Lecture 2
Retrieval Models
Part 1

Reference:

James Allan, University of Massachusetts Amherst
Pandu Nayak and Prabhakar Raghavan, Stanford University
Partially modified by Qingcai Chen, HIT Shenzhen

What is a retrieval model?

- Retrieval models (检索模型) can describe the computational process of IR
  - e.g. how documents are ranked
  - Note that how documents or indexes (索引) are *stored* is implementation
- Retrieval models can attempt to describe the human process
  - e.g. the information need, interaction
- Retrieval variables
  - queries (查询), documents (文档), terms (术语), relevance judgments (相关性判别), users, information needs, …
- Retrieval models have an explicit or implicit definition of relevance (相关度)

Models we’ll consider

- Boolean (布尔模型) (exact match)
- Statistical language models (统计语言模型)
- Vector space(向量空间）
- Latent Semantic Indexing （潜层语义分析）
- Inference network
- Classic probabilistic approaches（经典概率模型）
- Other models exist
  - Topological
  - Generalized vector space
  - Logic-based

in this lecture

Exact vs. Best Match

- Exact-match (精确匹配)
  - (例: “哈工大” ≠ “哈尔滨工业大学”, “哈工程” ≠ “哈尔滨工业大学”)
  - query specifies precise retrieval criteria
  - every document either matches or fails to match query
  - result is a set of documents
  - Unordered in pure exact match
- Best-match (最佳匹配)
  - (例: “哈工大” ≈” 哈尔滨工业大学”, 相似度80%, “哈工程” ≈” 哈尔滨工业大学”, 相似度50%)
  - Query describes good or “best” matching document
  - Every document matches query to some degree
  - Result is *ranked list* of documents
- Popular approaches often provide some of each
  - E.g., some type of ranking of result set
  - E.g., best-match query language that incorporates exact-match operators

(Unranked) Boolean retrieval

- Boolean model
  - is most common exact-match model
  - queries are logic expressions with document features as operands
  - In pure Boolean model, retrieved documents are not ranked
  - Most implementations provide some sort of ranking
  - query formulation difficult for novice users (新用户)
- Boolean queries (布尔查询)
  - Used by Boolean model
  - and in other models (Boolean query ≠ Boolean model)
- “Pure” Boolean operators
  - AND, OR, AND-NOT
- Most systems have proximity operators (邻接算子)
- Most systems support simple regular expressions (正则表达式) as search terms to match spelling variants

Unstructured data for Boolean retrieval

- Example:
  - Information needs: Which plays of Shakespeare contain the words Brutus AND Caesar but NOT Calpurnia?
  - 注:Brutus布鲁图 (85-42B.C.,罗马政治家,暗杀恺撒者之一)
    Calpurnia 是Julius Caesar的第三任，也是最后一位妻子

- One could *grep* (a command in Linux system) all of Shakespeare's plays for Brutus and Caesar, then strip out lines containing Calpurnia?
- Why is that not the answer?
  - Slow (for large corpora)
  - NOT Calpurnia is non-trivial
  - Other operations (e.g., find the word Romans near countrymen) not feasible

- What's the solution for such problem?

6

Document Representation: Term-document incidence

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

Brutus *AND* ***Caesar*** *BUT NOT* ***Calpurnia***

1 if play contains word, 0 otherwise

Incidence vectors（关联向量）

- So we have a 0/1 vector for each term.
- To answer query: take the vectors for ***Brutus, Caesar*** and ***Calpurnia*** (complemented) ➔ bitwise *AND*.

Proximity operators

- Proximity operators
  - Phrases - “two fish”
  - Same sentence “tiger /s photo”
    - To force order to be honored, “tiger +s photo”
  - Same paragraph - “southchina tiger” /p “photo”
    - Similarly, +p forces order
  - Word proximity “southchina /5 tiger” or “southchina +5 tiger”

Features to Note about Queries

- Queries are developed incrementally
  - Change query until reasonable number of documents retrieved
    - “language models”
    - “language models” /s statistical
    - “language models” /s stat!
    - (“language models” /s stat!) % toolkit
    - (“language models” /s stat!) % (toolkit HMM)
  - implicit relevance feedback
- Queries are complex
  - proximity operators used very frequently
  - implicit OR used for synonyms
  - NOT (%) is rare
- Queries are long (av. 9-10 words)
  - not typical Internet queries (1-2 words)

Boolean query languages still used

- Many users prefer Boolean
  - Especially professional searchers
  - “Control”
  - Understandability
- For some queries or collections, Boolean often works better (e.g., using AND on the Web)
- Boolean and free text find different documents
  - Need retrieval models that support both
  - “Extended Boolean”
    - vector space
    - Probabilistic inference network
- Need interfaces that provide good cognitive models (认知模型) for ranking (排序)

Models we'll consider

- *Boolean (exact match)*
- Statistical language models

Example: Small document

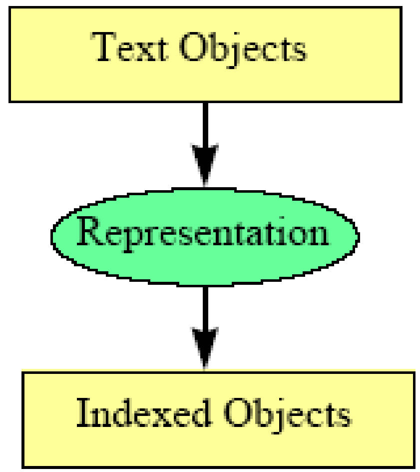
$$D = \begin{cases} \text{One fish, two fish, red fish, blue fish.} \\ \text{Black fish, blue fish, old fish, new fish.} \end{cases}$$

So we know that document D is more likely to talk about "fish" rather than "egg"!

Statistical language model: basic idea

Queries are used to describe the topic of user's needs.

- Document comes from a topic
- Topic (unseen) describes how words appear in documents on the topic
- Use document to guess what the topic looks like
  - Words common in document are common in topic
  - Words not in document much less likely
- Assign probability to words based on document
  - $P(w|Topic) \approx P(w|D) = tf(w,D) / len(D)$
- Index estimated topics



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

p_1 = P(“a quick brown dog”)

p_2 = P(“dog quick a brown”)

p_3 = P(“ 狗 brown dog”)

p_4 = P(“棕色狗”)

- … depends on what “language” we are modeling
  - In English: p1 > p2 > p3 > p4
  - In most of IR, assume that p1 == p2
  - for some applications we will want p3 to be highly probable, when?

A quick review of probabilistic

- Independence
  - If events w_1 and w_2 are independent then
    - $P(w_1 \text{ AND } w_2) = P(w_1) \cdot P(w_2)$

$$P(q_1, \ldots, q_t | D) = \prod_{i=1}^{t} P(q_i | D)$$

- Bayes' rule

$$P(B|A) = \frac{P(A|B) \cdot P(B)}{P(A)}$$

Language Modeling Notation

- Convenient to make explicit what we are modeling:
    - M … “language” we are trying to model
    - *s* … observation (string of tokens from some vocabulary)
    - P(*s*|M) … probability of observing “*s*” in language M
- M can be thought of as a “source” or a generator
  - a mechanism that can spit out strings that are legal in the language
    - E.g. “人活着就要吃东西” is a legal sentence in Chinese,
    - But “活着人东西吃” is not (generally) a sentence in Chinese (not include some novels or on Internet)
  - P(*s*|M) … probability of getting “s” during random sampling from M

Language Modeling for IR

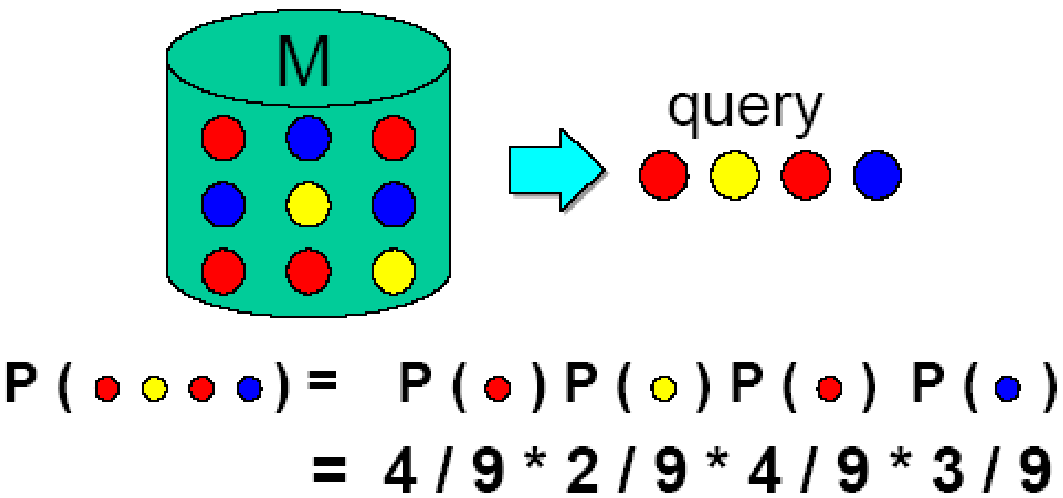
- Every document in a collection defines a "language"
  - consider all possible sentences (strings) that author could have written down when creating some given document
  - some are perhaps more likely to occur than others
    - subject to topic, writing style, language …
  - $P(s|M_D)$ … probability 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 $P(Q|M_D)$ [1]
  - probability of observing "Q" during random sampling from the language model of document D

Major issues in applying LMs

- Which kind of language model should we use?
  - Unigram or higher-order models?
- How can we estimate model parameters
  - Basic models
  - Translation models
  - non-parametric models
- How can we use the model for ranking?
  - Query-likelihood
  - Document-likelihood
  - Divergence (差异) of query and document models

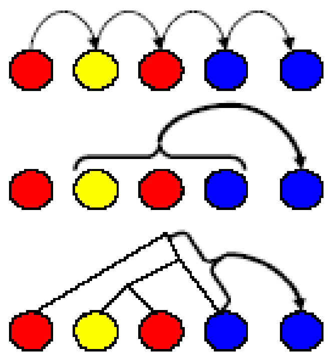
Unigram Language Models

- words are “sampled” independently of each other
  - Metaphor (一个比喻): randomly pulling words from an urn (缸) (with replacement)
  - joint probability decomposes into a product of each element
  - estimation of probabilities: simple counting



Higher-order Models

- 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 (passages, etc.)
  - parameter estimation is prohibitively expensive

Ranking with Language Models

- Standard approach: **query-likelihood (查询的似然排序)**
  - estimate a language model M_D for every document D in the collection
    - (assume that we know how to do this)
  - rank docs by the probability of “generating” the query

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

- Example:
- Q = “人民 创造”
- D1 = “在 漫长 的 历史 进程 中 中国 人民 辛勤 劳动 不懈 探索 勇于 创造 中国 人民 热爱 和平 ”
- D2= “中国 人民 勇于 创造”
-
-
-

Example (Cont'd)

D_1 = { This one, I think, is called a Yink.
He likes to wink, he likes to drink.

D_2 = { He likes to drink, and drink, and drink.
The thing he likes to drink is ink.

D_3 = { The ink he likes to drink is pink.
He links to wink and drink pink ink.

Query "drink"
- $P(drink|D_1) = 1/16$
- $P(drink|D_2) = 4/16$
- $P(drink|D_3) = 2/16$

Query "wink drink"
- $P(Q|D_1) = 0.004$
- $P(Q|D_2) = 0$
- $P(Q|D_3) = 1/16 \cdot 2/16 = 0.008$

Query "pink ink"
- $P(Q|D_1) = 0 \cdot 0 = 0$
- $P(Q|D_2) = 0 \cdot 1/16 = 0$
- $P(Q|D_3) = 2/16 \cdot 2/16 = 0.016$

Ranking with Query Likelihood

- Drawbacks:
  - 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 M_D
    - the only option is augmenting the original query Q with extra terms
    - however, we could make use of sample queries for which D is relevant
  - does not directly allow weighted or structured queries

Ranking: Document-likelihood (文档似然)

- Flip(翻转) the direction of the query-likelihood approach
  - estimate a language model M_Q for the query Q
  - rank docs D by the likelihood of being a random sample from M_Q
  - M_Q expected to “predict” a typical relevant document

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

- Problems:
  - different doc lengths, probabilities not comparable
  - favors documents that contain frequent (low content) words
  - consider “ideal” (highest-ranked) document for a given query*:
  - $$\max_D \prod_{w \in D} P(w \mid M_Q) = \max_{w \in D} P(w \mid M_Q)^n$$

Ranking: Likelihood Ratio

- Try to fix document likelihood: [8]
  - Bayes' likelihood that M_q was the source, given that we observed D

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

  - 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 M_Q
- Cons:
  - does not provide for modeling on the document side
  - "ideal" doc: $$\max_D \prod_{w \in D} \frac{P(w \mid M_Q)}{P(w \mid GE)} = \max_{w \in D} \frac{P(w \mid M_Q)^n}{P(w \mid GE)^n}$$

Ranking by Model Comparison

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

$$H(M_Q \,||\, M_D) = -\sum_{w} P(w|M_Q) \log P(w|M_D)$$

  - number of bits we would need to "encode" M_Q using M_D
  - equivalent to Kullback-Leiblar divergence: $H(M_Q \,||\, M_D) - H(M_Q \,||\, M_Q)$
  - equivalent to query-likelihood if M_Q is simply counts of words in Q
- Cross-entropy is not symmetric: use $H(M_Q \,||\, M_D)$
  - reverse works consistently worse, favors different document [9]
  - use reverse if ranking multiple queries w.r.t. one document

Summary of LM 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 Model Comparison for ranking
  - allows feedback, expansion, etc. through estimation of M_Q and M_D
  - use $H(M_Q \| M_D)$ for ranking multiple documents against a query
- Estimation of M_Q and M_D is a crucial step
  - very significant impact on performance (more than other choices)
  - key to cross-language, cross-media and other applications

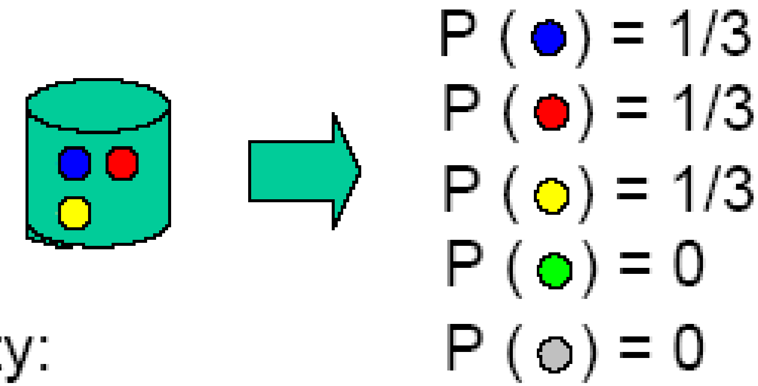
Estimation

- Want to estimate M_Q and/or M_D from Q and/or D
- General problem:
  - given a string of text S (= Q or D), estimate its language model M_S
  - S is commonly assumed to be an i.i.d. random sample from M_S
    - Independent and identically distributed
- Basic Language Models:
  - maximum-likelihood estimator and the zero frequency problem
  - discounting techniques:
    - Laplace correction, Lindstone correction, absolute discounting, leave-one-out discounting, Good-Turing method
  - interpolation/back-off techniques:
    - Jelinek-Mercer smoothing, Dirichlet smoothing, Witten-Bell smoothing, Zhai-Lafferty two-stage smoothing, interpolation vs. back-off techniques
  - Bayesian estimation

Maximum-likelihood

- count relative frequencies of words in S

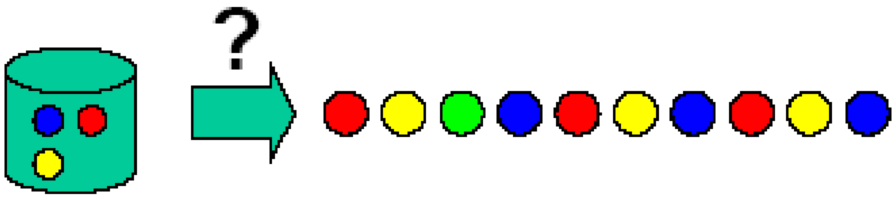
$$P_{ml}(w|M_S) = \#(w,S) / |S|$$



- 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…

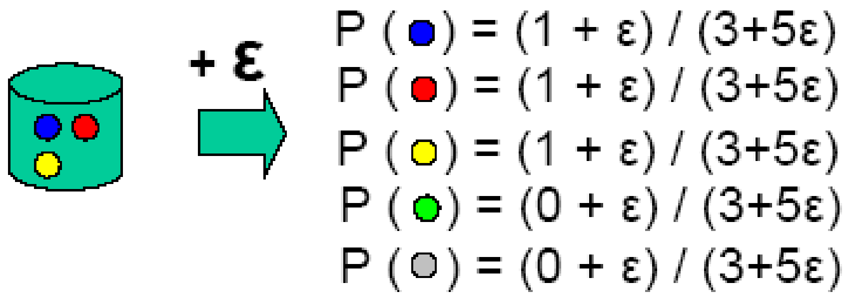
The Zero-frequency Problem

- 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



Discounting Methods (折扣法)[6]

- Laplace correction:
  - add 1 to every count, normalize
  - problematic for large vocabularies
- Lindstone correction:
  - add a small constant ***ε*** to every count, re-normalize



- Absolute Discounting
  - subtract a constant ***ε***, re-distribute the probability mass

Basic Models: 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 is 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

Homework 2

Back Up (Self-study)

*Discounting methods (cont.)

- Held-out estimation
  - Assume $P(w_1|M)=P(w_2|M)$ if w_1 and w_2 have same frequency
  - Divide data into training and held-out sections
  - In training data, count N_r, the number of words occurring *r* times
  - In held-out data, count T_r, the number of times those words occur
    - Doesn't matter how often they actually occur in held-out data
  - $r^* = T_r/N_r$ 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 $T_r(k)$ and $N_r(k)$
  - Estimate for words that occur *r* times is average of each

$$r^* = \frac{\frac{1}{K}\sum_{i=1}^{K} T_r(k)}{\frac{1}{K}\sum_{i=1}^{K} N_r(k)} = \frac{\sum_{i=1}^{K} T_r(k)}{\sum_{i=1}^{K} N_r(k)}$$

Discounting Methods (cont.)

- 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

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

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

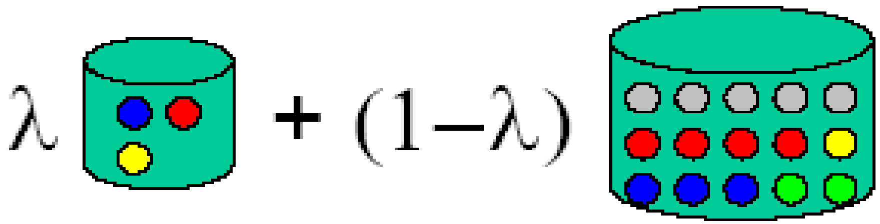
  - What good does this do?
  - Provides probability mass for words that occur r=0 times
    - Take what's leftover from r>0 to ensure adds to one
- Cleaning up Good-Turing estimation
  - What happens if there are no words that occur *r* times for some *r* ?
    - Makes estimate very unreliable for large values of *r*
  - Need to use expected count $E(N_r)$ and $E(N_{r+1})$
    - Regression to smooth out counts
    - Simply use maximum-likelihood probabilities r/N

$$\frac{E(N_{r+1})}{E(N_r)} \cdot \frac{r+1}{N}$$

  - Can also be derived from leaving out a single word at a time (K=|D|)
    - Called "leave one out" discounting

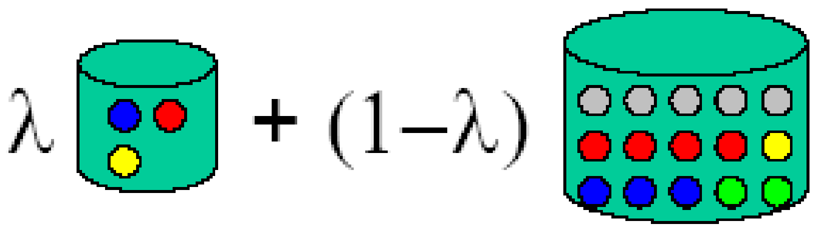
Interpolation Methods [6,7]

- 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
- 2-state HMM analogy



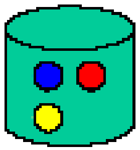
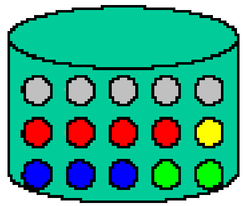
“Jelinek-Mercer” Smoothing

- 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.



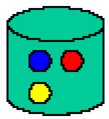
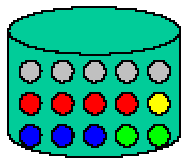
"Dirichlet" Smoothing

- Problem with Jelinek-Mercer:
  - longer documents provide better estimates
  - could get by with less smoothing
- Make smoothing depend on sample size
- Here N is length of sample and μ is a constant

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

“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
  - V is number of unique events

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

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 (HMM, mixture)

References for language modeling

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.