in S
- maximum-likelihood property:
  - assigns highest possible likelihood to the observation
- unbiased estimator:
  - if we repeat estimation an infinite number of times with different starting points S, we will get correct probabilities (on average)
  - this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

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



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



32

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 $(\frac{m_1}{N}, \frac{m_2}{N}, ..., \frac{m_k}{N})$

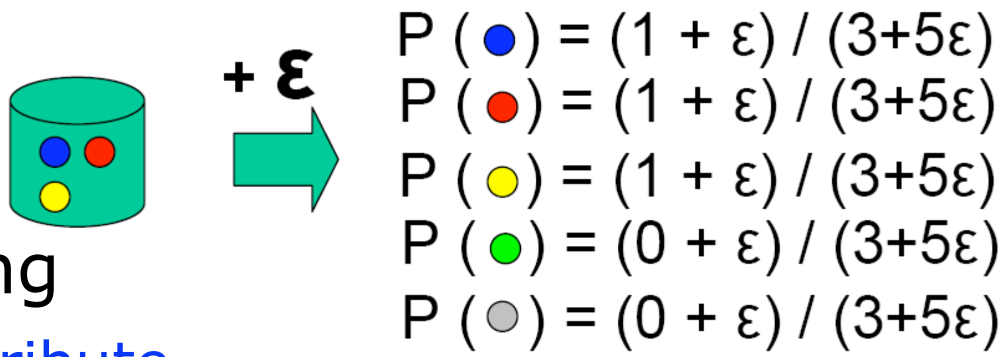
laplace estimates are $(\frac{m_1+1}{N+k}, \frac{m_2+1}{N+k}, ..., \frac{m_k+1}{N+k})$

33

discounting methods

- Laplace smoothing

- Lindstone correction
  - add **ε** to all count, renormalize



- absolute discounting
  - substract **ε** , redistribute probab mass

34

discounting methods

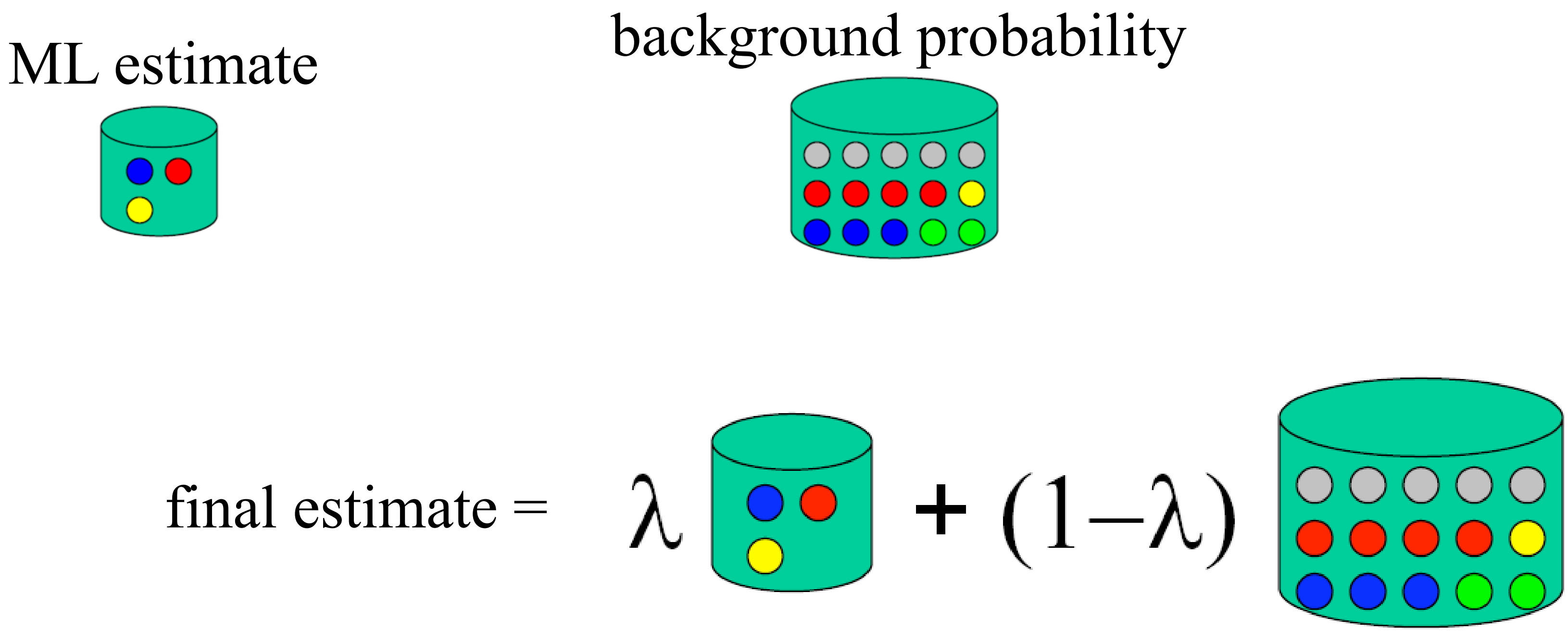
- Held-out estimation
  - Divide data into training and held-out sections
  - In training data, count Nr, the number of words occurring r times
  - In held-out data, count Tr, the number of times those words occur
  - r* = Tr/Nr is adjusted count (equals r if training matches held-out)
  - Use r*/N as 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

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

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

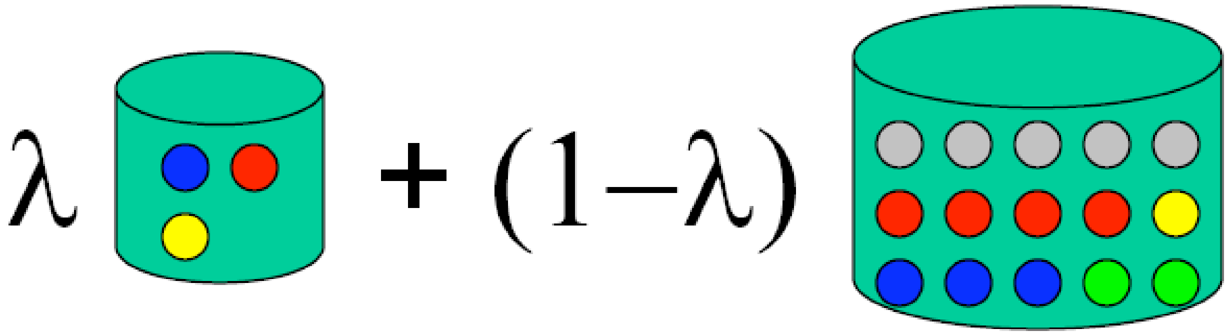
interpolation methods

- 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.



37

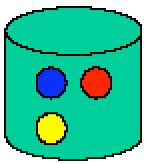
Dirichlet smoothing

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

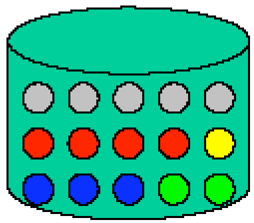
- Make smoothing depend on sample size

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

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



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

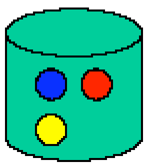
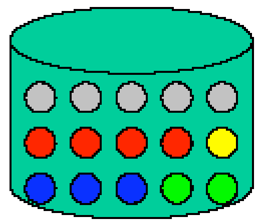


38

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

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

39

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

40

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

41

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

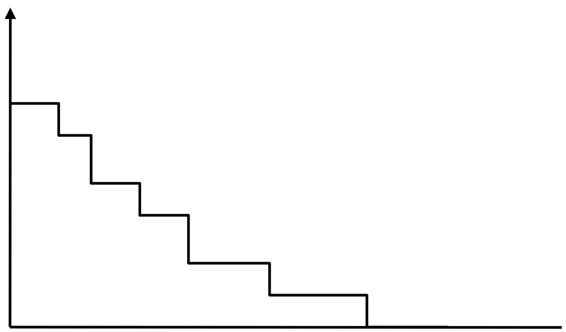
p("algorithms"|d1) = p("algorithm"|d2)
p("data"|d1) < p("data"|d2)
p("mining"|d1) < p("mining"|d2)

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

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

41

Two-stage smoothing



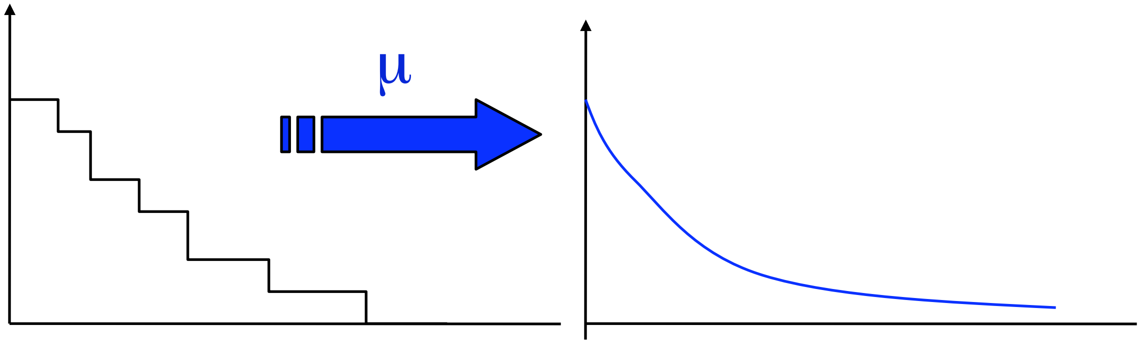
$$P(w|d) = \frac{c(w,d)}{|d|}$$

42

Two-stage smoothing

Stage-1

-Explain unseen words
-Dirichlet prior(Bayesian)



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

42

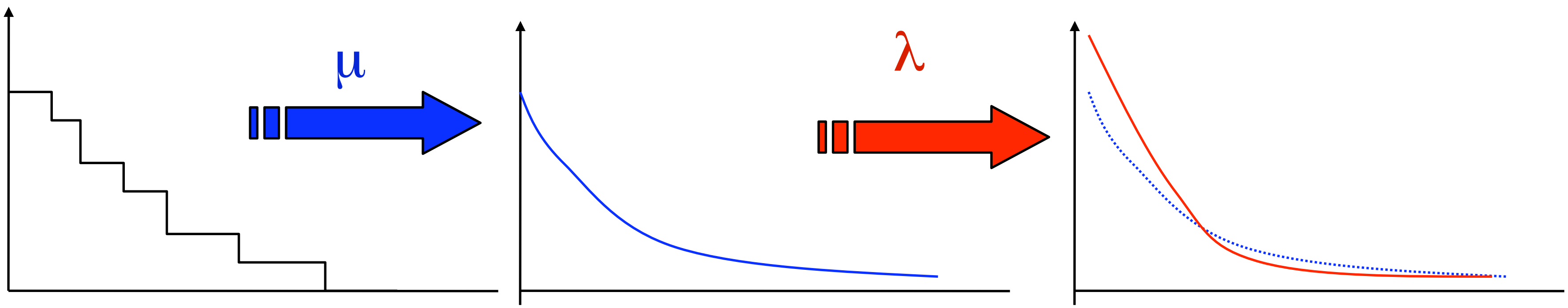
Two-stage smoothing

Stage-1

-Explain unseen words
-Dirichlet prior(Bayesian)

Stage-2

-Explain noise in query
-2-component mixture



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

42