

Language Modeling: Part 2

Many slides are adapted from Jurafsky and Manning's online lectures, some are adapted from online lectures by Michael Collins

The perils of overfitting

- N-grams only work well for word prediction if the test corpus looks like the training corpus
 - In real life, it often doesn't
 - We need to train robust models that generalize!
 - One kind of generalization: Zeros!
 - Things that don't ever occur in the training set
 - But occur in the test set

Zeros

Training set:

- ... denied the allegations
- ... denied the reports
- ... denied the claims
- ... denied the request

$$P(\text{"offer"} \mid \text{denied the}) = 0$$

Test set

- ... denied the offer
- ... denied the loan

Zero probability bigrams

- Bigrams with zero probability
 - mean that we will assign 0 probability to the test set!
- And hence we cannot compute perplexity (can't divide by 0)!

The intuition of smoothing (from Dan Klein)

- When we have sparse statistics:

$P(w \mid \text{denied the})$

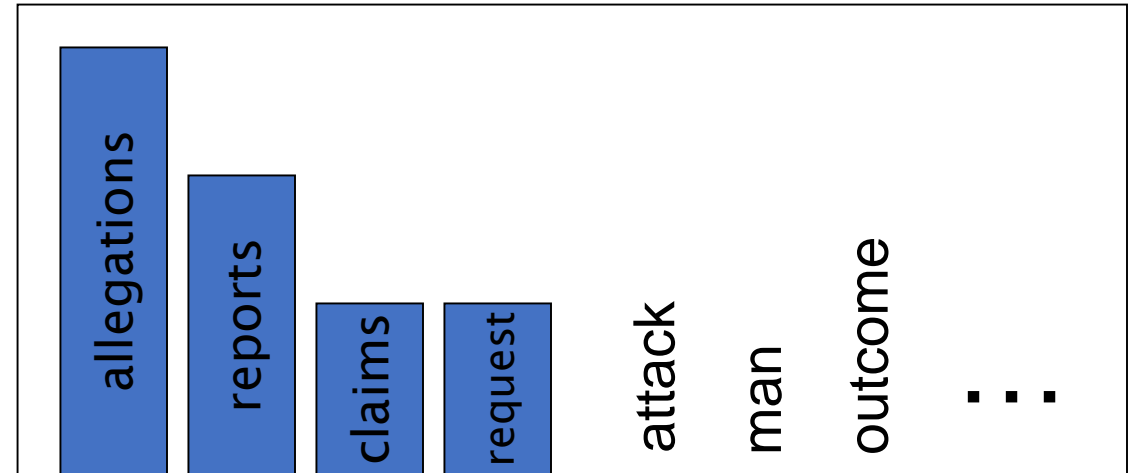
3 allegations

2 reports

1 claims

1 request

7 total



- Steal probability mass to generalize better:

$P(w \mid \text{denied the})$

2.5 allegations

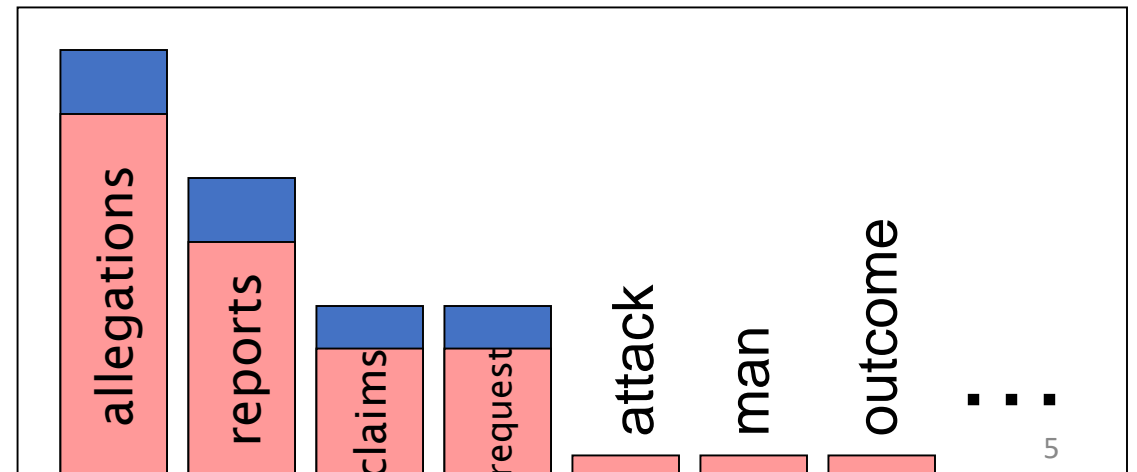
1.5 reports

0.5 claims

0.5 request

2 other

7 total



Add-one estimation

- Also called Laplace smoothing
- Pretend we saw each word one more time than we did
- Just add one to all the counts!

- MLE estimate:
$$P_{MLE}(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

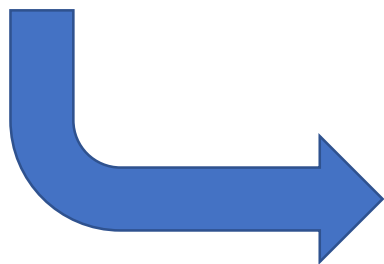
- Add-1 estimate:
$$P_{Add-1}(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i) + 1}{c(w_{i-1}) + V}$$

Maximum Likelihood Estimates

- The maximum likelihood estimate
 - of some parameter of a model M from a training set T
 - maximizes the likelihood of the training set T given the model M
- Suppose the word “bagel” occurs 400 times in a corpus of a million words
- What is the probability that a random word from some other text will be “bagel”?
- MLE estimate is $400/1,000,000$
- This may be a bad estimate for some other corpus
 - But it is the **estimate** that makes it **most likely** that “bagel” will occur 400 times in a million word corpus.

Berkeley Restaurant Corpus: Laplace smoothed bigram counts

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0



	i	want	to	eat	chinese	food	lunch	spend
i	6	828	1	10	1	1	1	3
want	3	1	609	2	7	7	6	2
to	3	1	5	687	3	1	7	212
eat	1	1	3	1	17	3	43	1
chinese	2	1	1	1	1	83	2	1
food	16	1	16	1	2	5	1	1
lunch	3	1	1	1	1	2	1	1
spend	2	1	2	1	1	1	1	1

Laplace-smoothed bigrams

$$P^*(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n) + 1}{C(w_{n-1}) + V}$$

	i	want	to	eat	chinese	food	lunch	spend
i	6	828	1	10	1	1	1	3
want	3	1	609	2	7	7	6	2
to	3	1	5	687	3	1	7	212
eat	1	1	3	1	17	3	43	1
chinese	2	1	1	1	1	83	2	1
food	16	1	16	1	2	5	1	1
lunch	3	1	1	1	1	2	1	1
spend	2	1	2	1	1	1	1	1



	i	want	to	eat	chinese	food	lunch	spend
i	0.0015	0.21	0.00025	0.0025	0.00025	0.00025	0.00025	0.00075
want	0.0013	0.00042	0.26	0.00084	0.0029	0.0029	0.0025	0.00084
to	0.00078	0.00026	0.0013	0.18	0.00078	0.00026	0.0018	0.055
eat	0.00046	0.00046	0.0014	0.00046	0.0078	0.0014	0.02	0.00046
chinese	0.0012	0.00062	0.00062	0.00062	0.00062	0.052	0.0012	0.00062
food	0.0063	0.00039	0.0063	0.00039	0.00079	0.002	0.00039	0.00039
lunch	0.0017	0.00056	0.00056	0.00056	0.00056	0.0011	0.00056	0.00056
spend	0.0012	0.00058	0.0012	0.00058	0.00058	0.00058	0.00058	0.00058

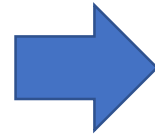
Reconstituted counts

$$\frac{c^*(w_{n-1}w_n)}{C(w_{n-1})} = \frac{[C(w_{n-1}w_n) + 1]}{C(w_{n-1}) + V} \Rightarrow \frac{c^*(w_{n-1}w_n)}{C(w_{n-1})} = \frac{[C(w_{n-1}w_n) + 1] \times C(w_{n-1})}{C(w_{n-1}) + V}$$

	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

Comparison with raw bigram counts

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0



	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

Add-1 estimation is a blunt instrument

- So add-1 isn't used for N-grams:
 - We'll see better methods
- But add-1 is used to smooth other NLP models
 - For text classification
 - In domains where the number of zeros isn't so huge.

Backoff and Interpolation

- Sometimes it helps to use **less** context
 - Condition on less context for contexts you haven't learned much about
- **Backoff:**
 - use trigram if you have good evidence,
 - otherwise bigram, otherwise unigram
- **Interpolation:**
 - mix unigram, bigram, trigram

Linear Interpolation

- Simple interpolation

$$\begin{aligned}\hat{P}(w_n|w_{n-1}w_{n-2}) &= \lambda_1 P(w_n|w_{n-1}w_{n-2}) \\ &\quad + \lambda_2 P(w_n|w_{n-1}) \\ &\quad + \lambda_3 P(w_n)\end{aligned}$$

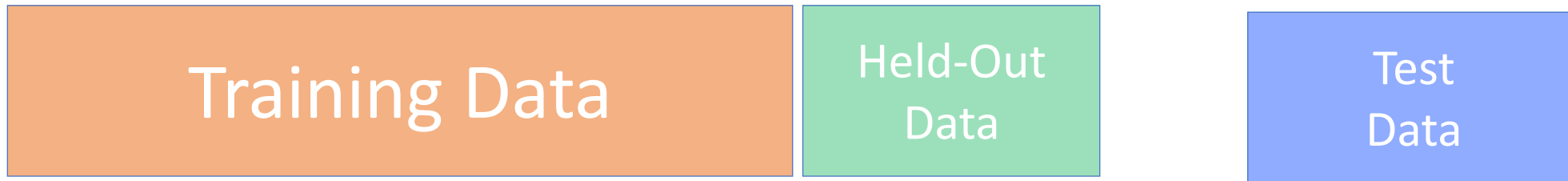
$$\sum_i \lambda_i = 1$$

- Lambdas conditional on context:

$$\begin{aligned}\hat{P}(w_n|w_{n-2}w_{n-1}) &= \lambda_1(w_{n-2}^{n-1}) P(w_n|w_{n-2}w_{n-1}) \\ &\quad + \lambda_2(w_{n-2}^{n-1}) P(w_n|w_{n-1}) \\ &\quad + \lambda_3(w_{n-2}^{n-1}) P(w_n)\end{aligned}$$

How to set the lambdas?

- Use a **held-out** corpus



- Choose λ s to maximize the probability of held-out data:
 - Fix the N-gram probabilities (on the training data)
 - Then search for λ s that give largest probability to held-out set:

$$\log P(w_1 \dots w_n \mid M(I_1 \dots I_k)) = \sum_i \log P_{M(I_1 \dots I_k)}(w_i \mid w_{i-1})$$

Reminder: Add-1 (Laplace) Smoothing

$$P_{Add-1}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i) + 1}{c(w_{i-1}) + V}$$

More general formulations: Add-k

$$P_{Add-k}(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i) + k}{c(w_{i-1}) + kV}$$



Use $m=kV$



$$P_{Add-k}(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i) + m(\frac{1}{V})}{c(w_{i-1}) + m}$$

Unigram prior smoothing

$$P_{Add-k}(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i) + m(\frac{1}{V})}{c(w_{i-1}) + m}$$

$$P_{\text{UnigramPrior}}(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i) + mP(w_i)}{c(w_{i-1}) + m}$$

Advanced smoothing algorithms

- Intuition used by many smoothing algorithms
 - Good-Turing
 - Kneser-Ney
 - Witten-Bell
- Use the count of things we've **seen once**
 - to help estimate the count of things we've **never seen**

N_c : no. of N-grams that occur c times.

MLE count for $N_c = c$

Good Turing Estimate replaces c by c^*

Formula:

$$c^* = (c+1) \frac{N_{c+1}}{N_c}$$

10 carp (c)

3 perch (p)

2 whitefish (w)

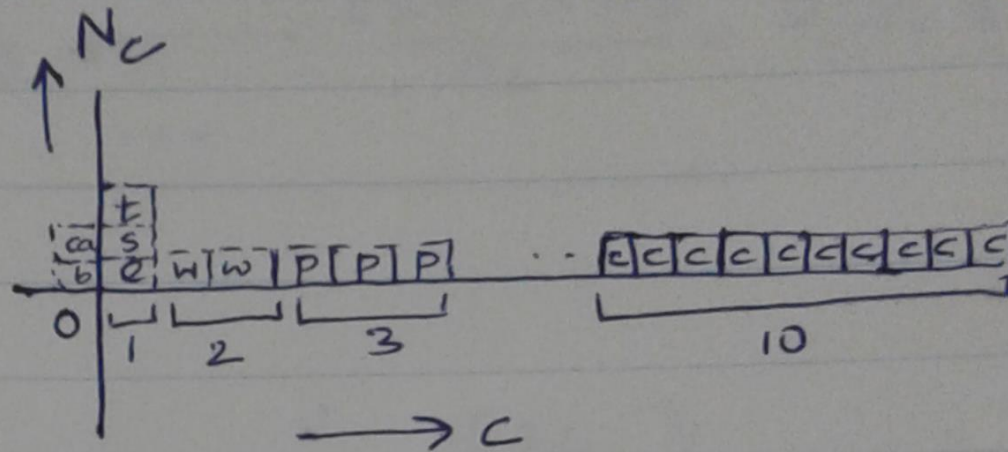
1 trout (t)

1 salmon (s)

1 eel (e)

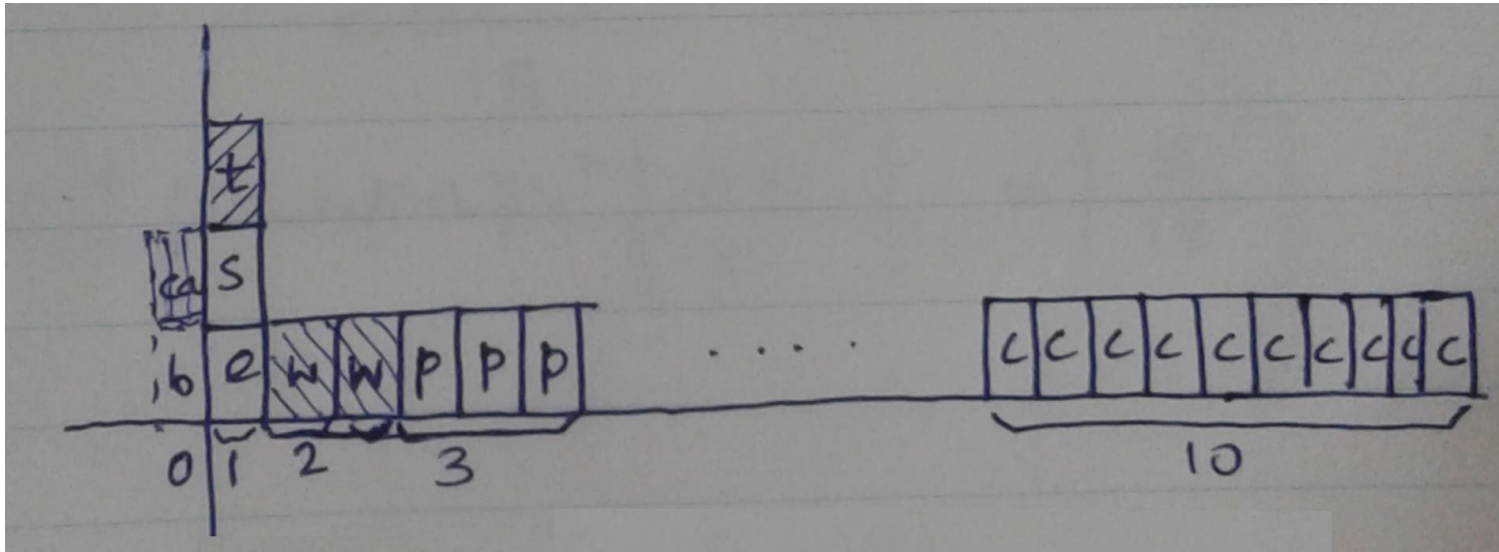
0 catfish (ca)

0 bass (b)



$$N_0 = 2 \quad N_2 = 1 \quad N_4 = 0$$

$$N_1 = 3 \quad N_3 = 1 \quad N_{10} = 1$$

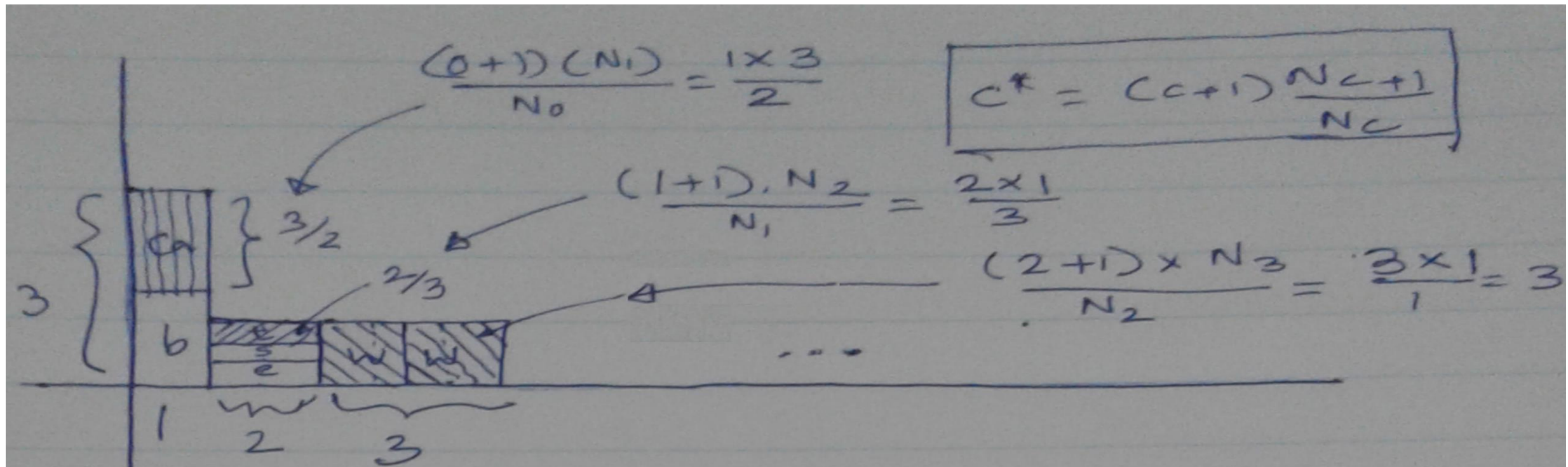
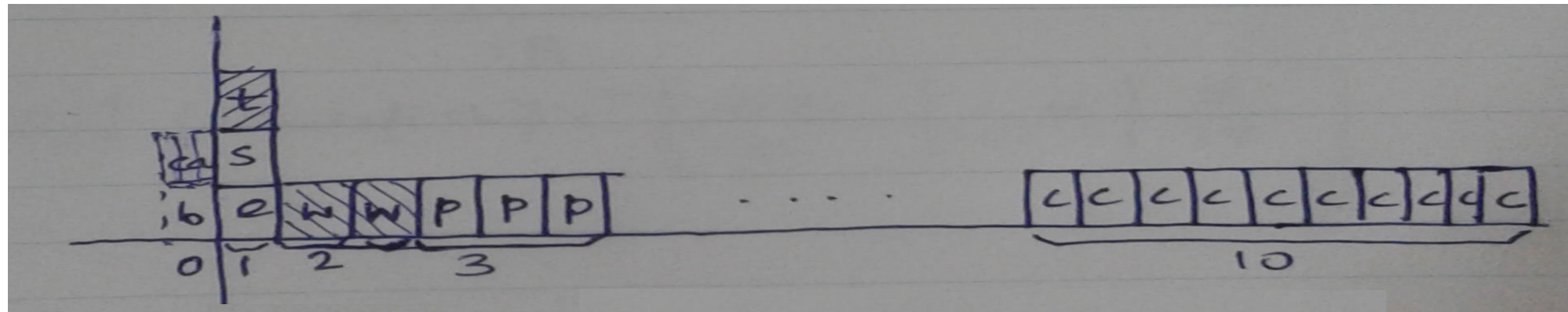


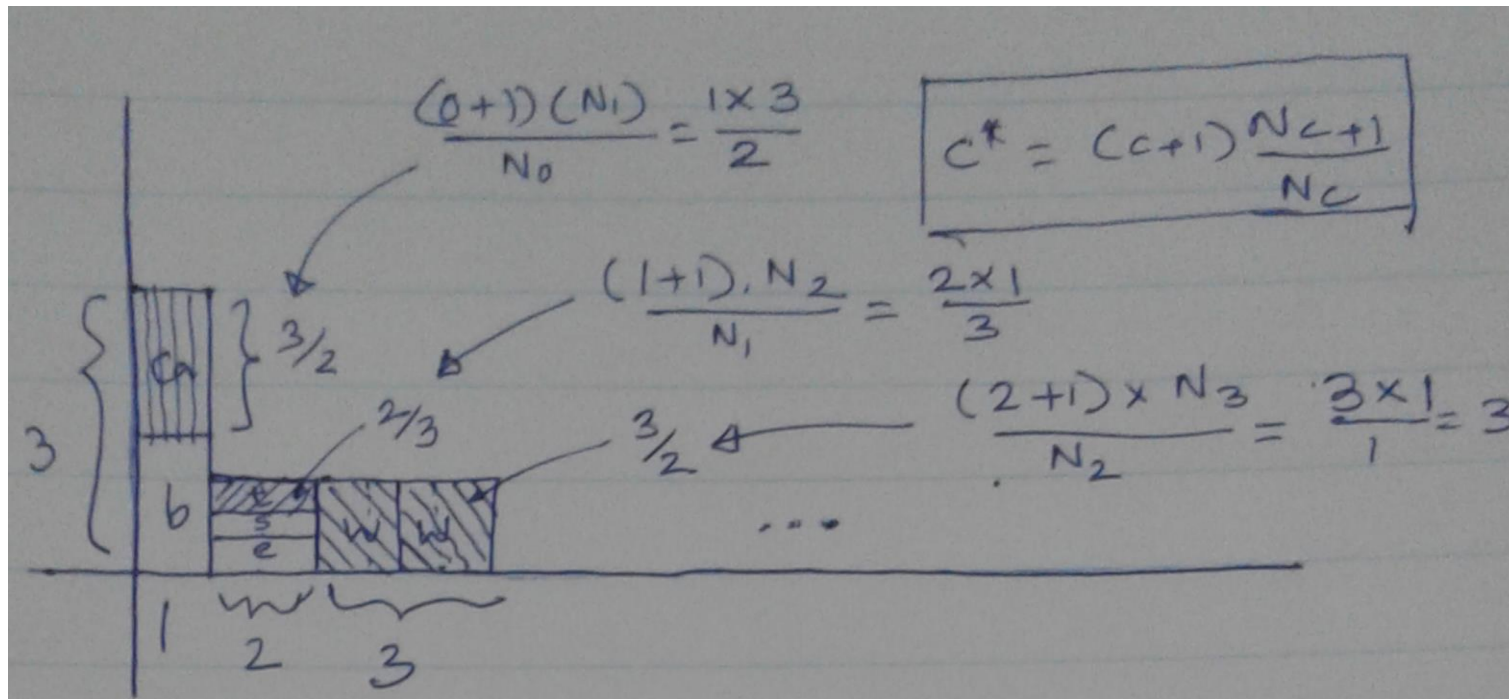
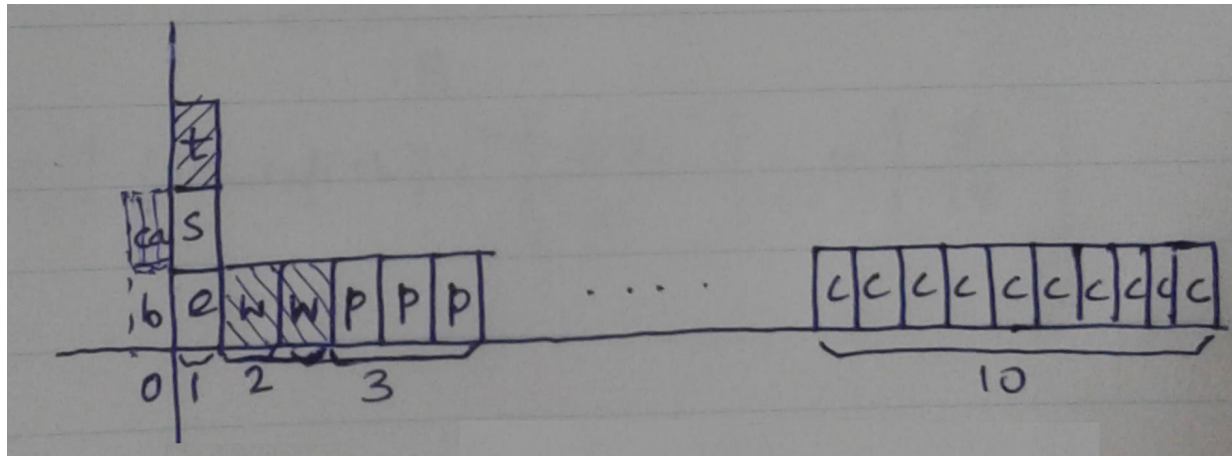
MLE Estimates

$$\begin{array}{|l|} \hline \text{||||} \\ \hline \end{array} P(\text{catfish}) = P(\text{bass}) = 0/18$$


$$\begin{array}{|l|} \hline \text{|||||} \\ \hline \end{array} P(\text{trout}) = P(\text{salmon}) = P(\text{eel}) = 1/18$$


$$\begin{array}{|l|} \hline \text{|||||} \\ \hline \end{array} P(\text{whitefish}) = 2/18$$







MLE Estimates


 $P(\text{catfish}) = P(\text{bass}) = 0/18$


 $P(\text{trout}) = P(\text{salmon}) = P(\text{eel}) = 1/18$

 $P(\text{whitefish}) = 2/18$



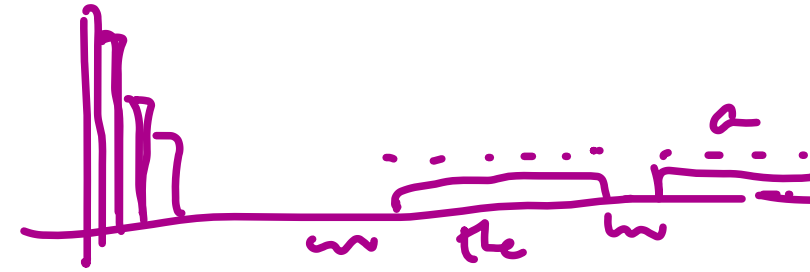
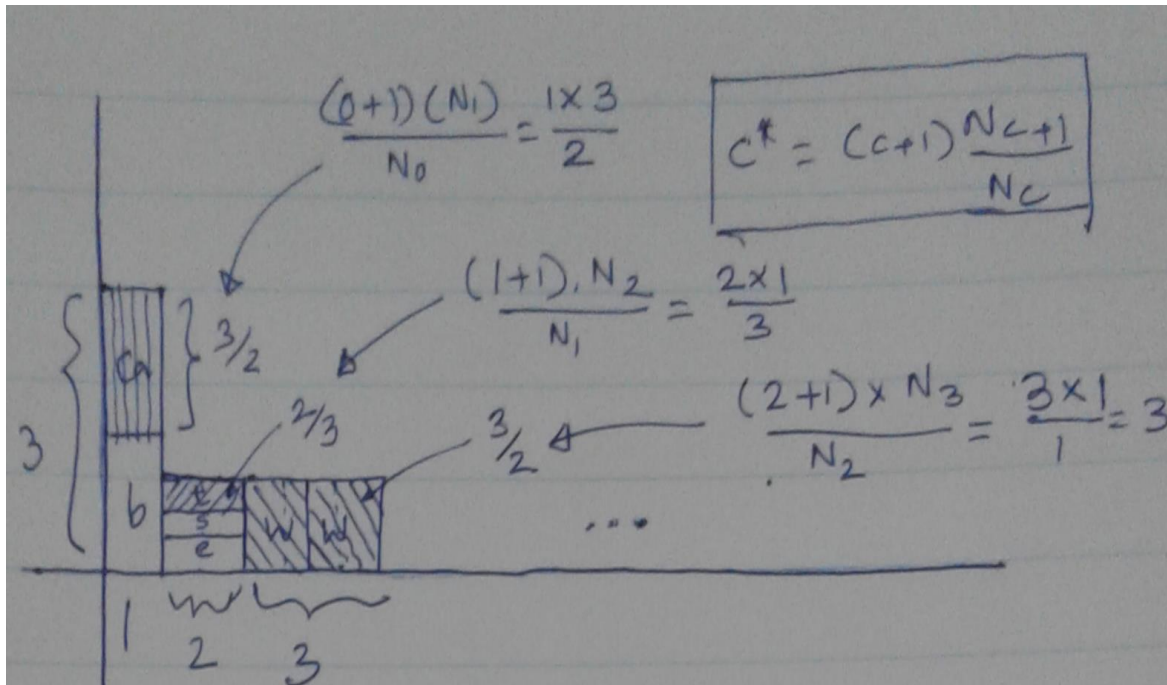
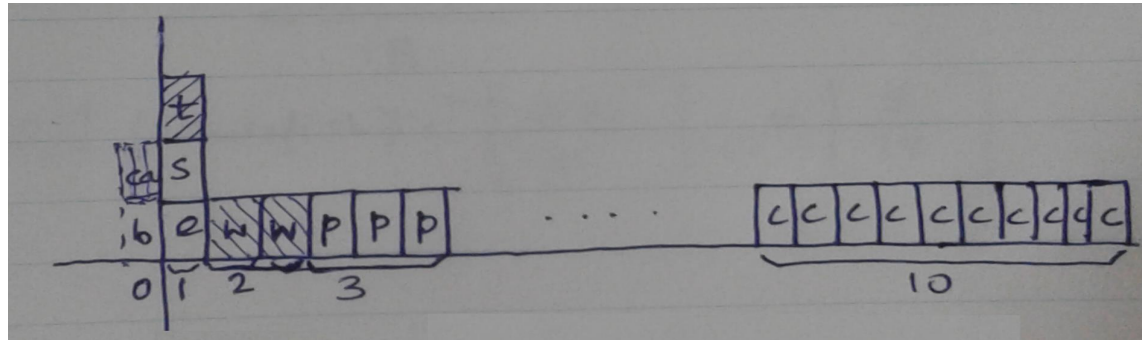
 $P_{GT}^*(\text{catfish}) = P_{GT}^*(\text{eel}) = \left[\frac{3/2}{18} \right]$

 $P_{GT}^*(\text{trout}) = P_{GT}^*(\text{salmon}) = P_{GT}^*(\text{eel}) = \frac{(2/3)}{18}$

 $P_{GT}^*(\text{whitefish}) = \left[\frac{3}{18} \right]$

$$c^* = \frac{0}{1} = 0$$

Intuition: Conservation of areas



$$c = 10$$

$$N_c = 1$$

$$c+1 = 11$$

$$N_{c+1} = 0$$

Relook at the equation:

$$c^* = (c+1) \frac{N_{c+1}}{N_c}$$

is equivalent to $\underbrace{c^* N_c}_{\text{area}} = \underbrace{(c+1) N_{c+1}}_{\text{area}}$

Example

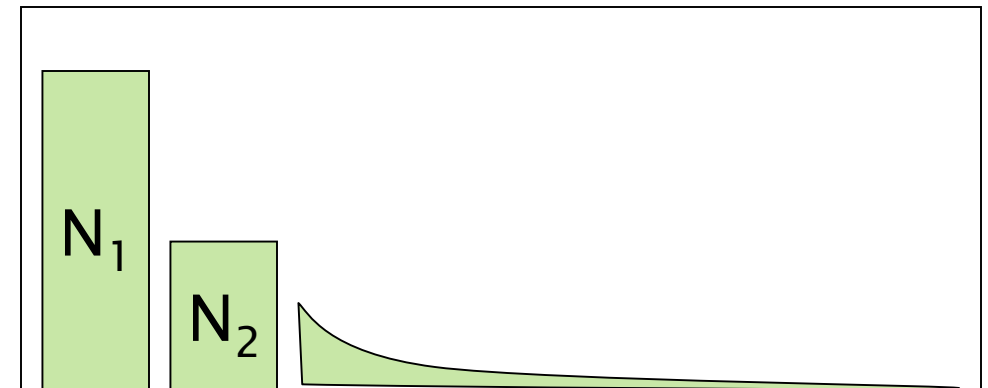
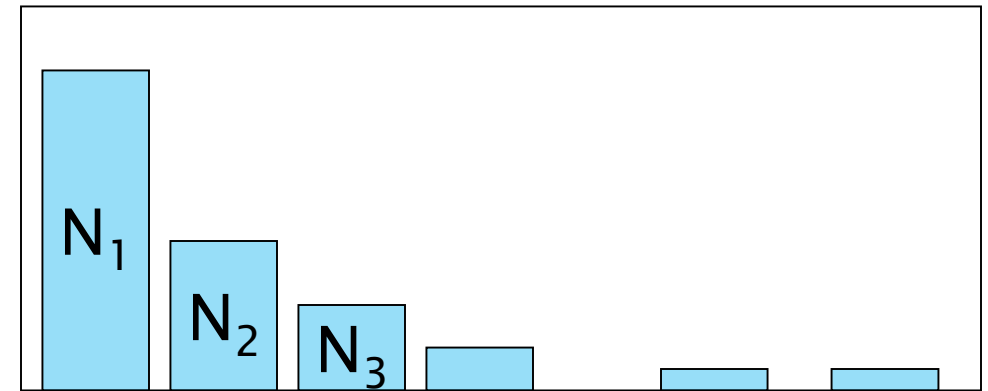
For trout: $c = 1$, $N_c = 3$,
 $N_{c+1} = 1$ (since $N_1 = 3, N_2 = 1$)
 so, $c^* = \frac{2 \times 1}{3} = \frac{2}{3}$

Good-Turing complications

(parts of slide from Dan Klein)

- Problem: what about “the”? (say $c=4417$)
 - For small k , $N_k > N_{k+1}$
 - For large k , too jumpy, zeros wreck estimates
- Simple Good-Turing [Gale and Sampson]:
replace empirical N_k with a best-fit power law
once counts get unreliable

$$\log(N_c) = a + b \log(c)$$



Good-Turing numbers

- Numbers from Church and Gale (1991)
- 22 million words of AP Newswire

$$c^* = \frac{(c + 1)N_{c+1}}{N_c}$$

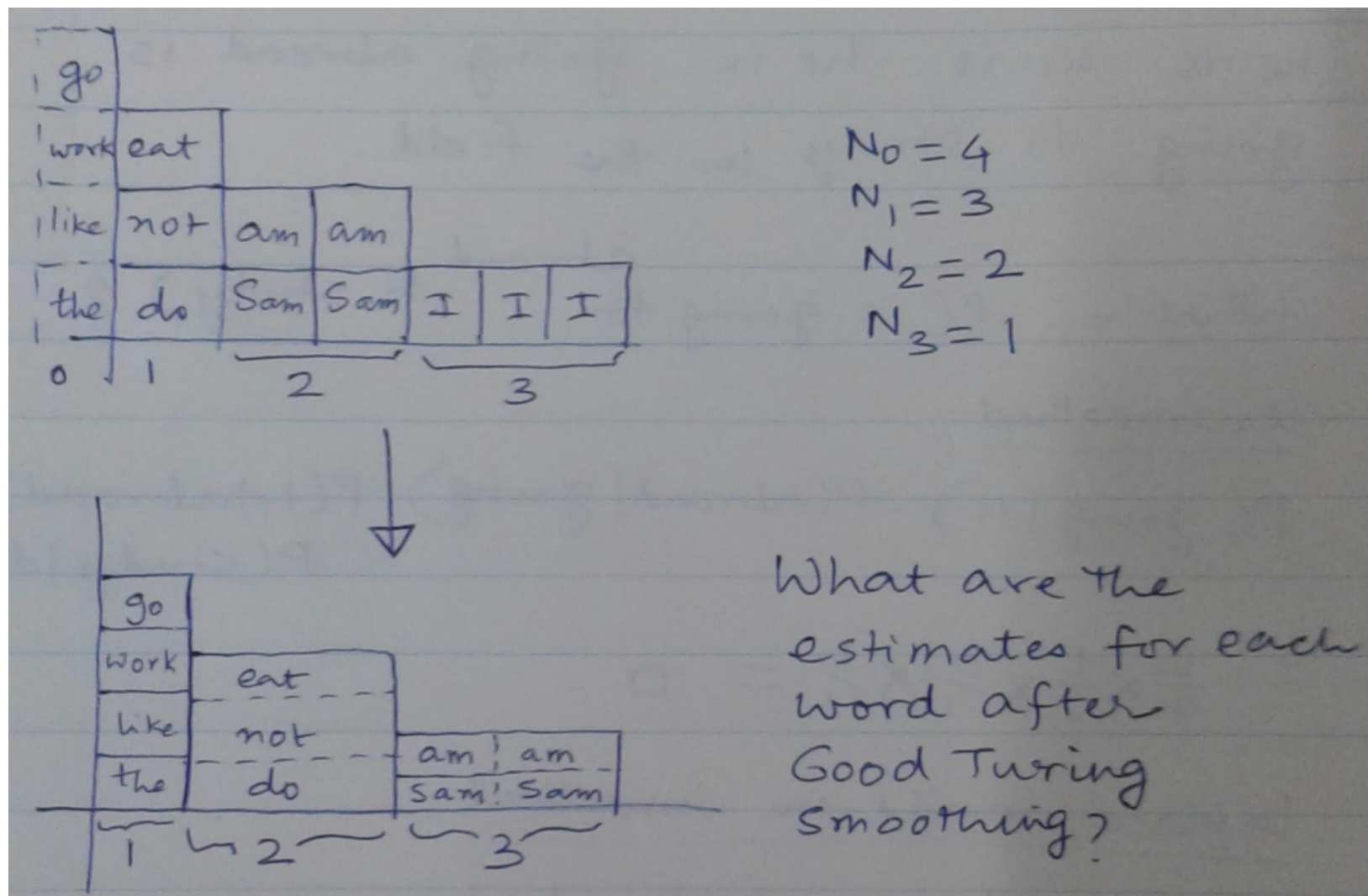
Count c	Good Turing c*
0	.0000270
1	0.446
2	1.26
3	2.24
4	3.24
5	4.22
6	5.19
7	6.21
8	7.24
9	8.25

Homework

- N_c = the count of things we've seen c times
- Corpus: Sam I am I am Sam I do not eat
- Words not seen: go, work, like, the

I	3	$N_0 = 4$
Sam	2	$N_1 = 3$
am	2	$N_2 = 2$
do	1	$N_3 = 1$
not	1	
eat	1	

Part of solution



Worked out example with bigrams

Corpus:

he is he is he is going abroad is
going to study in the field.

What is $P(\text{is going abroad to study})$?

Unsmoothed:

$$P(\text{going}|\text{is}) \cdot P(\text{abroad}|\text{going}) \cdot P(\text{to}|\text{abroad}) \cdot P(\text{study}|\text{do})$$

$$= \frac{2}{4} \times \frac{1}{2} \times \frac{0}{1} \times \frac{1}{1} = 0.$$

Laplace or add-one smoothed:

$$\frac{2+1}{4+9} \times \frac{1+1}{2+9} \times \frac{0+1}{1+9} \times \frac{1+1}{1+9}$$

$$= \frac{3}{13} \times \frac{2}{11} \times \frac{1}{10} \times \frac{2}{10}$$

Example Continued: Good Turing Smoothing

Corpus:
he is he is he is going abroad is
going to study in the field.
What is $P(\text{is going abroad to study})$?

Homework:

Complete the calculations, estimate the probability of sentence using bigram estimates, and compare with that obtained using MLE and Laplace estimates

Good Turing Smoothed.

c	N_c	P_{MLE}	P_{GT}^*
0	7	0	$7/14$
1	7	$2/14$	$4/14$
2	2	$4/14$	$3/14$
3	1	$3/14$	—

81-10

10 bigrams

Example calculation

N_c	
7	going abroad, abroad is, going to, to study, study in, in the, the field.
2	is he, is going.
3	he is.

Discounting

- Say we've seen the following counts:

x	Count(x)	$q_{\text{ML}}(w_i \mid w_{i-1})$
the	48	
the, dog	15	15/48
the, woman	11	11/48
the, man	10	10/48
the, park	5	5/48
the, job	2	2/48
the, telescope	1	1/48
the, manual	1	1/48
the, afternoon	1	1/48
the, country	1	1/48
the, street	1	1/48

- The maximum-likelihood estimates are high
(particularly for low count items)

Slides from NLP lectures by Michael Collins

Discounting

- ▶ Now define “discounted” counts,
 $\text{Count}^*(x) = \text{Count}(x) - 0.5$
- ▶ New estimates:

x	$\text{Count}(x)$	$\text{Count}^*(x)$	$\frac{\text{Count}^*(x)}{\text{Count}(\text{the})}$
the	48		
the, dog	15	14.5	14.5/48
the, woman	11	10.5	10.5/48
the, man	10	9.5	9.5/48
the, park	5	4.5	4.5/48
the, job	2	1.5	1.5/48
the, telescope	1	0.5	0.5/48
the, manual	1	0.5	0.5/48
the, afternoon	1	0.5	0.5/48
the, country	1	0.5	0.5/48
the, street	1	0.5	0.5/48

v_2 aa

$$\frac{10 \times 0.5}{48} = \frac{5}{48}$$

Slides from NLP lectures by Michael Collins

Discounting

We now have some "missing probability mass":

$$\alpha(w_{i-1}) = 1 - \sum_w \frac{\text{Count}^*(w_{i-1}, w)}{\text{Count}(w_{i-1})}$$

$w_{i-1} = \text{"the"}$
 $w_i = \text{word following } w_{i-1}$

e.g., in our example, $\alpha(\text{the}) = 10 \times 0.5/48 = 5/48$

Slides from NLP lectures by Michael Collins

Katz Back-off Models (Bigrams)

- For a bigram model, define two sets

$$\mathcal{A}(w_{i-1}) = \{w : \text{Count}(w_{i-1}, w) > 0\}$$

$$\mathcal{B}(w_{i-1}) = \{w : \text{Count}(w_{i-1}, w) = 0\}$$

- A bigram model

$$q_{BO}(w_i | w_{i-1}) = \begin{cases} \frac{\text{Count}^*(w_{i-1}, w_i)}{\text{Count}(w_{i-1})} & \text{If } w_i \in \mathcal{A}(w_{i-1}) \\ \alpha(w_{i-1}) \frac{q_{ML}(w_i)}{\sum_{w \in \mathcal{B}(w_{i-1})} q_{ML}(w)} & \text{If } w_i \in \mathcal{B}(w_{i-1}) \end{cases}$$

where

$$\alpha(w_{i-1}) = 1 - \sum_{w \in \mathcal{A}(w_{i-1})} \frac{\text{Count}^*(w_{i-1}, w)}{\text{Count}(w_{i-1})}$$

5/48

the institute
the cat
the monkey
the the

Slides from NLP lectures by Michael Collins

Katz Back-off Models (Bigrams)

- For a bigram model, define two sets

$$\mathcal{A}(w_{i-1}) = \{w : \text{Count}(w_{i-1}, w) > 0\}$$

$$\mathcal{B}(w_{i-1}) = \{w : \text{Count}(w_{i-1}, w) = 0\}$$

- A bigram model

$$q_{BO}(w_i | w_{i-1}) = \begin{cases} \frac{\text{Count}^*(w_{i-1}, w_i)}{\text{Count}(w_{i-1})} & \text{If } w_i \in \mathcal{A}(w_{i-1}) \\ \alpha(w_{i-1}) \frac{q_{ML}(w_i)}{\sum_{w \in \mathcal{B}(w_{i-1})} q_{ML}(w)} & \text{If } w_i \in \mathcal{B}(w_{i-1}) \end{cases}$$

where

$$\alpha(w_{i-1}) = 1 - \sum_{w \in \mathcal{A}(w_{i-1})} \frac{\text{Count}^*(w_{i-1}, w)}{\text{Count}(w_{i-1})}$$

Discounted counts distributed to bigrams whose counts are non-zero

Missing probability mass distributed to zero count bigrams

Slides from NLP lectures by Michael Collins

Katz Back-off Models (Trigrams)

- For a trigram model, first define two sets

$$\mathcal{A}(w_{i-2}, w_{i-1}) = \{w : \text{Count}(w_{i-2}, w_{i-1}, w) > 0\}$$

$$\mathcal{B}(w_{i-2}, w_{i-1}) = \{w : \text{Count}(w_{i-2}, w_{i-1}, w) = 0\}$$

- A trigram model is defined in terms of the bigram model:

$$q_{BO}(w_i \mid w_{i-2}, w_{i-1}) = \begin{cases} \frac{\text{Count}^*(w_{i-2}, w_{i-1}, w_i)}{\text{Count}(w_{i-2}, w_{i-1})} & \text{If } w_i \in \mathcal{A}(w_{i-2}, w_{i-1}) \\ \frac{\alpha(w_{i-2}, w_{i-1}) q_{BO}(w_i \mid w_{i-1})}{\sum_{w \in \mathcal{B}(w_{i-2}, w_{i-1})} q_{BO}(w \mid w_{i-1})} & \text{If } w_i \in \mathcal{B}(w_{i-2}, w_{i-1}) \end{cases}$$

where

$$\alpha(w_{i-2}, w_{i-1}) = 1 - \sum_{w \in \mathcal{A}(w_{i-2}, w_{i-1})} \frac{\text{Count}^*(w_{i-2}, w_{i-1}, w)}{\text{Count}(w_{i-2}, w_{i-1})}$$

Slides from NLP lectures by Michael Collins

Katz Back-off Models (Trigrams)

- For a trigram model, first define two sets

$$\mathcal{A}(w_{i-2}, w_{i-1}) = \{w : \text{Count}(w_{i-2}, w_{i-1}, w) > 0\}$$

$$\mathcal{B}(w_{i-2}, w_{i-1}) = \{w : \text{Count}(w_{i-2}, w_{i-1}, w) = 0\}$$

- A trigram model is defined in terms of the bigram model:

$$q_{BO}(w_i | w_{i-2}, w_{i-1}) = \begin{cases} \frac{\text{Count}^*(w_{i-2}, w_{i-1}, w_i)}{\text{Count}(w_{i-2}, w_{i-1})} & \text{If } w_i \in \mathcal{A}(w_{i-2}, w_{i-1}) \\ \frac{\alpha(w_{i-2}, w_{i-1}) q_{BO}(w_i | w_{i-1})}{\sum_{w \in \mathcal{B}(w_{i-2}, w_{i-1})} q_{BO}(w | w_{i-1})} & \text{If } w_i \in \mathcal{B}(w_{i-2}, w_{i-1}) \end{cases}$$

Discounted counts distributed to trigrams whose counts are non-zero

Missing probability mass distributed to zero count trigrams

NOTE THE RECURSIVE FORMULATION

where

$$\alpha(w_{i-2}, w_{i-1}) = 1 - \sum_{w \in \mathcal{A}(w_{i-2}, w_{i-1})} \frac{\text{Count}^*(w_{i-2}, w_{i-1}, w)}{\text{Count}(w_{i-2}, w_{i-1})}$$

Summary

- Add-1 smoothing and its generalization: add-k smoothing
- Interpolation and Backoff
- Good Turing Smoothing
- Katz Backoff

Reference

- Relevant sections from shared chapter from J&M