

n-gram models

Computational Linguistics

Alexander Koller

30 October 2018

Let's play a game

- I will write a sentence on the board.
- Each of you, in turn, gives me a word to continue that sentence, and I will write it down.




Let's play another game


- You write a word on a piece of paper.
- You get to see the piece of paper of your neighbor, but none of the earlier words.
- In the end, I will read the sentence you wrote.

Statistical models in NLP

- *Generative statistical model* of language:
pd $P(w)$ over NL expressions that we can observe.
 - ▶ w may be complete sentences or smaller units
 - ▶ will later extend this to pd $P(w, t)$ with hidden random variables t
- Assumption: A *corpus* of observed sentences w is generated by repeatedly sampling from $P(w)$.
- We try to estimate the *parameters* of the prob dist from the corpus, so we can make *predictions* about unseen data.

Predictive text models

BOB DYLAN   [BOB DYLAN NEWSLETTER](#) [SEARCH & LYRICS](#) 

[TOUR](#) [NEWS](#) [ALBUMS](#) [SONGS](#) [BOOKS](#) [ART](#) [WHISKEY](#) [STORE](#) 


Bob On Bob


WRITTEN BY: BOB DYLAN, AMAZON REVIEWS OF BOB DYLAN

The album is another instant classic from a man
Who knows how to be wandering the road
I bought it for my darling but she said it was not for her
Now i know that dylan magic just cant go on blonde
She said his voice is simply horrible
It sounds like a south-bound train
It sounds like every sparrow is blowing through his face

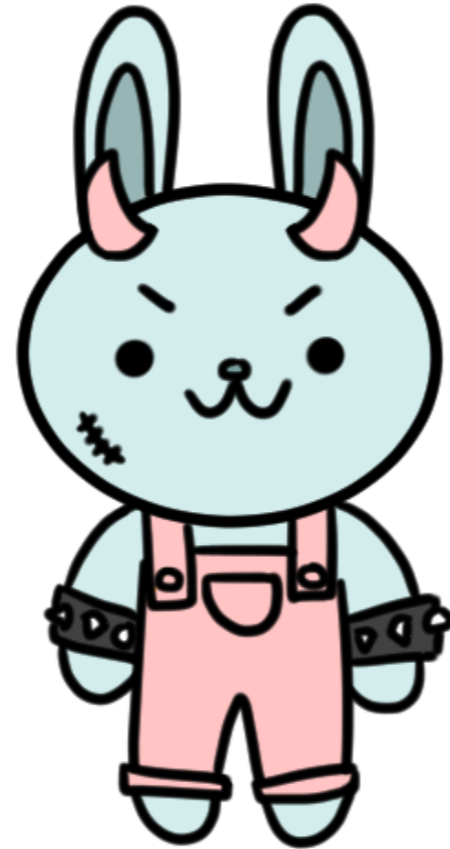
Hey mama is it hopeless?
Is no exception made?
Are dylan's millions of songs all written in vain?

APPEARS ON



Blonde On Blonde
(Original Release)  [BUY](#)

Predictive text models



Aggressive Daniel

Aggressive Daniel is a male rabbit who gives in to peer pressure. His trademark is horns. In 2007, Sanrio finally gave him a mouth. His birthday is wrong.

Word-by-word random process

- A *language model (LM)* is a probability distribution $P(w)$ over sentences.
- Think of it as random process that generates sentences word by word:

X_1

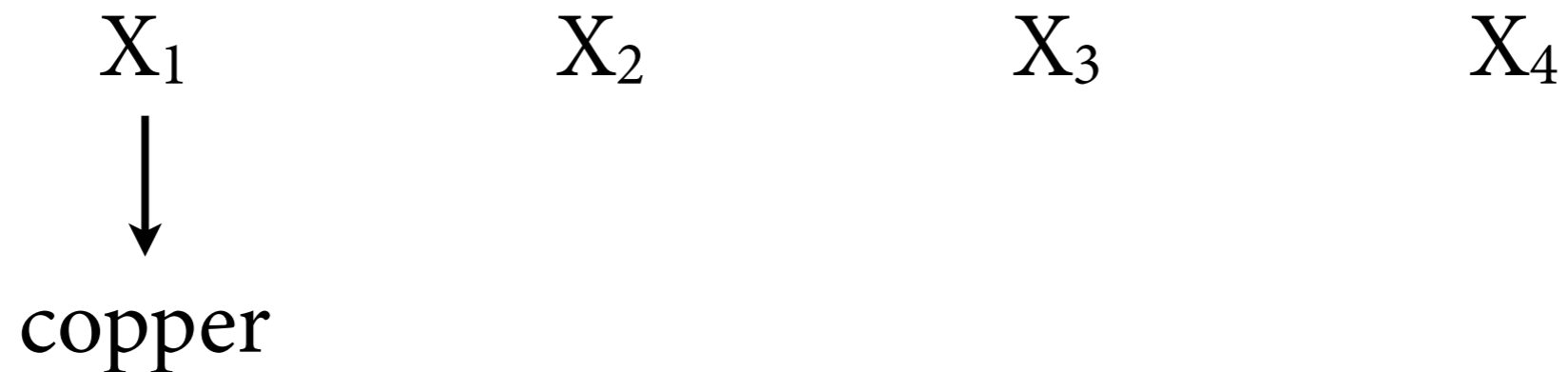
X_2

X_3

X_4

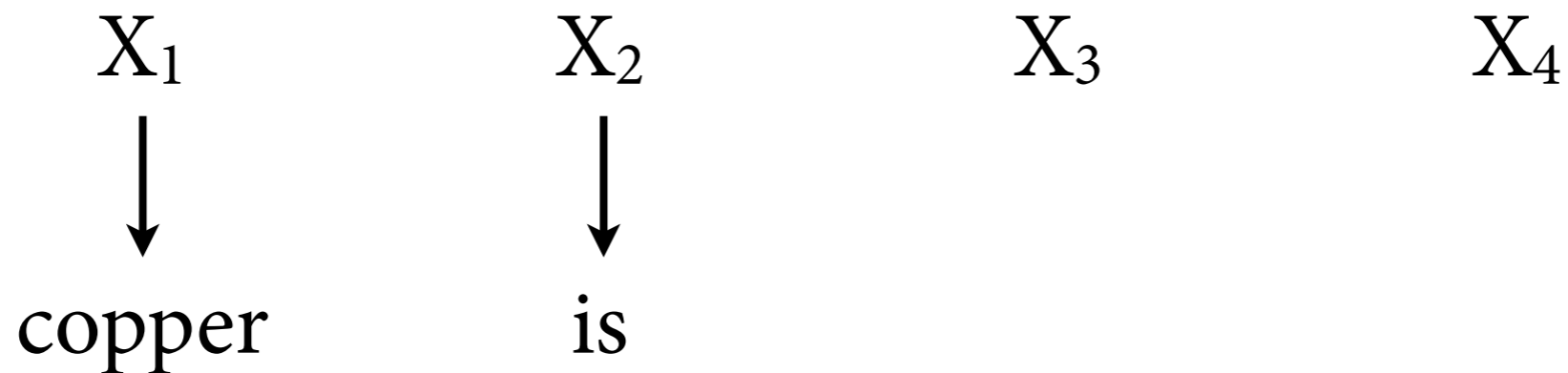
Word-by-word random process

- A *language model (LM)* is a probability distribution $P(w)$ over sentences.
- Think of it as random process that generates sentences word by word:



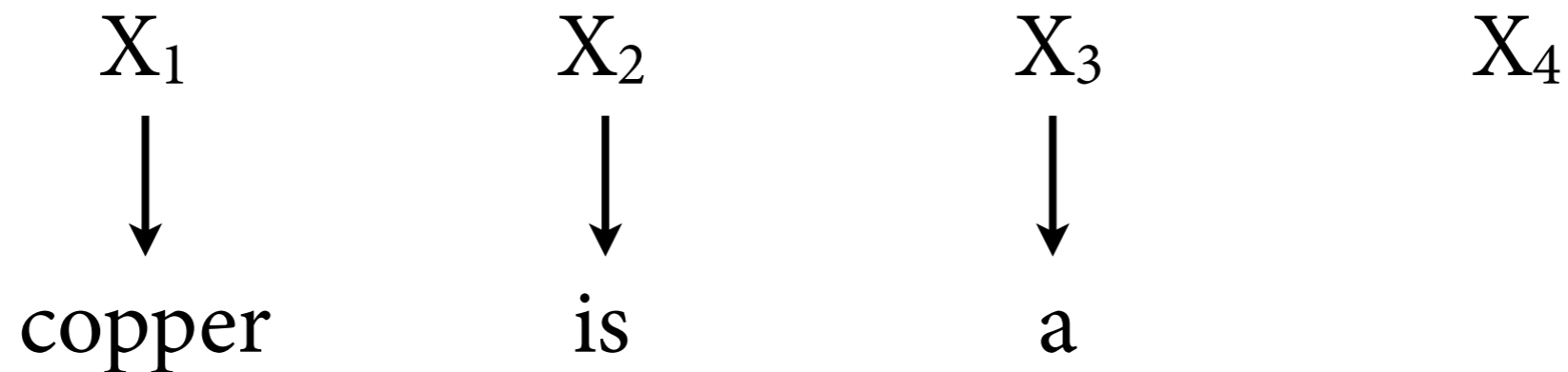
Word-by-word random process

- A *language model (LM)* is a probability distribution $P(w)$ over sentences.
- Think of it as random process that generates sentences word by word:



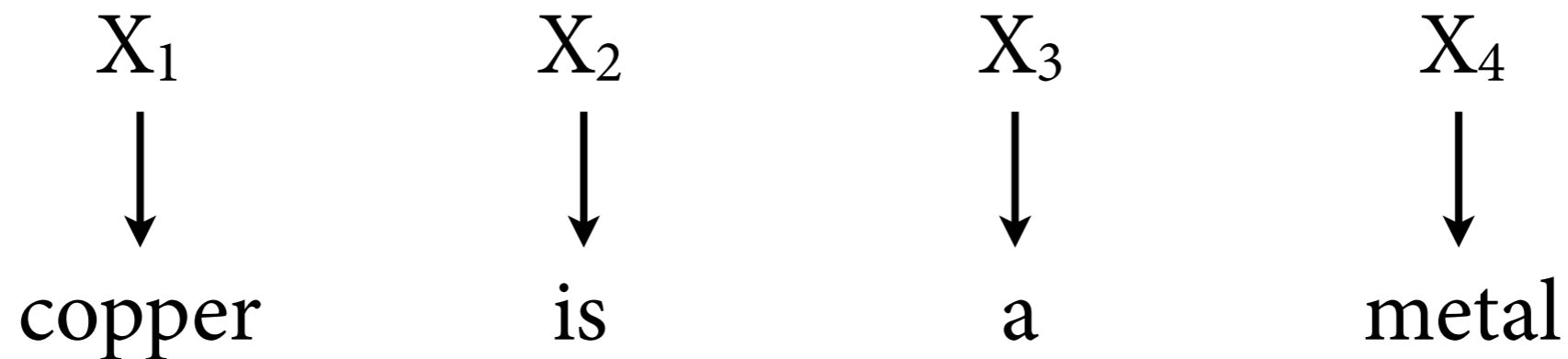
Word-by-word random process

- A *language model (LM)* is a probability distribution $P(w)$ over sentences.
- Think of it as random process that generates sentences word by word:



Word-by-word random process

- A *language model (LM)* is a probability distribution $P(w)$ over sentences.
- Think of it as random process that generates sentences word by word:



Process from our game

- Each of you = a random variable X_t ;
event “ $X_t = w_t$ ” means word at position t is w_t .
- When you chose w_t , you could see the outcomes of the previous variables: $X_1 = w_1, \dots, X_{t-1} = w_{t-1}$.
- Thus, X_t followed a pd

$$P(X_t = w_t \mid X_1 = w_1, \dots, X_{t-1} = w_{t-1})$$

Process from our game

- Assume that X_t follows some given PD

$$P(X_t = w_t \mid X_1 = w_1, \dots, X_{t-1} = w_{t-1})$$

- Then probability of the entire corpus (or sentence)
 $w = w_1 \dots w_n$ is joint probability

$$P(w_1 \dots w_n) = P(w_1) \cdot P(w_2 \mid w_1) \cdot P(w_3 \mid w_1, w_2) \\ \cdot \dots \cdot P(w_n \mid w_1, \dots, w_{n-1})$$

Process from our game

- Assume that X_t follows some given PD

$$P(X_t = w_t \mid X_1 = w_1, \dots, X_{t-1} = w_{t-1})$$

- Then probability of the entire corpus (or sentence)
 $w = w_1 \dots w_n$ is joint probability

$$P(w_1 \dots w_n) = P(w_1) \cdot P(w_2 \mid w_1) \cdot P(w_3 \mid w_1, w_2) \cdot \dots \cdot P(w_n \mid w_1, \dots, w_{n-1})$$

How do we estimate these?



Statistical models

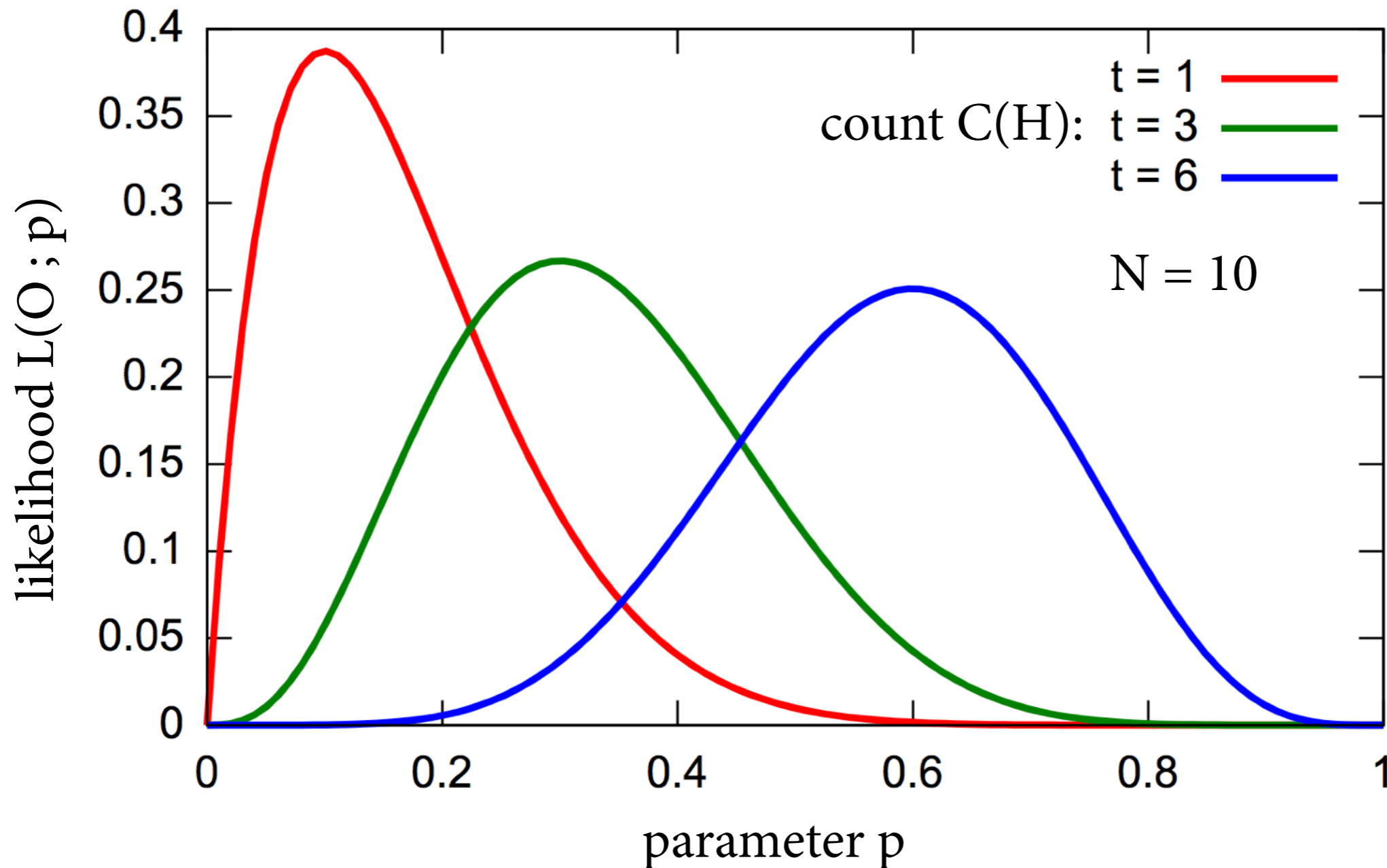
- We want to use prob theory to estimate a *model* of a generating process from *observations* about its outcomes.
- Simpler case: we flip a coin 100 times and observe H 61 times. Should we believe that it is a fair coin?
 - ▶ observation: absolute freq $C(H) = 61$, $C(T) = 39$;
thus relative freq $f(H) = 0.61$, $f(T) = 0.39$
 - ▶ model: assume rv X follows a *Bernoulli* distribution, i.e. X has two outcomes, and there is a value p such that $P(X = H) = p$ and $P(X = T) = 1 - p$.
 - ▶ want to estimate the *parameter* p of this model

Fit of model and observations

- How do we quantify how well a model fits with the observations we made?
- Out of the many possibilities, easiest is to look at the *likelihood*: probability $P(O ; \mathbf{p})$ of the observations O given the values \mathbf{p} for the model parameters.
- *Maximum likelihood estimation*: find parameter values for which the likelihood of O is maximal.

Likelihood functions

$$\text{likelihood } L(O ; p) = p^{C(H)} * (1-p)^{C(T)} * \text{binom}(N, C(H))$$



(Wikipedia page on MLE; licensed from Casp11 under CC BY-SA 3.0)

ML Estimation

- Goal: Find value for p that maximizes the likelihood of the observations.
- For Bernoulli models, it is extremely easy to estimate the parameters that maximize the likelihood:
 - ▶ $P(X = a) = f(a)$
 - ▶ in the coin example above, just take $p = f(H)$
- Can prove that relative frequency is an ML estimator for a lot of different statistical models (Bernoulli, multinomial, etc.; see link on course page).

Parameters of the model

- Our model has one parameter for $P(X_t = w_t \mid w_1, \dots, w_{t-1})$ for each t and w_1, \dots, w_t .

- Can use maximum likelihood estimation:

$$P(w_t \mid w_1, \dots, w_{t-1}) = \frac{C(w_1 \dots w_{t-1} w_t)}{C(w_1 \dots w_{t-1})}$$

- Let's say a natural language has 10^5 different words. How many tuples w_1, \dots, w_t of length t ?

- ▶ $t = 1$: 10^5
- ▶ $t = 2$: 10^{10} different contexts
- ▶ $t = 3$: 10^{15} ; etc.

Sparse data problem

- Typical corpus sizes:
 - ▶ Brown corpus: about 10^6 tokens
 - ▶ Gigaword corpus: about 10^9 tokens
- Problem exacerbated by *Zipf's Law*:
 - ▶ Order all words by their absolute frequency in corpus (rank 1 = most frequent word).
 - ▶ Then $\log(\text{absolute frequency})$ falls linearly with $\log(\text{rank})$; i.e., most words are really rare.
 - ▶ Zipf's Law is very robust across languages and corpora.

Independence assumptions

- Let's pretend that word at position t depends only on the words at positions $t-1, t-2, \dots, t-k$ for some fixed k (*Markov assumption* of degree k).

- Then we get an n -gram model, with $n = k+1$:

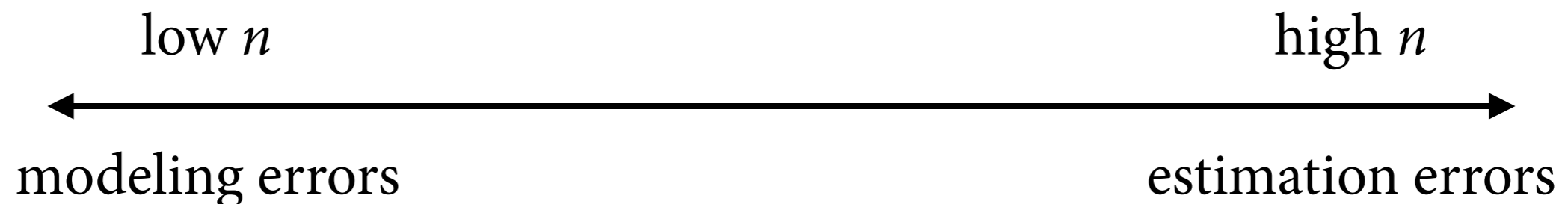
$$P(X_t \mid X_1, \dots, X_{t-1}) = P(X_t \mid X_{t-k}, \dots, X_{t-1})$$

for all t .

- Special names for *unigram models* ($n = 1$), *bigram models* ($n = 2$), *trigram models* ($n = 3$).
 - ▶ Thus our second game was a bigram model.

Independence assumptions

- We assume statistical independence of X_t from events that are too far in the past, although we know that this assumption is incorrect.
- Typical tradeoff in statistical NLP:
 - ▶ if model is too shallow, it won't represent important linguistic dependencies
 - ▶ if model is too complex, its parameters can't be estimated accurately from the available data



Bigrams: an example

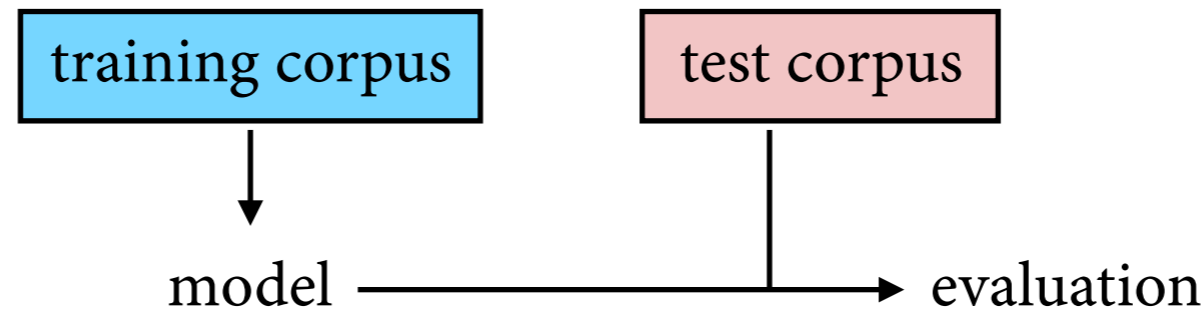
JOHN READ MOBY DICK
MARY READ A DIFFERENT BOOK
SHE READ A BOOK BY CHER

$p(\text{JOHN READ A BOOK})$

$$\begin{aligned} &= p(\text{JOHN}|\bullet) p(\text{READ}|\text{JOHN}) p(\text{A}|\text{READ}) p(\text{BOOK}|\text{A}) p(\bullet|\text{BOOK}) \\ &= \frac{c(\bullet \text{ JOHN})}{\sum_w c(\bullet w)} \frac{c(\text{JOHN READ})}{\sum_w c(\text{JOHN } w)} \frac{c(\text{READ A})}{\sum_w c(\text{READ } w)} \frac{c(\text{A BOOK})}{\sum_w c(\text{A } w)} \frac{c(\text{BOOK } \bullet)}{\sum_w c(\text{BOOK } w)} \\ &= \frac{1}{3} \quad \frac{1}{1} \quad \frac{2}{3} \quad \frac{1}{2} \quad \frac{1}{2} \\ &\approx 0.06 \end{aligned}$$

n-grams: Evaluation

- Measure quality of n-gram model using *perplexity*
 $PP(w) = P(w_1 \dots w_N)^{-1/N}$ of test data $w = w_1 \dots w_N$.
- To get honest picture of model's performance, evaluate it on test data that was not used for training.



- Maximum likelihood model for training corpus is not necessarily good for test corpus (overfitting).

Bigrams: a problem

JOHN READ MOBY DICK
MARY READ A DIFFERENT BOOK
SHE READ A BOOK BY CHER

$$p(\text{CHER READ A BOOK})$$

$$= p(\text{CHER}|\bullet) p(\text{READ}|\text{CHER}) p(\text{A}|\text{READ}) p(\text{BOOK}|\text{A}) p(\bullet|\text{BOOK})$$

$$= \frac{c(\bullet \text{ CHER})}{\sum_w c(\bullet w)} \frac{c(\text{CHER READ})}{\sum_w c(\text{CHER } w)} \frac{c(\text{READ A})}{\sum_w c(\text{READ } w)} \frac{c(\text{A BOOK})}{\sum_w c(\text{A } w)} \frac{c(\text{BOOK } \bullet)}{\sum_w c(\text{BOOK } w)}$$

$$= \frac{0}{3} \quad \frac{0}{1} \quad \frac{2}{3} \quad \frac{1}{2} \quad \frac{1}{2}$$

$$= 0$$

Unseen data

- ML estimate is “optimal” only for the corpus from which we computed it.
- Usually does not generalize directly to new data.
 - ▶ Ok for unigrams, but there are *so many* bigrams.
- ML estimate predicts probability of 0 for n-grams that were not observed in training. This is a disaster because product with 0 is always 0.

Smoothing techniques

- Basic idea: Replace ML estimate

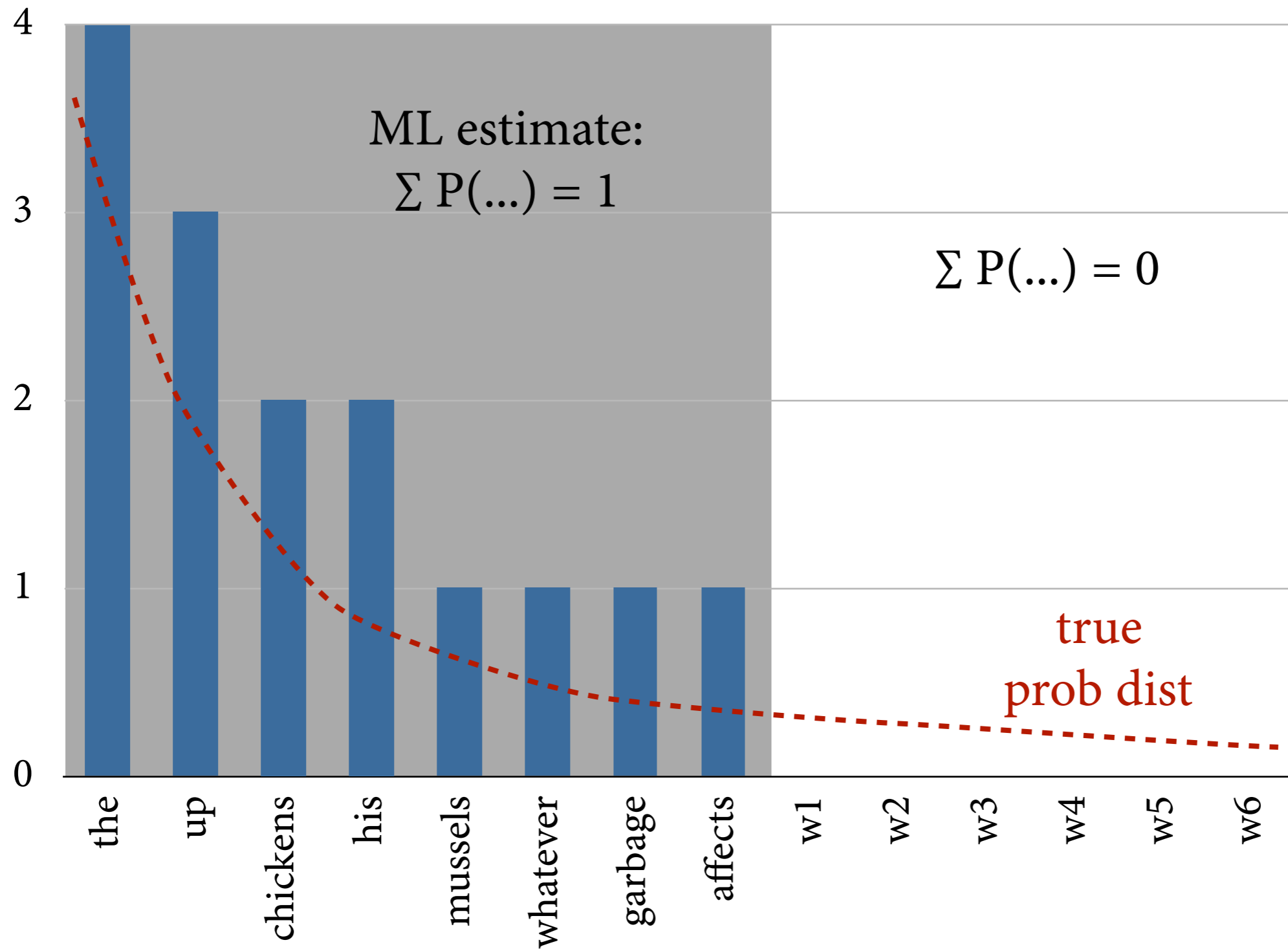
$$P_{\text{ML}}(w_i | w_{i-1}) = \frac{C(w_{i-1}w_i)}{C(w_{i-1})}$$

by estimate with adjusted bigram count

$$P^*(w_i | w_{i-1}) = \frac{C^*(w_{i-1}w_i)}{C(w_{i-1})}$$

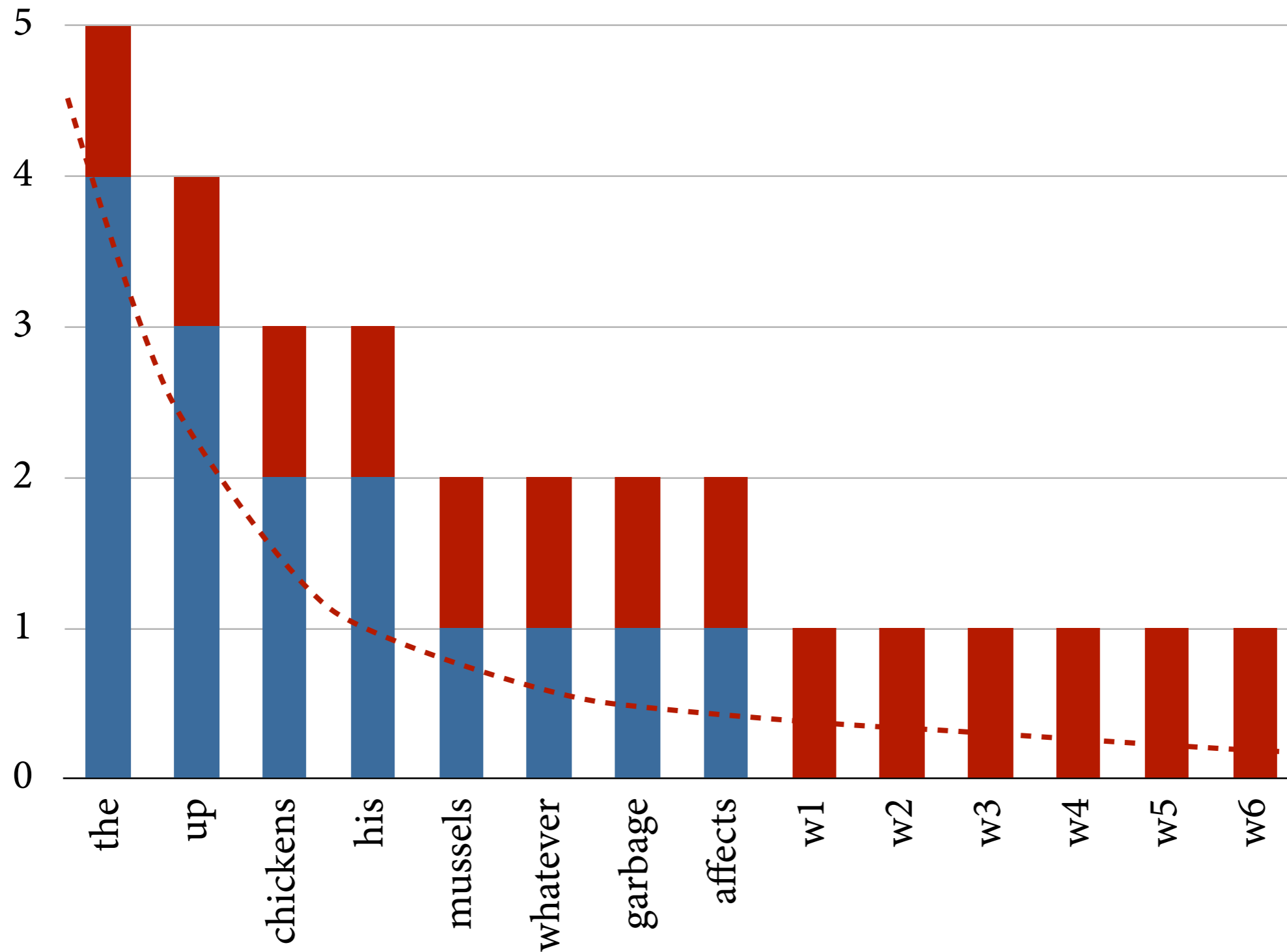
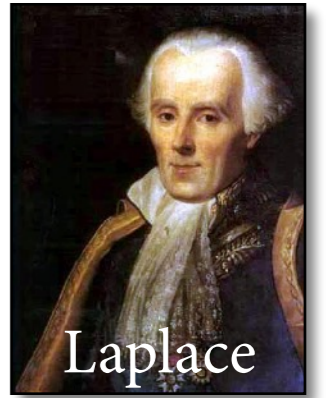
- Redistribute counts from seen to unseen bigrams.
- Generalizes easily to n-gram models with $n > 2$.

Smoothing

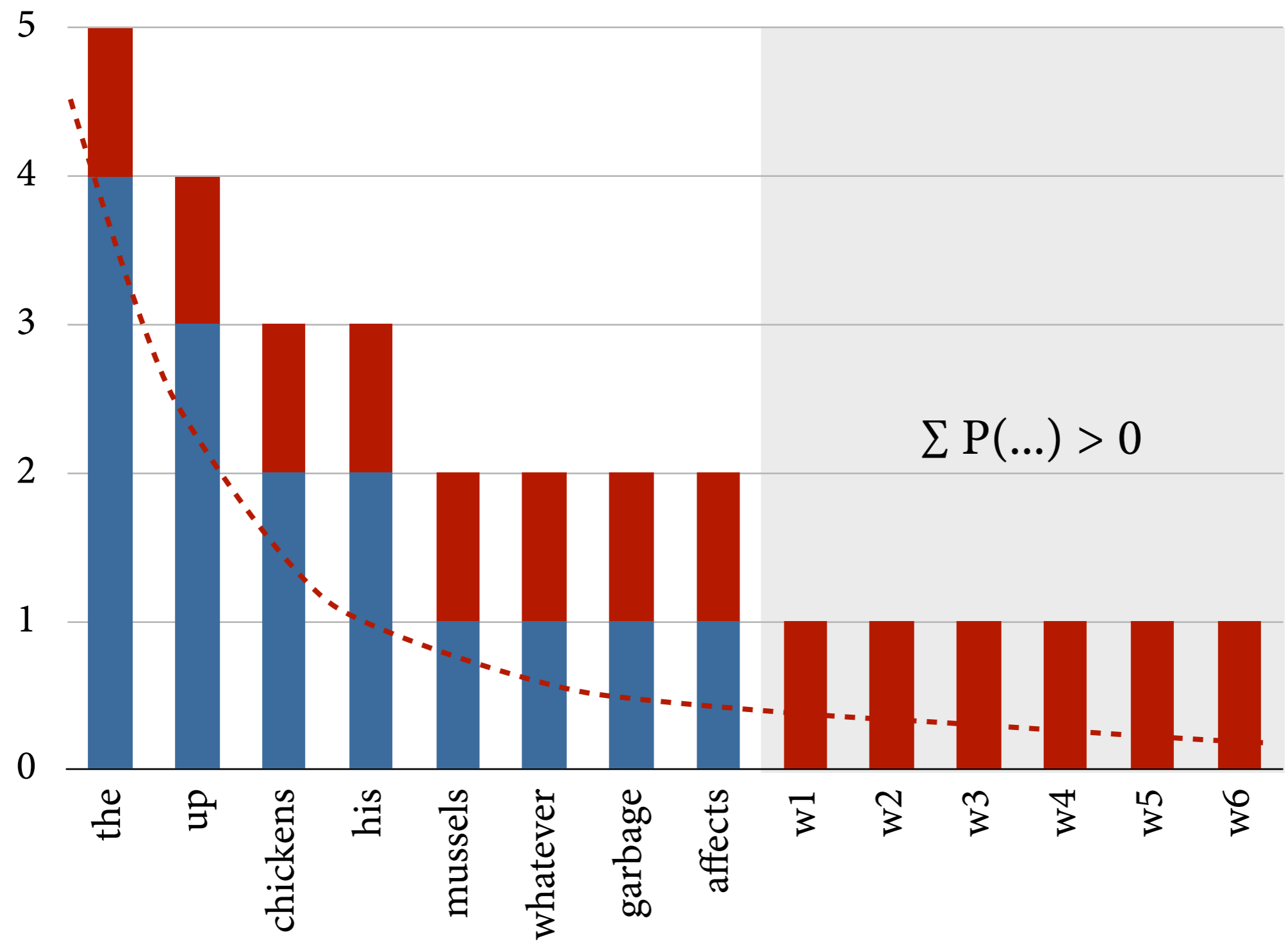
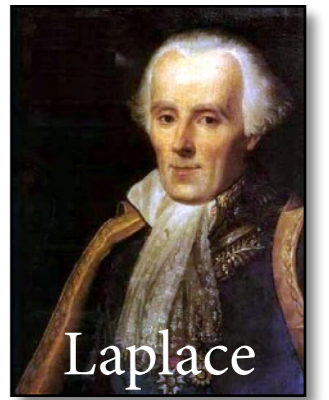


C(eat X) in Brown corpus

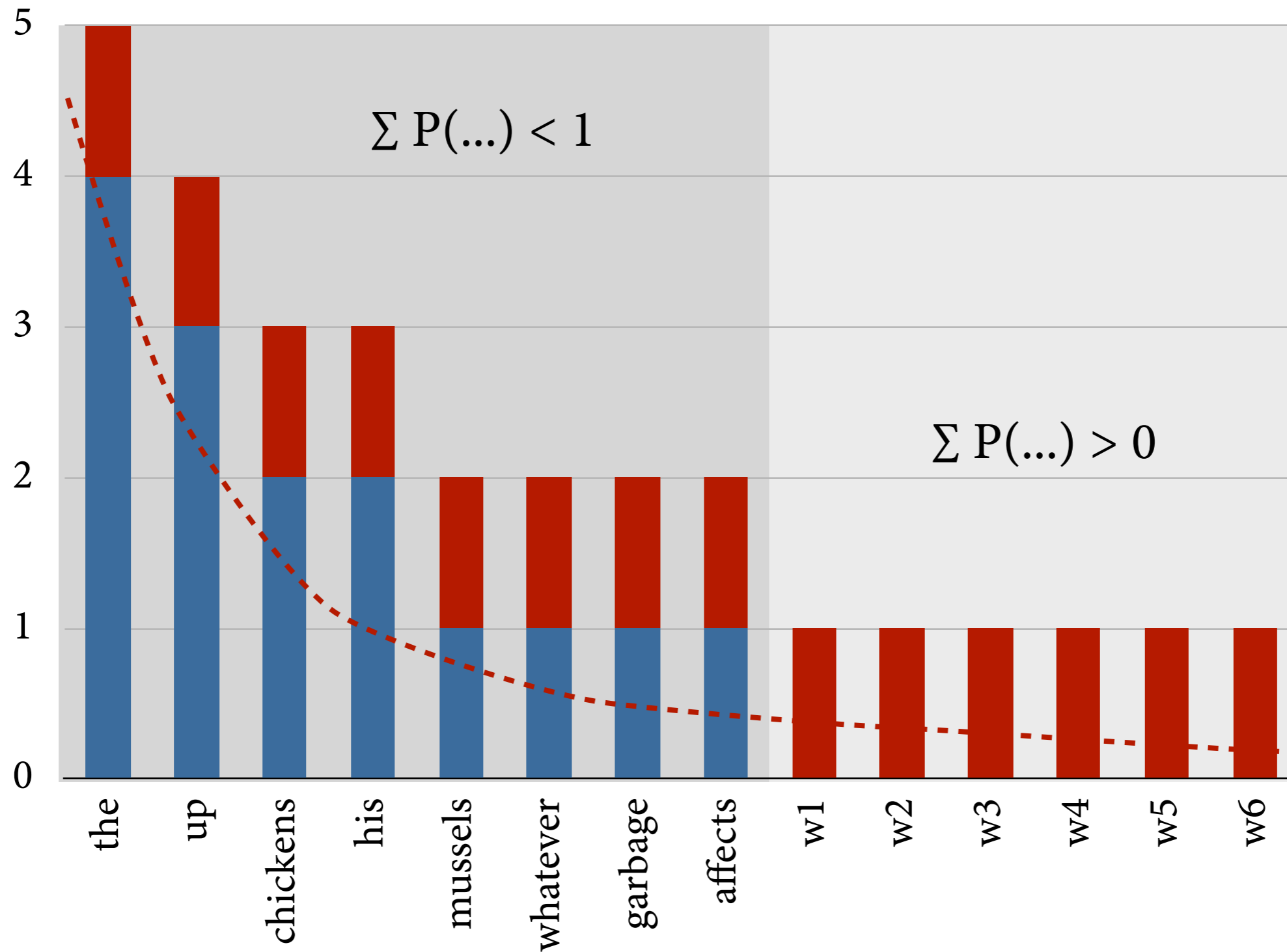
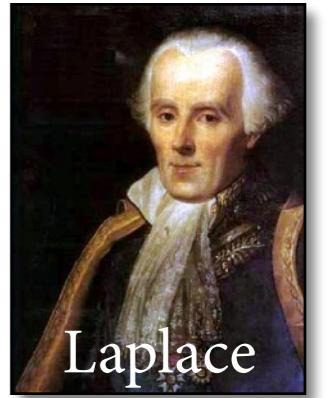
Add-one Smoothing



Add-one Smoothing



Add-one Smoothing



Add-one Smoothing

- Count every bigram (seen or unseen) one more time than in corpus and normalize:

$$P_{\text{lap}}(w_i | w_{i-1}) = \frac{C(w_{i-1}w_i) + 1}{\sum_w (C(w_{i-1}w) + 1)} = \frac{C(w_{i-1}w_i) + 1}{C(w_{i-1}) + |V|}$$

JOHN READ MOBY DICK
MARY READ A DIFFERENT BOOK
SHE READ A BOOK BY CHER

$|V| = 11$, $|\text{seen bigram types}| = 11$
 $\Rightarrow 110$ unseen bigrams

$p(\text{JOHN READ A BOOK})$

$$= \frac{1+1}{11+3} \frac{1+1}{11+1} \frac{1+2}{11+3} \frac{1+1}{11+2} \frac{1+1}{11+2}$$

$$\approx 0.0001$$

$p(\text{CHER READ A BOOK})$

$$= \frac{1+0}{11+3} \frac{1+0}{11+1} \frac{1+2}{11+3} \frac{1+1}{11+2} \frac{1+1}{11+2}$$

$$\approx 0.00003$$

Add-one Smoothing

- Easy to implement, but dramatically overestimates probability of unseen events.
 - ▶ In the Cher example: $P_{\text{lap}}(\text{unseen} \mid w_{i-1}) \geq 1/14$;
thus “count”(w_{i-1} unseen) $\approx 110 * 1/14 = 7.8$.
 - ▶ Compare against 12 bigram tokens in training corpus.
- Learn all about how to *really* do smoothing in the course “Statistical NLP”.

Conclusion

- Statistical models of natural language.
- Language models with n-grams.
- The problem of data sparseness.
- Smoothing.