ECE8813
Statistical Natural Language Processing

Lectures 9 & 10: *N*-gram Estimation

Chin-Hui Lee
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332, USA
chl@ece.gatech.edu

Statistical NLP

- Some computational linguistics examples
  - Part-of-speech tagging for word sense disambiguation
  - Probabilistic parsing for sentence structures
  - Message understanding using semantics models
  - Statistical machine translation
  - Statistical transliteration
- Central to all problems in language modeling (LM)
  - Modeling of linguistic units and production rules
  - Discrete r. v. with very sparse observations
  - Language structure is crucial for efficient modeling

Probabilities of Word Sequences

- Language modeling (LM): Markov approximation

Given a sequence of word: $W = [w_1, \ldots, w_{|W|}]$, what is $P(W)$?

$$P(W) = P(w_1)P(w_2 \mid w_1)\cdots P(w_{|W|} \mid w_1, \ldots, w_{|W|-1}) \quad n-gram$$

$$\approx P(w_1)P(w_2 \mid w_1)\cdots P(w_n \mid w_1, \ldots, w_{n-1})\prod_{k=n}^{|W|} P(w_k \mid w_{k-1}, w_{k-2}, \ldots, w_{k-n+1})$$

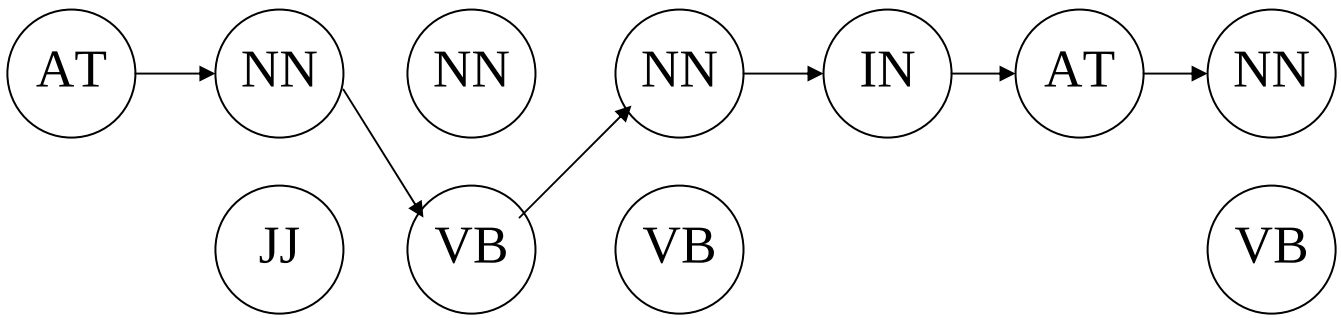
$$= P(w_1)P(w_2 \mid w_1)\prod_{k=n}^{|W|} P(w_k \mid w_{k-1}, w_{k-2}) \quad \text{trigram approximation}$$

- Many will argue that this is a poor assumption, and would not be able to handle nested linguistic structures, but the higher order n-gram are difficulty to estimate so that a trigram approximation has been a very effective one that follows Shannon's channel modeling paradigm

Problem Mapping of POS Tagging

- Finite state network (FSN) representation
  - State (node) space: the set of tags
  - Arc: tag transition (probabilities)
  - State output: tag-specific word probabilities
  - State-sequence: tag sequence
- An example:

The representative put chairs on the table.



Statistical POS Tagging

$$\hat{T} = \arg\max_{T \in \Psi} P(T \mid W)$$
$$= \arg\max_{T \in \Psi} P(W \mid T) P(T)$$

P(W|T): tag-specific word LM

P(T): tag language model

- Bigram tag language model approximation

$$P(T) = P(t_1^Q) \approx \prod_{q=1}^{Q} P(t_q \mid t_{q-1}) \quad P(t_1 \mid t_0) = 1$$

- Localized tag-specific language model

$$P(W \mid T) = P(w_1^Q \mid t_1^Q) \approx \prod_{q=1}^{Q} P(w_q \mid t_1^n) \approx \prod_{q=1}^{Q} P(w_q \mid t_q)$$

- Overall approximation

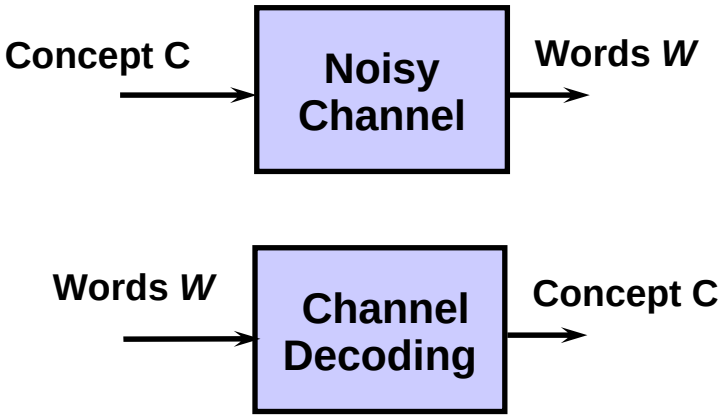
$$\hat{t}_1^Q = \arg\max_T P(W \mid T) P(T) \approx \arg\max_{t_1^Q} \prod_{q=1}^{Q} P(w_q \mid t_q) P(t_q \mid t_{q-1})$$

Problem Mapping for Text Understanding

- Finite state network (FSN) representation
  - State (node) space: the set of concepts
  - Arc: concept transition (probabilities)
  - State output: concept-specific word sequences
  - State-sequence: concept sequence (meaning expressed in sequence of semantic attributes)
- An example:

I want to flv to Boston from Dallas Friday noon on coach.

Statistical Concept Decoding



$$\hat{C} = \text{argmax}_{C \in \Psi} P(C \mid W)$$
$$= \text{argmax}_{C \in \Psi} P(W \mid C) P(C)$$

P(W|C): concept-specific word LM

P(C): concept language model

- Bigram concept language model approximation

$$P(C) = P(c_1^Q) \approx \prod_{q=1}^{Q} P(c_q \mid c_{q-1}) \quad P(c_1 \mid c_0) = 1$$

- Localized concept-specific bigram or trigram LM

$$P(W \mid C) = P(w_1^Q \mid c_1^Q) \approx \prod_{q=1}^{Q} P(w_1^Q \mid c_q) \approx \prod_{q=1}^{Q} P(w_{q-2}^q \mid c_q)$$

- Overall approximation

$$\hat{c}_1^Q = \arg\max_C P(W \mid C) P(C) \approx \arg\max_{c_1^Q} \prod_{q=1}^{Q} P(w_{q-2}^q \mid c_q) P(c_q \mid c_{q-1})$$

Some Issues before Moving on

- Under-sampling problems already in unigram
  - Too little data to estimate too many parameters
  - But we can not ignore unobserved events

$$U_1(x;V)=\begin{cases} f_x & 1\le x\le V \\ \varepsilon_1 & \text{otherwise} \end{cases} \quad \Omega_x=\{w_1,..,w_V,..\}$$

- For *n* greater, more estimation & storage problem
  - When *V*=60K, we need *V*x*V*x*V*=256 trillion trigrams
  - Serious underflow problem in computing
  - Hierarchical data structure is needed, but what and how?
- Recall multinomial distribution, what's the MLE?
  - Count the number of occurrences for unit events
  - Count the number of co-occurrences for joint/conditional events
- Are there better ways to count discrete events?

Text Corpora for *N*-gram Studies

- Existing: WSJ, Brown corpus, Treebank, AP wire, etc.
- Ongoing: million-book project (Internet Archive)
- For learning purpose: Project Gutenberg (small & doable): http://www.gutenberg.org/wiki/Main_page
- Jane Austen's novels (download on-line, 40GB) http://www2.hn.psu.edu/faculty/jmanis/j-austen.htm
  - Used in the Manning's textbook for illustration purposes
    - Training set: *Emma, Mansfield Park, Northanger Abbey, Pride and Prejudice, and Sense and Sensibility*
    - Testing set: *Persuasion*
    - *N*=617,091 single-words of text, *V*=14585 distinct words

Pre-processing: Clean-up and Normalization

- Handling of punctuations? capitalized words? Other?
- Bracketing of group of words (for easier modeling)
- 'Non-words': sentence beginning and ending marks, <UNK>
- Numerals: 12 vs. twelve
- Capitalized word can be used for some purposes
- What's needed is usually application-dependent
- Sometimes tokenization is important
  - e.g. no space between Chinese words, i.e. multiple word segmentations, many single-character words

Bernoulli Trials and Applications

- Binary Events:

$$P(A) = P("success") = p,\ P(\bar{A}) = P("failure") = q = 1 - p$$

- How about *k* successes in *n* independent trials?
  - How many such possibilities: *binomial coefficient*

$$_nC_k = \binom{n}{k} = \frac{1}{k!}[n*(n-1)\cdots*(n-k+1)] = \frac{n!}{k!(n-k)!}$$

$$p_n(k) = P(k \text{ successes in } n \text{ trials}) = \binom{n}{k} p^k q^{(n-k)}$$

Extension to Multinomial Distribution

- *Multinomial* Distribution: e.g. animal population

$$M(r_1,\ldots,r_M;N;p_1,\ldots,p_M)=\frac{N!}{r_1!\cdots r_M!}\prod_{i=1}^{M}p_i^{r_i}\quad 0\le r_i\quad \sum_{i=1}^{M}r_i=N$$

- *n*-gram usage:

1. r_i : i – th event for observing a specific n - gram, $e_i=(w_1,\ldots,w_n)$
2. N_n : total number of n - gram events observed in the corpus : $\sum_i C(e_i)=N_n$
3. Total number of distinct events of interests : $M=|V|^n$
4. Conditional event $(W\mid w_1,\ldots,w_{n-1})$ is a unigram distribution over all W
5. Each unigram r.v. follows a multinomial distribution
6. Issue with unobserved events and spare training data

Observing *N*-Gram Estimates

- Looking into Table 6.3 for examples from Austen
- Unigram: Zipf's Law again
  - "inferior" is less common than "to"
- Bigram: remember collocation
  - P("to"|"inferior")=0.212, a very high combination
- Trigram: many unseen events
- 4-gram: even more unseen events

Generalization Issues

- Set aside some data for cross-validation but there is only very little training data
  - Too many parameters: over-fitting model will get good scores on training data but usually does not generalize to unseen testing data
  - Regularization: adding penalty terms to penalize too good over-fitting of training data
  - Dividing *training set* into initial training and *held-out set or development set*
  - Always testing models on *unseen evaluation sets*
  - Sometimes imposing the *cross-validation* strategy

Statistical Estimators

- Example:
  - Corpus: five Jane Austen novels
  - N = 617,091 words, V = 14,585 unique words
  - Task: predict the next word of the trigram “inferior to ___”
    - from test data, *Persuasion*: “[In person, she was] inferior to *both* [sisters.]”
- Given the observed training data …
- How do you develop a model (probability distribution) to predict future events?

The Perfect Language Model

- Sequence of word forms
- Notation: $W = (w_1, w_2, w_3, \ldots, w_d)$
- The big (modeling) question is "what is $p(W)$"?
- Well, we know (Bayes/chain rule):
  - $p(W) = p(w_1, w_2, w_3, \ldots, w_d) = p(w_1) \times p(w_2|w_1) \times p(w_3|w_1, w_2) \times \ldots \times p(w_d|w_1, w_2, \ldots, w_{d-1})$
- Not practical (even for short W there are still too many parameters)

Markov Chain

- Unlimited memory (cf. previous foil):
  - for w_i, we know <u>all</u> its predecessors $w_1, w_2, w_3, \ldots, w_{i-1}$
- Limited memory:
  - we disregard predecessors that are "too old"
  - remember only k previous words: $w_{i-k}, w_{i-k+1}, \ldots, w_{i-1}$
  - called "k^{th} order Markov approximation"
- Stationary character (no change over time):
  - $p(W) = \Pi_{i=1..d} p(w_i | w_{i-n+1}, w_{i-n+2}, \ldots, w_{i-1})$ $d = |W|$

N-gram Language Models

(n-1)th order Markov approximation gives n-gram LM:

$$p(W) = \prod_{i=1..d} p(w_i | w_{i-n+1}, w_{i-n+2}, \ldots, w_{i-1})$$

- In particular (assume vocabulary size $|V|$ = 20k):
  - 0-gram : uniform model — $p(w) = 1/|V|$ — 1 parameter
  - 1-gram : unigram model — $p(w)$ — 2×10^4 parameters
  - 2-gram : bigram model — $p(w_i|w_{i-1})$ — 4×10^8 parameters
  - 3-gram : trigram mode — $p(w_i|w_{i-2},w_{i-1})$ — 8×10^{12} parameters
  - 4-gram : tetragram model — $p(w_i| w_{i-3},w_{i-2},w_{i-1})$ — 1.6×10^{17} parameters

Reliability vs. Discrimination

- "large green _____"
  tree? mountain? frog? car?
- "swallowed the large green _____"
  pill? tidbit?

- Larger *n*: more information about the context of the specific instance (greater discrimination)
- Smaller *n*: more instances in training data, better statistical estimates (more reliability)

LM Observations

- How large *n*?
  - Zero is enough (theoretically)
  - But anyway: as much as possible (as close to “perfect” model as possible)
  - Empirically: ***3***
    - parameter estimation? (reliability, data availability, storage space, ...)
    - 4 is too much: $|V|$ =60k gives 1.296×10^{19} parameters
    - but: 6-7 would be (almost) ideal (having enough data)
  - Reliability decreases with increase in detail (need compromise)
- For now, word forms only

Parameter Estimation

- Parameter: numerical value needed to compute *p(w|h)*
- From data (how else?)
- Data preparation:
  - get rid of formatting etc. (“text cleaning”)
  - define words (separate but include punctuation, call it “word”, unless speech)
  - define sentence boundaries (insert “words” <s> and </s>)
  - letter case: keep, discard, or be smart:
    - name recognition
    - number type identification
  - numbers: keep, replace by <num>, or be smart (form ~ pronunciation)

Maximum Likelihood Estimation of *N*-grams

- Properties of *n*-grams

$$P(w_n \mid w_1, \ldots, w_{n-1}) = \frac{P(w_1, \ldots, w_{n-1}, w_n)}{P(w_1, \ldots, w_{n-1})},$$

$$\sum_{w_n \in V} P(w_n \mid w_1, \ldots, w_{n-1}) = 1,$$

$$\sum_i C(e_i) = N_n \quad e_i : i\text{-th event}$$

- *MLE of* Multinomial Distribution Parameters

$$P_{MLE}(w_1, \ldots, w_{n-1}, w_n) = \frac{C(w_1, \ldots, w_{n-1}, w_n)}{N_n},$$

$$P_{MLE}(w_n \mid w_1, \ldots, w_{n-1}) = \frac{C(w_1, \ldots, w_{n-1}, w_n)}{C(w_1, \ldots, w_{n-1})},$$

$$\sum_{W \in V} C(w_1, \ldots, w_{n-1}, W) = C(w_1, \ldots, w_{n-1})$$

Maximum Likelihood Estimate

- MLE: Relative Frequency...
  - ...best predicts the data at hand (the "training data")
- Trigrams from Training Data *T*:
  - count sequences of three words in *T*: $C_3(w_{i-2},w_{i-1},w_i)$
  - count sequences of two words in *T*: $C_2(w_{i-2},w_{i-1})$:

$$P_{\text{MLE}}(w_i|w_{i-2},w_{i-1}) = C_3(w_{i-2},w_{i-1},w_i) \,/\, C_2(w_{i-2},w_{i-1})$$

Character Language Model

- Use individual characters instead of words:

$$p(W) = \prod_{i=1..d} p(c_i | c_{i-n+1}, c_{i-n+2}, \ldots, c_{i-1})$$

- Same formulas and methods
- Might consider 4-grams, 5-grams or even more
- Good for cross-language comparisons
- Transform cross-entropy between letter- and word-based models:
  - $H_S(p_c) = H_S(p_w)$ / avg. # of characters per word in S

LM: An Example

Training data: <s_0> <s> He can buy you the can of soda </s>

- Unigram: (8 words in vocabulary)

  p_1(He) = p_1(buy) = p_1(you) = p_1(the) = p_1(of) = p_1(soda) = .125 p_1(can) = .25

- Bigram:

  p_2(He|<s>) = 1, p_2(can|He) = 1, p_2(buy|can) = .5, p_2(of|can) = .5, p_2(you |buy) = 1,...

- Trigram:

  p_3(He|<s_0>,<s>) = 1, p_3(can|<s>,He) = 1, p_3(buy|He,can) = 1, p_3(of|the,can) = 1, ...p_3(</s>|of,soda) = 1.

- Entropy: $H(p_1)$ = 2.75, $H(p_2)$ = 1, $H(p_3)$ = 0

LM: an Example (The Problem)

- Cross-entropy:

  S = <s_0> <s> It was the greatest buy of all </s>

- Even $H_S(p_1)$ fails because:
  - all unigrams but p_1(the), p_1(buy), and p_1(of) are 0
  - all bigram probabilities are 0
  - all trigram probabilities are 0
- Need to make all "theoretically possible" probabilities non-zero

LM: Another Example

- Training data S: |V| =11 (not counting <s> and </s>)

  <s> John read Moby Dick </s>

  <s> Mary read a different book </s>

  <s> She read a book by Cher </s>

- Bigram estimates:

  P(She | <s>) = C(<s> She)/ Sum_w C(<s> w) = 1/3

  P(read | She) = C(She read)/ Sum_w C(She w) = 1

  P (Moby | read) = C(read Moby)/ Sum_w C(read w) = 1/3

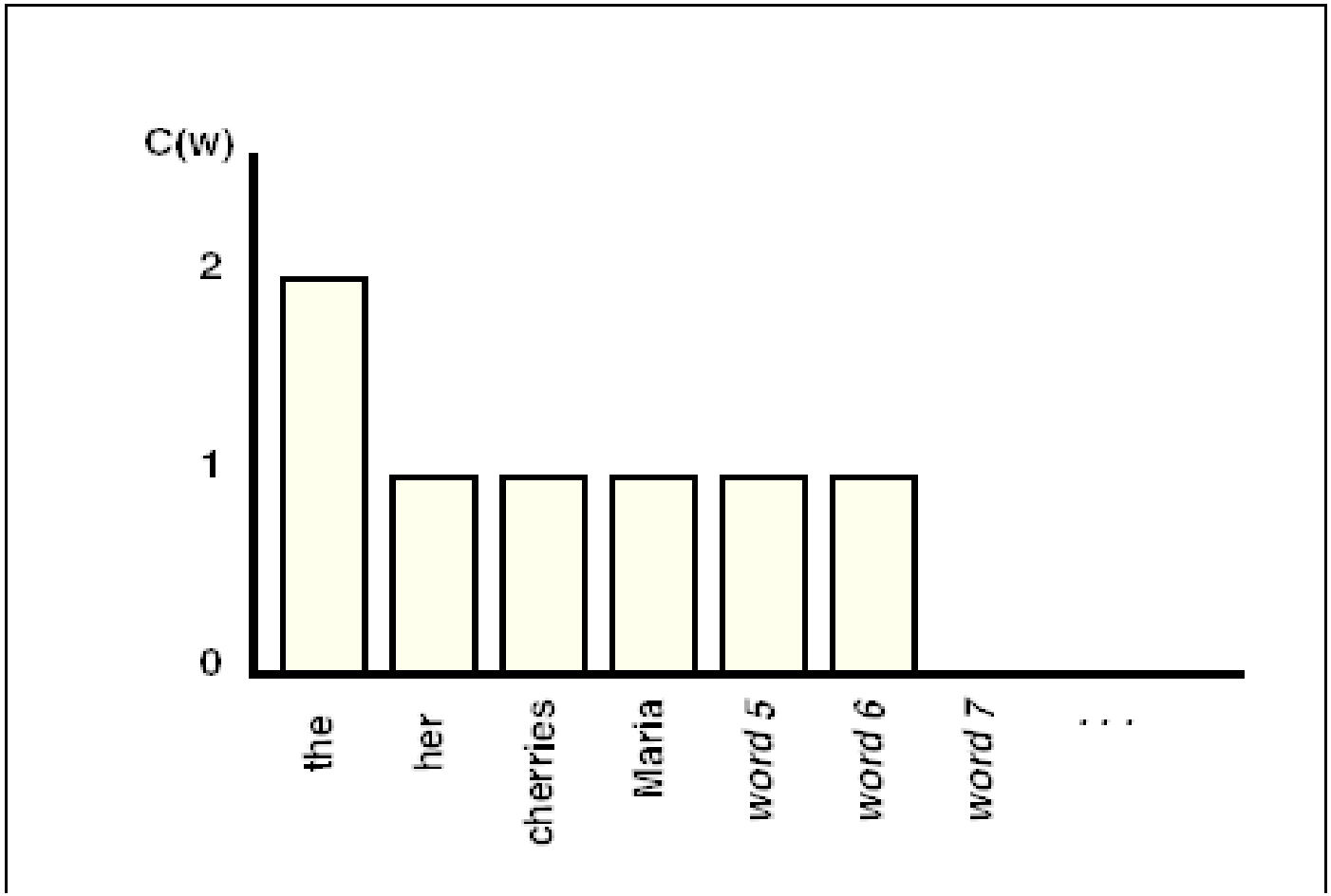
  P (Dick | Moby) = C(Moby Dick)/ Sum_w C(Moby w) = 1

  P(</s> | Dick) = C(Dick </s>)/ Sum_w C(Dick w) = 1

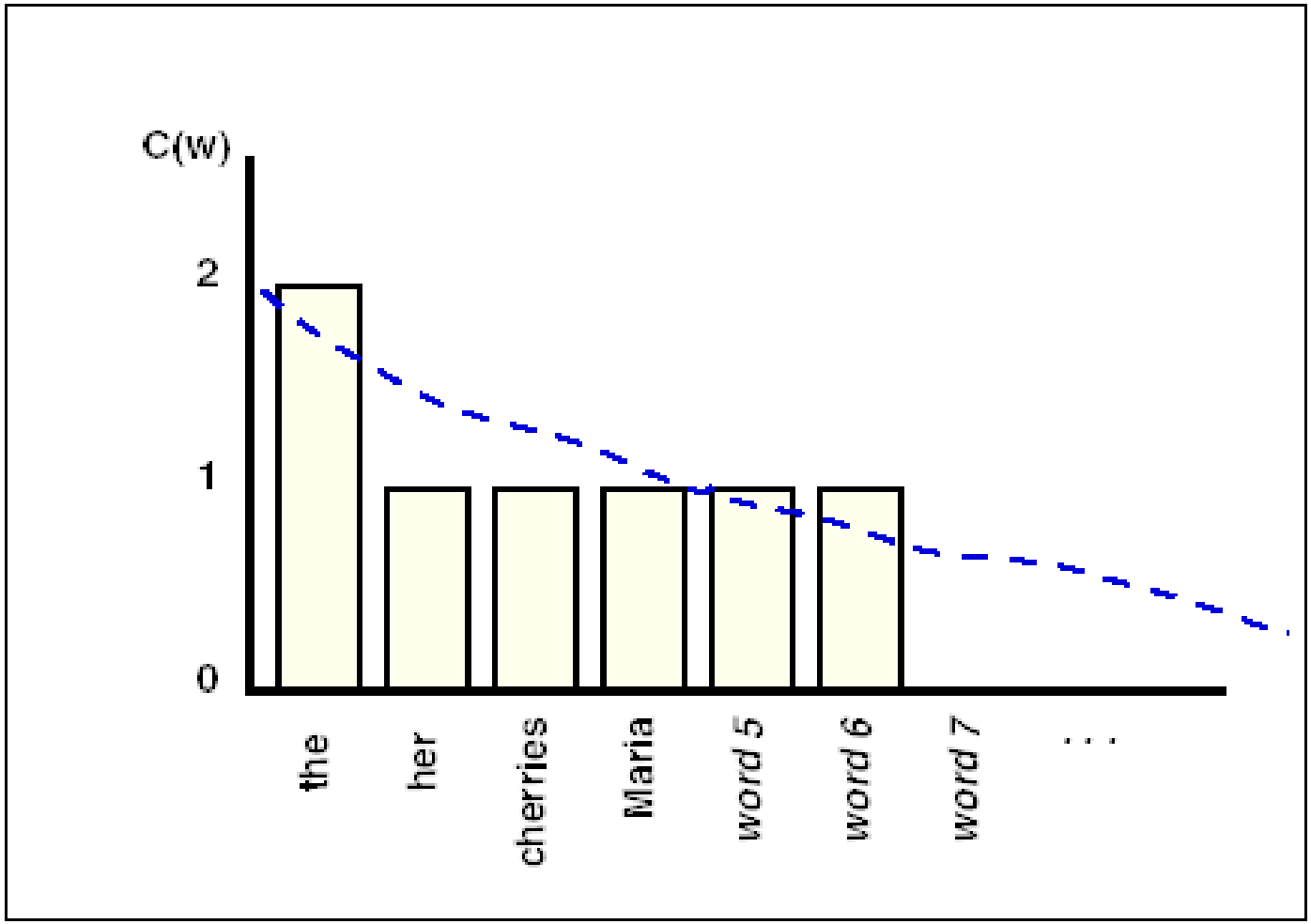
- p(She read Moby Dick) =

  p(She | <s>) $\times$ p(read | She) $\times$ p(Moby | read) $\times$ p(Dick | Moby) $\times$ p(</s> | Dick) = 1/3 $\times$ 1 $\times$ 1/3 $\times$ 1 $\times$ 1 = 1/9

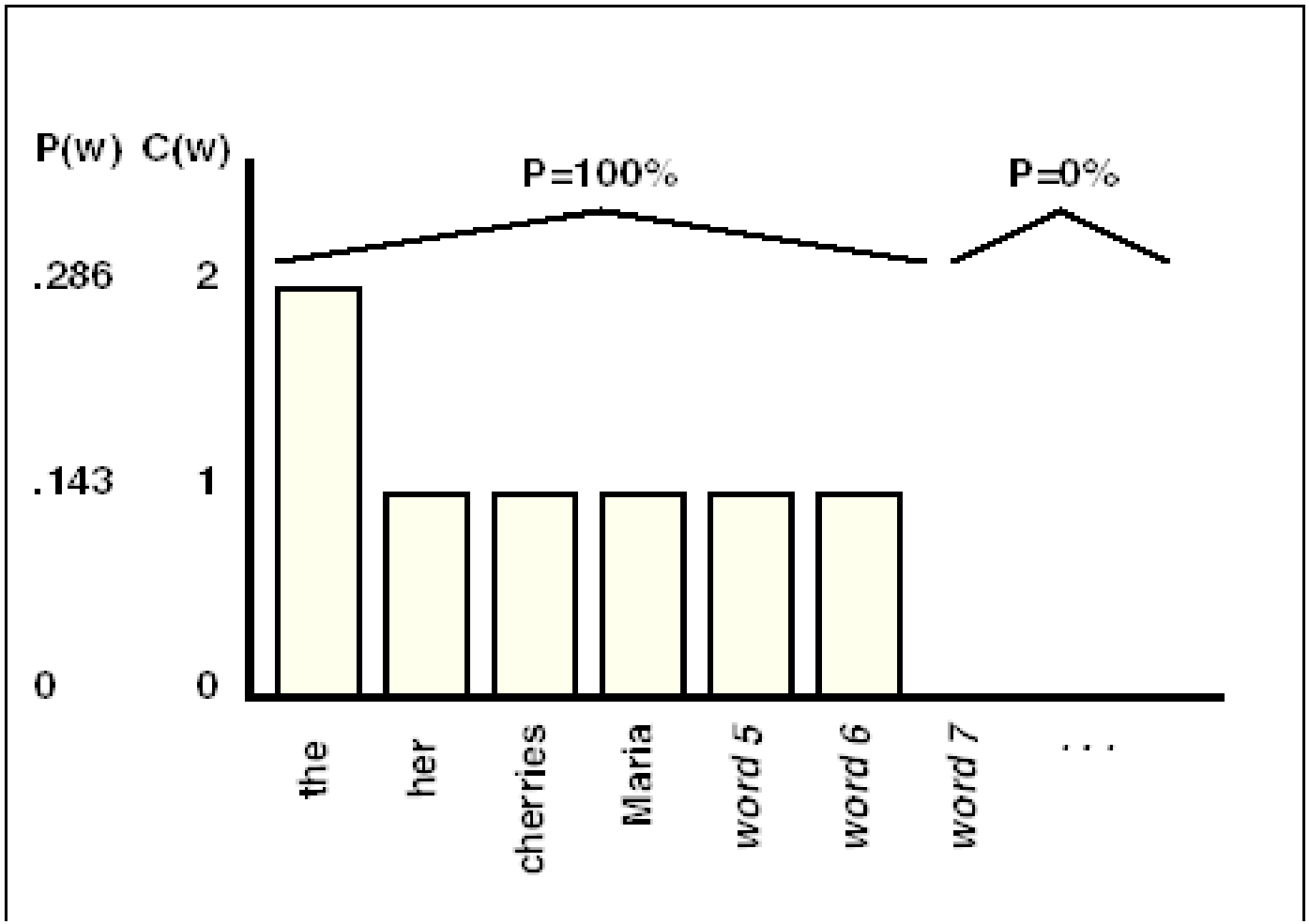
Training Corpus Instances: *"inferior to___"*



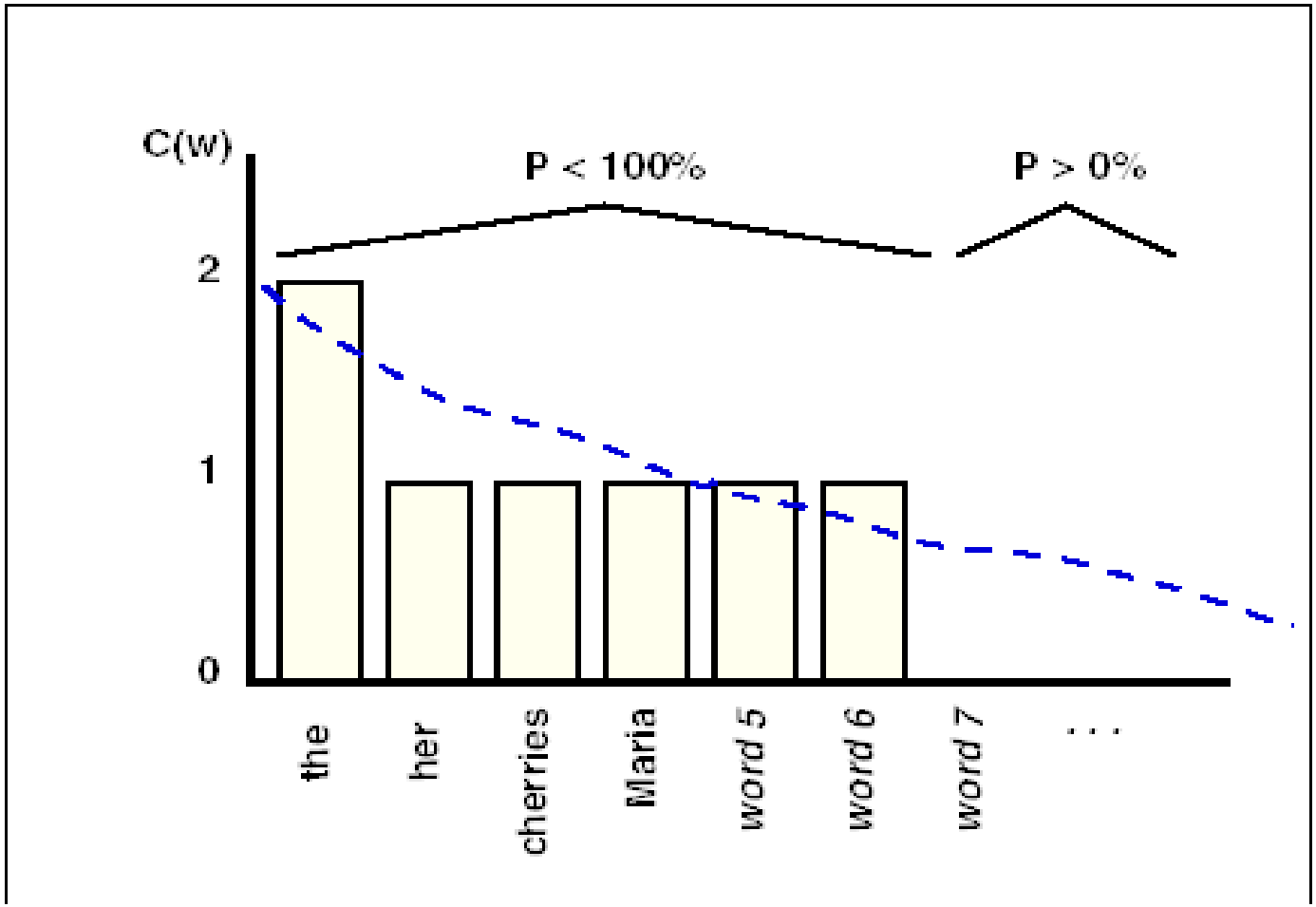
Actual Probability Distribution



Maximum Likelihood Estimate



Comparison



The Zero Cell Problem

- "Raw" *n*-gram language model estimate:
  - Necessarily, there will be some zeros
    - Often trigram model gives 2.16×10^{14} parameters, and the required data ~ 10^9 words
  - Which are true zeros?
    - optimal situation: even the least frequent trigram would be seen several times, in order to distinguish it's probability vs. other trigrams (hapax legomena)
    - optimal situation cannot happen, unfortunately (question: how much data would we need?)
  - We don't know; hence, we eliminate them
- Different kinds of zeros: $p(w|h) = 0$, $p(w) = 0$

Need Nonzero Probabilities?

- Avoid infinite Cross Entropy:
  - happens when an event is found in the test data which has not been seen in training data
- Make the system more robust
  - low count estimates:
    - they typically happen for “detailed” but relatively rare appearances
  - high count estimates: reliable but less “detailed”

Eliminating Zero Probability: Smoothing

- Get new *p'(w)* (same *W*): almost *p(w)* except for eliminating zeros
- Discount *w* for some *p(w)* > 0: new *p'(w)* < *p(w)*

  Sum $_{\text{discounted}}$ *(p(w) - p'(w)) = D*
- Distribute *D* to all *w*; *p(w) = 0*: new *p'(w)* > *p(w)*
  - possibly also to other *w* with low *p(w)*
- For some *w* (possibly): *p'(w) = p(w)*
- Make sure $\text{Sum}_W\, p'(w) = 1$
- There are many ways of ***smoothing***

Improving MLE by Discounting

- Handle out-of-vocabulary (OOV) classes
  - Not seen in training: count = 0 or 1 (<UNK>)
- Laplace Law (adding one): more for unseen events
  - Bayesian estimates assuming a uniform prior
  - 99.97% probability mass given to unseen bigrams (Table 6.4)

$$p_{\text{Lap}} = [C(w_1, \ldots, w_n) + 1] / [C(\text{total}) + M]$$

- Lidstone's Law $p_{\text{Lid}} = [C(w_1, \ldots, w_n) + \lambda] / [C(\text{total}) + M\lambda], \lambda < 1$

$$= \mu * \frac{C(w_1, \ldots, w_n)}{C(\text{total})} + (1-\mu)\frac{1}{M}, \mu = \frac{C(\text{total})}{C(\text{total}) + M\lambda}$$

- Jeffrey-Perks Law: Expected Likelihood Estimation

$$\lambda = 0.5 \quad \text{and } \mu = \frac{C(\text{total})}{C(\text{total}) + 0.5M}$$

Laplace's Law: Smoothing by Adding 1

- Laplace's Law:
  - $P_{LAP}(w_1,..,w_n)=(C(w_1,..,w_n)+1)/(N+B)$, where $C(w_1,..,w_n)$ is the frequency of n-gram $w_1,..,w_n$, N is the number of training instances, and B is the number of bins training instances are divided into (vocabulary size)
  - Problem if $B > C(W)$ (can be the case; even >> C(W))
  - $P_{LAP}(w \mid h) = (C(h,w) + 1) \ / \ (C(h) + B)$
- The idea is to give a little bit of the probability space to unseen events

Add 1 Smoothing Example

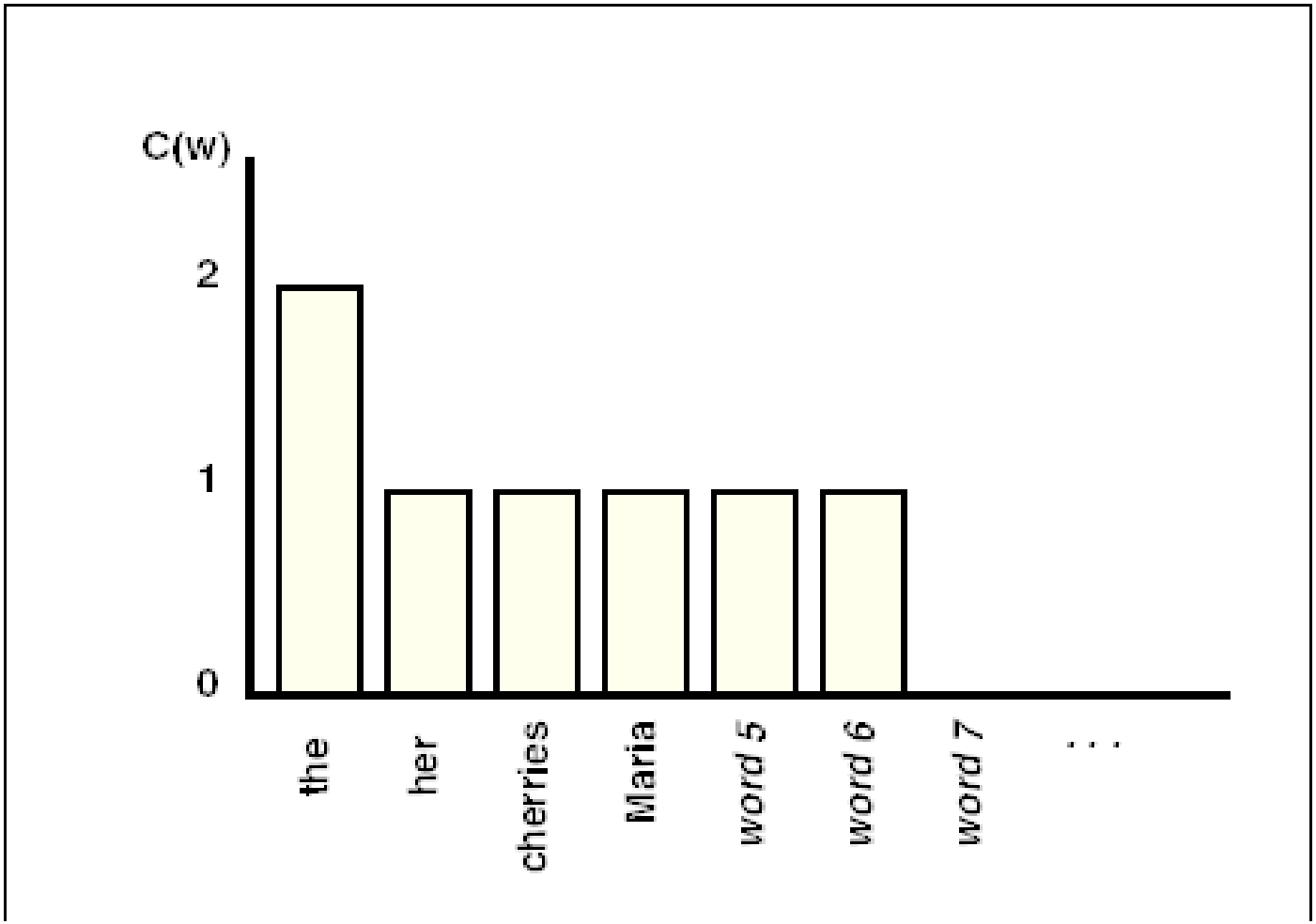
p_{MLE}(Cher read Moby Dick) =

p(Cher | <s>) × p(read | Cher) × p(Moby | read) × p(Dick | Moby) × p(</s> | Dick) = 0 × 0 × 1/3 × 1 × 1 = 0

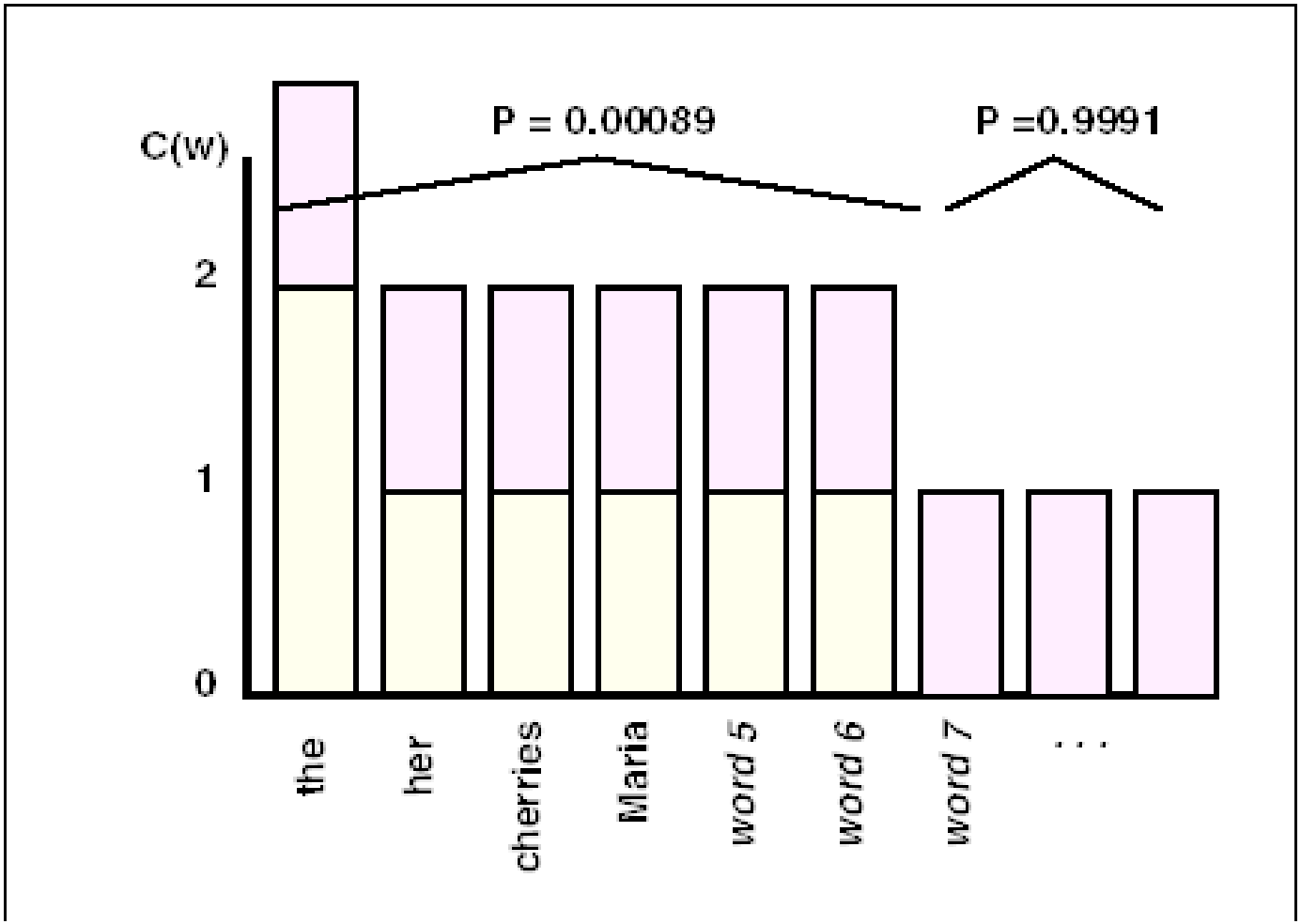
- p(Cher | <s>) = (1 + C(<s> Cher))/(11 + C(<s>)) = (1 + 0) / (11 + 3) = 1/14 = .0714
- p(read | Cher) = (1 + C(Cher read))/(11 + C(Cher)) = (1 + 0) / (11 + 1) = 1/12 = .0833
- p(Moby | read) = (1 + C(read Moby))/(11 + C(read)) = (1 + 1) / (11 + 3) = 2/14 = .1429
- P(Dick | Moby) = (1 + C(Moby Dick))/(11 + C(Moby)) = (1 + 1) / (11 + 1) = 2/12 = .1667
- P(</s> | Dick) = (1 + C(Dick </s>))/(11 + C<s>) = (1 + 1) / (11 + 3) = 2/14 = .1429

p'(Cher read Moby Dick) = p(Cher | <s>)×p(read | Cher)×p(Moby | read) × p(Dick | Moby)×p(</s> | Dick) = 1/14×1/12×2/14×2/12×2/14 = $2.02e^{-5}$

Laplace's Law *(Rriginal)*



Laplace's Law (Adding One)



Objections to Laplace's Law

- For NLP applications that are very sparse, Laplace's Law actually gives far too much of the probability space to unseen events
- Worse at predicting the actual probabilities of bigrams with zero counts than other methods
- Count variances are actually greater than the MLE

Lidstone's Law

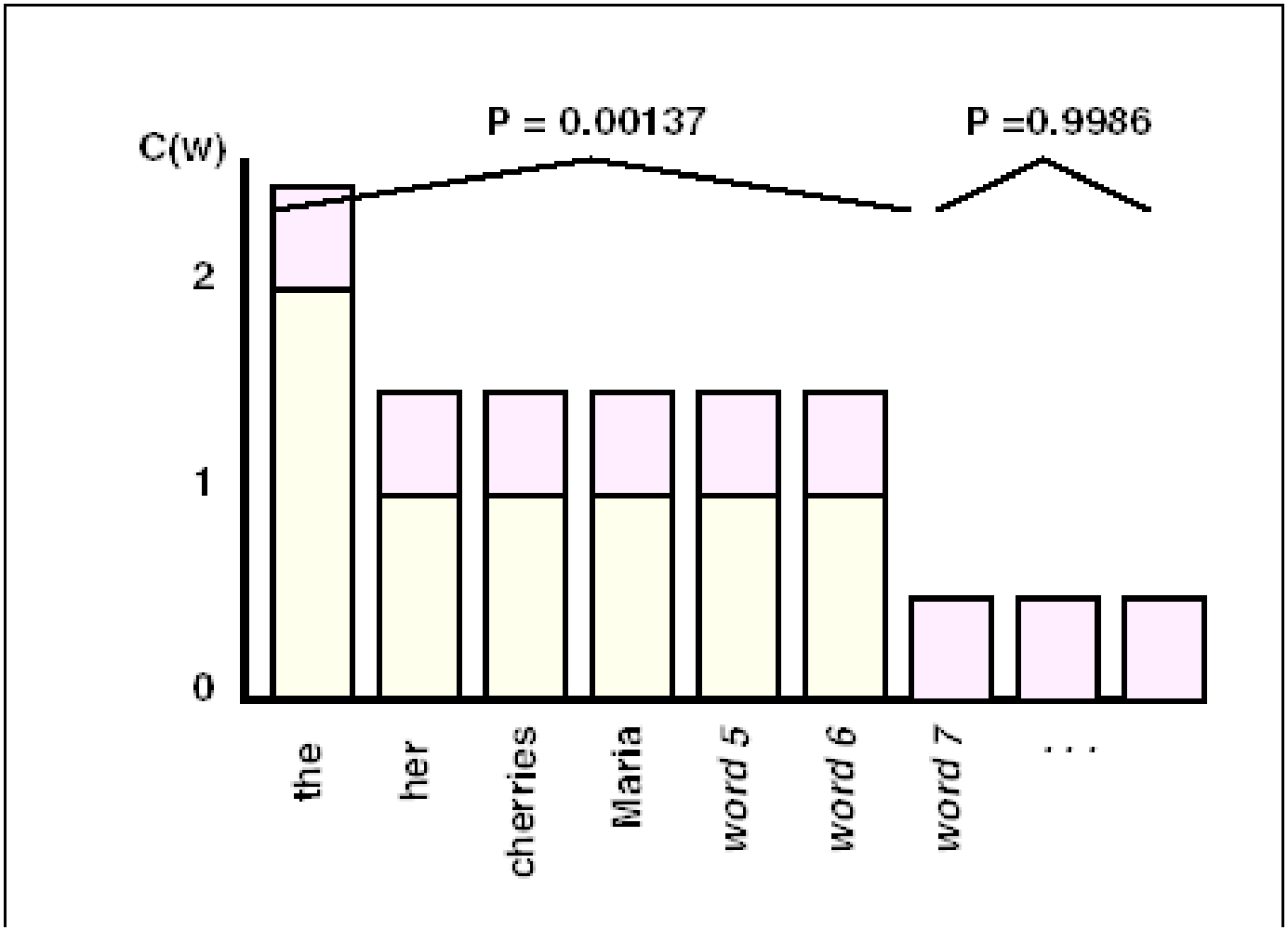
$$P_{Lid}(w_1 \cdots w_n) = \frac{C(w_1 \cdots w_n) + \lambda}{N + B\lambda}$$

P = probability of specific n-gram
C = count of that n-gram in training data
N = total *n*-grams in training data
B = number of "bins" (possible n-grams)
λ = small positive number

MLE: $\lambda = 0$
LaPlace's Law: $\lambda = 1$
Jeffreys-Perks Law: $\lambda = ½$

$P_{Lid}(w \mid h) = (C(h,w) + \lambda) \;/\; (C(h) + B\,\lambda)$

Jeffreys-Perks Law



Objections to Lidstone's Law

- Need an *a priori* way to determine λ
- Predicts all unseen events to be equally likely
- Gives probability estimates linear in the M.L.E. frequency

Lidstone's Law with λ=.5

p_{MLE}(Cher read Moby Dick) =

p(Cher | <s>) × p(read | Cher) × p(Moby | read) × p(Dick | Moby) × p(</s> | Dick) = 0 × 0 × 1/3 × 1 × 1 = 0

p(Cher | <s>) = (.5 + C(<s> Cher))/(.5* 11 + C(<s>)) = (.5 + 0) / (.5*11 + 3) = .5/8.5 =.0588

p(read | Cher) = (.5 + C(Cher read))/(.5* 11 + C(Cher)) = (.5 + 0) / (.5* 11 + 1) = .5/6.5 = .0769

p(Moby | read) = (.5 + C(read Moby))/(.5* 11 + C(read)) = (.5 + 1) / (.5* 11 + 3) = 1.5/8.5 = .1765

P(Dick | Moby) = (.5 + C(Moby Dick))/(.5* 11 + C(Moby)) = (.5 + 1) / (.5* 11 + 1) = 1.5/6.5 = .2308

P(</s> | Dick) = (.5 + C(Dick </s>))/(.5* 11 + C<s>) = (.5 + 1) / (.5* 11 + 3) = 1.5/8.5 = .1765

p'(Cher read Moby Dick) =

p(Cher | <s>) × p(read | Cher) × p(Moby | read) × p(Dick | Moby) × p(</s> | Dick) = .5/8.5 × .5/6.5 × 1.5/8.5 × 1.5/6.5 × 1.5/8.5 = $3.25e^{-5}$

Held-Out Estimator

- How much of the probability distribution should be reserved to allow for previously unseen events?
- Can validate choice by holding out part of the training data
- How often do events seen (or not seen) in training data occur in validation data?
- Held-out estimator by Jelinek and Mercer (1985)

Held-Out Estimation

- Held-out estimator, define

$$(C^n(w_1^n)) = \sum_{\{w_1^n : C^n_{\text{train}}(w_1^n) = r\}} 1$$

$$T^n(r) = \sum_{\{w_1^n : C^n_{\text{train}}(w_1^n) = r\}} [C_{\text{ho}}(w_1, \ldots, w_n)]$$

- Then using equivalent class of *r* occurrences

$$p_{\text{ho}}(w_1, \ldots, w_n) = \frac{T^n(r) / (C^n(w_1^n))^r}{C(\text{total})} \quad \text{where} \quad C(w_1^n) = r$$

Testing Models

- Divide data into training and testing sets.
- Training data: divide into normal training plus validation (smoothing) sets: around 10% for validation (fewer parameters typically)
- Testing data: distinguish between the “real” test set and a development set.
  - Use a development set prevent successive tweaking of the model to fit the test data
  - ~ 5 – 10% for testing
  - useful to test on multiple sets of test data in order to obtain the variance of results.
  - Are results (good or bad) just the result of chance? Use t-test

Deleted Estimation

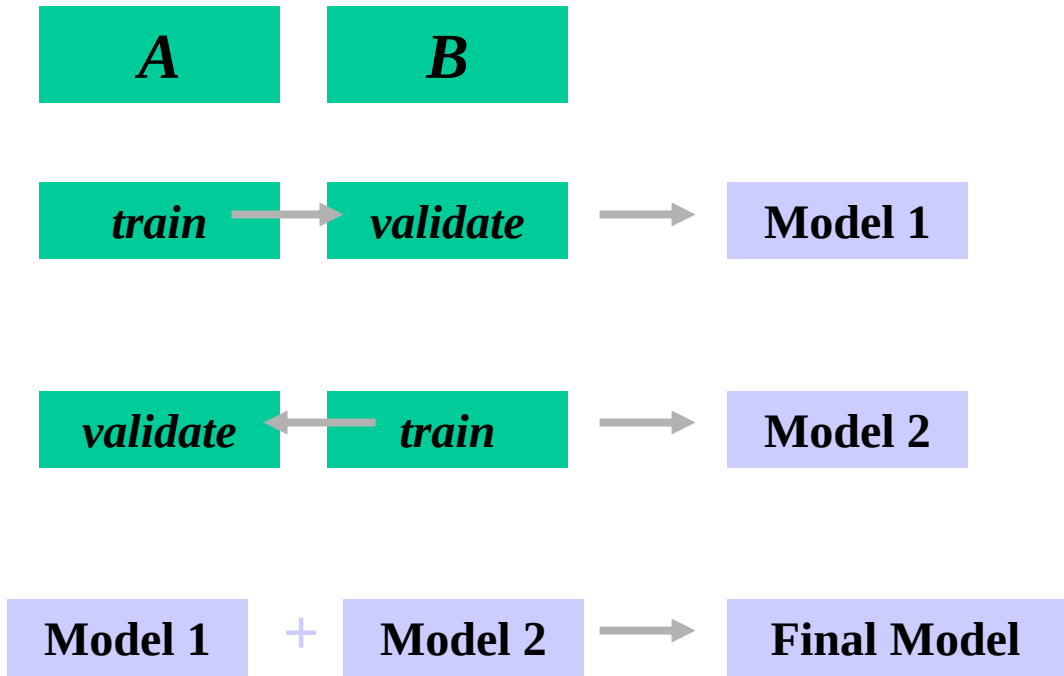
- Use data for both training and validation

Divide training data into 2 parts

(1) Train on *A*, validate on *B*

(2) Train on *B*, validate on *A*

Combine two models



Cross-Validation

Two estimates:

$$P_{ho} = \frac{T_r^{01}}{N_r^0 N} \qquad P_{ho} = \frac{T_r^{10}}{N_r^1 N}$$

N_r^a = number of *n*-grams occurring *r* times in *a*-th part of training set

T_r^{ab} = total number of those found in *b*-th part

Combined estimate:

$$P_{ho} = \frac{T_r^{01} + T_r^{10}}{N(N_r^0 + N_r^1)}$$

(arithmetic mean)

Good-Turing Estimation

- Intuition: re-estimate the amount of mass assigned to *n*-grams with low (or zero) counts using the number of n-grams with higher counts. For any *n*-gram that occurs *r* times, we should assume that it occurs *r** times, where N_r is the number of *n*-grams occurring precisely *r* times in the training data.

$$r^* = (r+1)\frac{N_{r+1}}{N_r}$$

- To convert the count to a probability, we normalize the *n*-gram α with *r* counts as:

$$P_{GT}(\alpha) = r^* / N$$

Good-Turing Estimation

- Note that *N* is equal to the original number of counts in the distribution.

$$N = \sum_{r=0}^{\infty} N_r r^* = \sum_{r=0}^{\infty} N_{r+1}(r+1) = \sum_{r=0}^{\infty} N_r r$$

- Makes the assumption of a binomial distribution, which works well for large amounts of data and a large vocabulary despite the fact that words and n-grams do not have that distribution.

Good-Turing Estimation (Cont.)

- *N*-grams with low counts are often treated as if they had a count of 0.
- In practice *r** is used only for small counts; counts greater than *k*=5 are assumed to be reliable: *r**=*r* if *r*> *k*; otherwise:

$$r^* = \frac{\frac{(r+1)N_{r+1}}{rN_r} - \frac{(k+1)N_{k+1}}{N_1}}{1 - \frac{(k+1)N_{k+1}}{N_1}}, \text{for } 1 \le r \le k$$

Good-Turing Estimation (Cont.)

- Based on count equivalent class as r. v.
- Define an adjusted count (another r. v.)

$$r^* = (r+1)\frac{E(C(X)=r+1)}{E(C(X)=r)}$$

- Good-Turning Estimator

$$p_{\mathrm{GT}}(w_1^n) = \frac{r^*}{C(\mathrm{total})} = \frac{(r+1)}{C(\mathrm{total})}\frac{S(r+1)}{S(r)} \qquad C(w_1^n) = r > 0$$

$$p_{\mathrm{GT}}(w_1^n) \approx \frac{C(r=1)}{C(\mathrm{total}) * C(r=0)}$$

- *S(r)* is some estimator of the expectation
- All new counts try to improve estimation in the case of sparse training data set

Discounting Methods

- *Absolute discounting:* Decrease probability of each observed *n*-gram by subtracting a small constant when $C(w_1, w_2, \ldots, w_n) = r$:

$$p_{abs}(w_1, w_2, \ldots, w_n) = \begin{cases} (r - \partial) / N, \text{ if } r > 0 \\ \dfrac{(B - N_0)\partial}{N_0 N}, \text{ otherwise} \end{cases}$$

- *Linear discounting:* Decrease probability of each observed *n*-gram by multiplying by the same proportion when $C(w_1, w_2, \ldots, w_n) = r$:

$$p_{lin}(w_1, w_2, \ldots, w_n) = \begin{cases} (1-\alpha) r / N, \text{ if } r > 0 \\ \dfrac{\alpha}{N_0}, \text{ otherwise} \end{cases}$$

Combining Estimators: Overview

- If we have several models of how the history predicts what comes next, then we might wish to combine them in the hope of producing an even better model.
- Some combination methods:
  - Katz's Back Off
  - Simple Linear Interpolation
  - General Linear Interpolation

Combining Estimators

- Combination over same or different corpora
- Linear interpolation of trigrams

$$\tilde{p}(w_n \mid w_{n-1}, w_{n-2}) = \lambda_1 p_1(w_n) + \lambda_2 p_2(w_n \mid w_{n-1}) + \lambda_3 p_3(w_n \mid w_{n-1}, w_{n-2})$$

where $0 \le \lambda_i \le 1$ and $\lambda_1 + \lambda_2 + \lambda_3 = 1$

- More extended Linear Interpolation:

$$\tilde{p}(w \mid h) = \sum_{i=1}^{H} \lambda_i p_i(w \mid h) \quad h - \text{history}$$

where $0 \le \lambda_i \le 1$ and $\sum_{i=1}^{H} \lambda_i = 1$

Backoff

- Back off to lower order *n*-gram if we have no evidence for the higher order form. Trigram backoff:

$$P_{bo}(w_i \mid w_{i-2}^{i-1}) = \begin{cases} P(w_i \mid w_{i-2}^{i-1}), \text{if } C(w_{i-2}^{i}) > 0 \\ \alpha_1 P(w_i \mid w_{i-1}), \text{if } C(w_{i-2}^{i}) = 0 \text{ and } C(w_{i-1}^{i}) > 0 \\ \alpha_2 P(w_i), \text{otherwise} \end{cases}$$

Katz's Back Off Model

- If the *n*-gram of concern has appeared more than *k* times, then an *n*-gram estimate is used but an amount of the MLE estimate gets discounted (it is reserved for unseen n-grams).
- If the *n*-gram occurred *k* times or less, then we will use an estimate from a shorter n-gram (back-off probability), normalized by the amount of probability remaining and the amount of data covered by this estimate.
- The process continues recursively.

Katz's Back Off Model (Cont.)

- Katz used Good-Turing estimates when an *n*-gram appeared *k* or fewer times.

$$P_{bo}(w_i \mid w_{i-n+1}^{i-1}) = \begin{cases} (1 - d_{w_{i-n+1}^{i-1}}) \dfrac{\mathrm{C}(w_{i-n+1}^{i})}{\mathrm{C}(w_{i-n+1}^{i-1})}, & \text{if } C(w_{i-n+1}^{i}) > k \\ \alpha_{w_{i-n+1}^{i-1}} P_{bo}(w_i \mid w_{i-n+2}^{i-1}), & \text{otherwise} \end{cases}$$

Problems with Backing-Off

- If bigram ($w_1\ w_2$) is common, but trigram ($w_1\ w_2\ w_3$) is unseen, it may be a *meaningful* gap, rather than a gap due to chance and scarce data
  - i.e., a "grammatical null"
- In that case, it may be inappropriate to back-off to lower-order probability

Linear Interpolation

- One way of solving the sparseness in a trigram model is to mix that model with bigram and unigram models that suffer less from data sparseness.
- This can be done by ***linear interpolation*** (also called ***finite mixture models***). When the functions being interpolated all use a subset of the conditioning information, this method is referred to as ***deleted interpolation***.
- The weights can be set using the Expectation-Maximization (EM) algorithm.

$$P_{li}(w_i \mid w_{i-2}, w_{i-1}) =$$

$$\lambda_1 P_1(w_i) + \lambda_2 P_2(w_i \mid w_{i-1}) + \lambda_3 P_3(w_i \mid w_{i-2}, w_{i-1})$$

General Linear Interpolation

$$P_{li}(w_i \mid w_{i-n+1}^{i-1}) = \sum_{k=1}^{n} \lambda(w_{i-k+1}^{i-1}) P_k(w_i \mid w_{i-k+1}^{i-1})$$

$$\text{where } 0 \le \lambda(w_{i-k+1}^{i-1}) \le 1, \text{ and } \sum_k \lambda(w_{i-k+1}^{i-1}) = 1$$

- In simple linear interpolation, the weights are just a single number, but one can define a more general and powerful model where the weights are a function of the history
- Need some way to group or bucket lambda histories

Deleted Interpolation Estimation

- Deleted interpolation estimator

$$p_{\text{del}}(w_1^n) = \mu_1 p_{\text{ho}}^{12}(w_1^n) + (1-\mu_1) p_{\text{ho}}^{21}(w_1^n) \quad \text{or}$$

$$p_{\text{del}}(w_1^n) = [(T^n(r))^{12} + (T^n(r))^{21}] \Big/ \{C(\text{total})[(C^n(w_1^n)^r)^1 + (C^n(w_1^n)^r)^2]\}$$

- Leaving-one-out (Jackknife example)
  - held-out one sample at a time (many splits)
  - average over all estimates to reduce variance (done often is estimating spectral densities)

Katz's Backing-Off Estimators

$$p_{\text{bo}}(w_i \mid w_{i-n+1}^{i-1}) = \begin{cases} (1-\alpha_{i-n+1}^{i-1}) * C(w_{i-n+1}^{i}) / C(w_{i-n+1}^{i-1}) & C(w_{i-n+1}^{i-1}) > k \\ \alpha_{i-n+1}^{i-1} p_{\text{bo}}(w_i \mid w_{i-n+1}^{i-1}) & \text{otherwise} \end{cases}$$

$$p_{\text{bo}}(w_3 \mid w_1^2) = \begin{cases} (1-\alpha_1^2) * C(w_1^3) / C(w_1^2) & C(w_1^2) > k \ (k = 0 \text{ or } 1) \\ \alpha_1^2 p_{\text{bo}}(w_3 \mid w_1^2) & \text{otherwise} \end{cases}$$

- *Arguably simple, but quite effective*
- *N*-gram is discounted by some amount so that some reserved counts can be used for unseen ones whose probabilities are estimated by back-off, e.g. unseen trigrams estimated by bigrams
- Discount can be done with Good-Turing estimator

Summary

- Today's Class
  - *N*-gram estimation on Feb 3 and Feb. 5
  - Lab2 assigned on Jan. 29, due on Feb. 10
- Next Class
  - Project discussion
  - Word Sense Disambiguation
- Reading Assignments
  - Manning and Schutze, Chapter 6