

corpora, language models, and smoothing

CSC401/2511 – Natural Language Computing – Spring 2019
Lecture 2 Frank Rudzicz and Chloé Pou-Prom
University of Toronto

Lecture 2 overview

- This lecture:
 - Linguistic data,
 - Language models
 - i.e., “ N -grams”
 - Smoothing
- Some slides are based on content from Bob Carpenter, Dan Klein, Roger Levy, Josh Goodman, Dan Jurafsky, Christopher Manning, Gerald Penn, and Bill MacCartney.

Word prediction with N -grams

- **N -grams**: $n.pl.$ **token** sequences of length N .
 - The fragment 'in this sentence is' contains the following 2-grams (i.e., '**bigrams**'):
 - (*in this*), (*this sentence*), (*sentence is*)
 - The next bigram **must** start with '*is*'.
 - What word is **most likely** to follow '*is*'?

Example bigram probabilities

- Obtain likelihoods by dividing bigram counts by unigram counts.

Unigram counts:

I	want	to	eat	Chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

$P(w_t w_{t-1})$	I	want	to	eat	Chinese	food	lunch	spend
I	0.002	0.33	0	0.0036	0	0	0	0.00079

$$P(\text{want}|I) \approx \frac{\text{Count}(I \text{ want})}{\text{Count}(I)} = \frac{827}{2533} \approx 0.33$$

$$P(\text{spend}|I) \approx \frac{\text{Count}(I \text{ spend})}{\text{Count}(I)} = \frac{2}{2533} \approx 7.9 \times 10^{-4}$$

Example bigram probabilities

- Obtain likelihoods by dividing bigram counts by unigram counts.

Unigram counts:

I	want	to	eat	Chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

$P(w_t w_{t-1})$	I	want	to	eat	Chinese	food	lunch	spend
I	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
Chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

intrinsic vs extrinsic Maximum likelihood estimate

- **Maximum likelihood estimate (MLE)** of parameters θ in a **model** M , given **training data** T is

the estimate that maximizes the likelihood of the *training data* using the *model*.

- e.g.,
 - T is the Brown corpus,
 - M is the bigram and unigram tables
 - $\theta_{(to|want)}$ is $P(to|want)$.
- In fact, we have been doing MLE, within the N -gram context, **all along** with our simple counting.

Perplexity

- Perplexity of corpus C , $PP(C) = 2^{-\left(\frac{\log_2 P(C)}{\|C\|}\right)}$
- If you have a vocabulary \mathcal{V} with $\|\mathcal{V}\|$ word types, and your LM is *uniform* (i.e., $P(w) = 1/\|\mathcal{V}\| \forall w \in \mathcal{V}$),

- Then

$$PP(C) = 2^{-\left(\frac{\log_2 P(C)}{\|C\|}\right)} = 2^{-\left(\frac{\log_2 [(1/\|\mathcal{V}\|)^{\|C\|}]}{\|C\|}\right)} = 2^{-\log_2 (1/\|\mathcal{V}\|)} = 2^{\log_2 \|\mathcal{V}\|} = \|\mathcal{V}\|$$

- Perplexity is sort of like a 'branching factor'.

- Minimizing perplexity \equiv maximizing probability of corpus

ZIPF AND THE NATURAL DISTRIBUTIONS IN LANGUAGE

Sparseness

- Problem with N -gram models:
 - New **words** appear often as we read **new data**.
 - e.g., *interfrastic*, *espepsia*, \$182,321.09
 - New **bigrams** occur *even more* often.
 - Recall that Shakespeare only wrote $\sim 0.04\%$ of all the bigrams he *could* have, given his vocabulary.
 - Because there are so many *possible* bigrams, we encounter new ones *more frequently* as we read.
 - New **trigrams** occur *even more even-more-often*.

Sparseness of unigrams vs. bigrams

- Conversely, we can see lots of every *unigram*, but still miss many *bigrams*:

Unigram counts:

I	want	to	eat	Chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

$Count(w_{t-1}, w_t)$		w_t							
		I	want	to	eat	Chinese	food	lunch	spend
w_{t-1}	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

Why does sparseness happen?

- The bigram table appears to be filled in **non-uniformly**.
- Clearly, some words (e.g., *want*) are very **popular** and will occur in **many bigrams** just from random chance.
- Other words are not-so-popular (e.g., *hippopotomonstrosesquipedalian*). They will occur **infrequently**, and when they do their partner word will have its own $P(w)$.
- *Is there some phenomenon that describes $P(w)$ in real language?*

Patterns of unigrams

- Words in *Tom Sawyer* by Mark Twain:

Word	Frequency
the	3332
and	2972
a	1775
to	1725
of	1440
was	1161
it	1027
in	906
that	877
he	877
...	...

- A ***few*** words occur ***very frequently***.
 - Aside: the *most frequent* 256 English word types account for 50% of English tokens.
 - Aside: for Hungarian, we need the top 4096 to account for 50%.
- Many*** words occur ***very infrequently***.

Frequency of frequencies

- How many words occur X number of times in *Tom Sawyer*?

Hapax legomena: *n.pl.*
words that occur once
in a corpus.

Word frequency	# of word types with that frequency
1	3993
2	1292
3	664
4	410
5	243
6	199
7	172
8	131
9	82
10	91
11-50	540
51-100	99
>100	102

e.g.,
1292 word types
occur twice

Notice how many
word types are
relatively rare!

hapax
lego-
menon

Ranking words in *Tom Sawyer*

- Rank word types in order of decreasing frequency.

Word	Freq. (<i>f</i>)	Rank (<i>r</i>)	<i>f</i> · <i>r</i>
the	3332	1	3332
and	2972	2	5944
a	1775	3	5235
he	877	10	8770
but	410	20	8400
be	294	30	8820
there	222	40	8880
one	172	50	8600
about	158	60	9480
more	138	70	9660
never	124	80	9920

Word	Freq. (<i>f</i>)	Rank (<i>r</i>)	<i>f</i> · <i>r</i>
name	21	400	8400
comes	16	500	8000
group	13	600	7800
lead	11	700	7700
friends	10	800	8000
begin	9	900	8100
family	8	1000	8000
brushed	4	2000	8000
sins	2	3000	6000
Could	2	4000	8000
Applausive	1	8000	8000

With some
(relatively minor)
exceptions,
f·*r* is very
consistent!

Zipf's Law

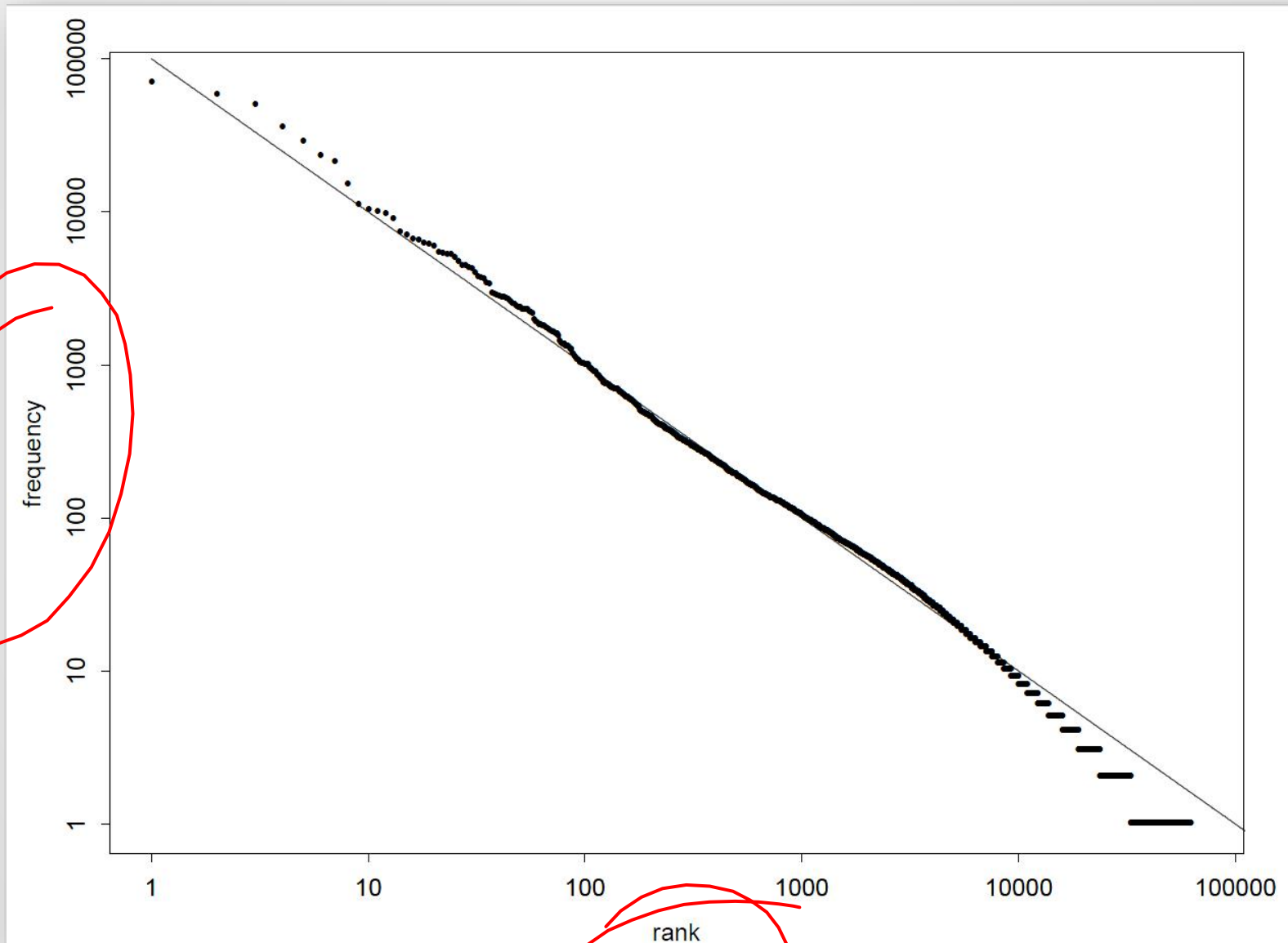
- In *Human Behavior and the Principle of Least Effort*, Zipf argues^(*) that all human endeavour depends on laziness.
 - Speaker minimizes effort by having a **small** vocabulary of **common** words.
 - Hearer minimizes effort by having a **large** vocabulary of **less ambiguous** words.
- Compromise: frequency and rank are inversely proportional.

$$f \propto \frac{1}{r} \quad \text{i.e., for some } k \quad f \cdot r = k$$

(*) This does not make it true.

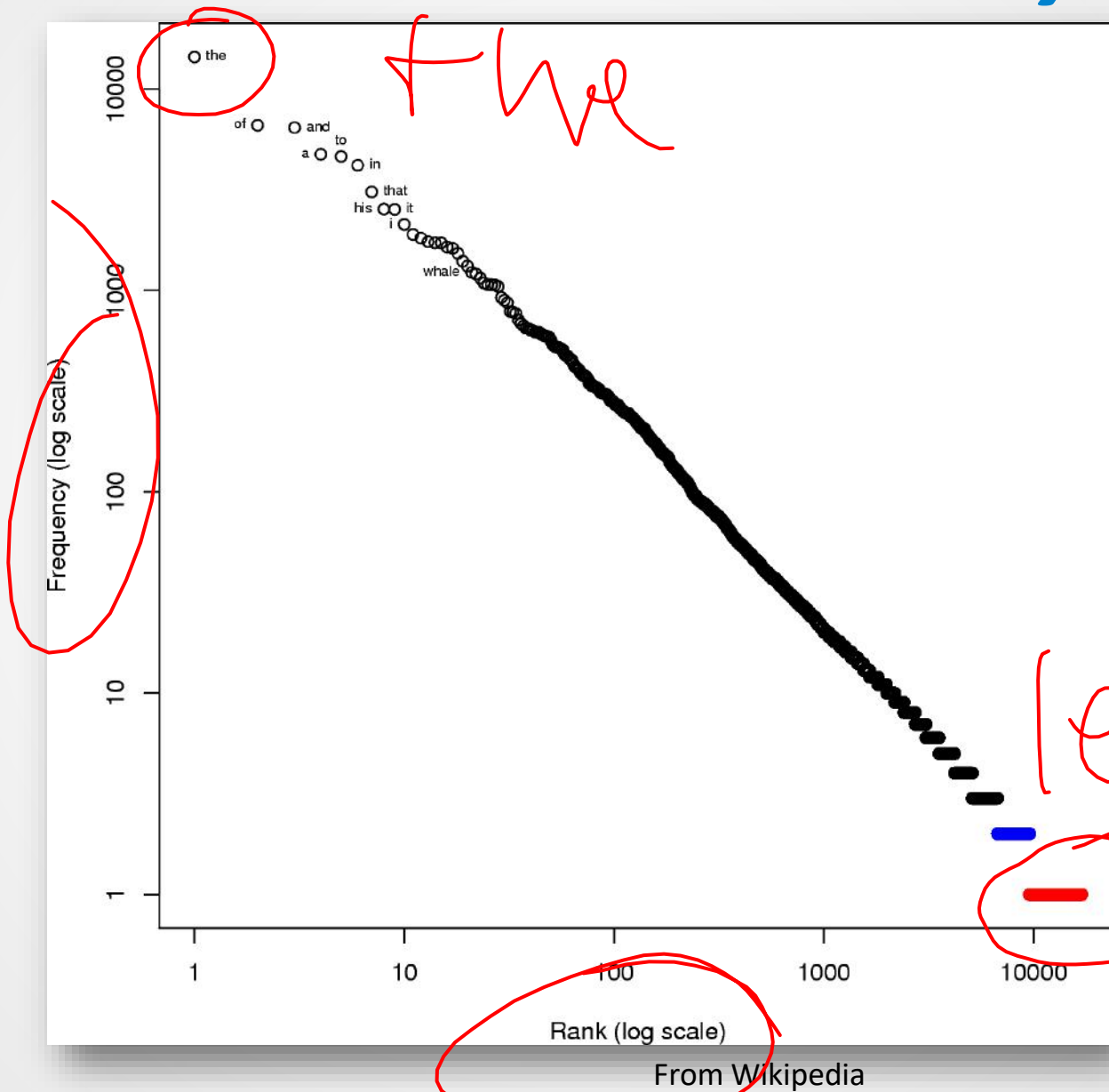
Zipf's Law on the Brown corpus

196015



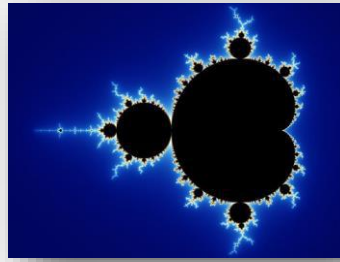
From Manning & Schütze

Zipf's Law on the novel *Moby Dick*



~~60~~

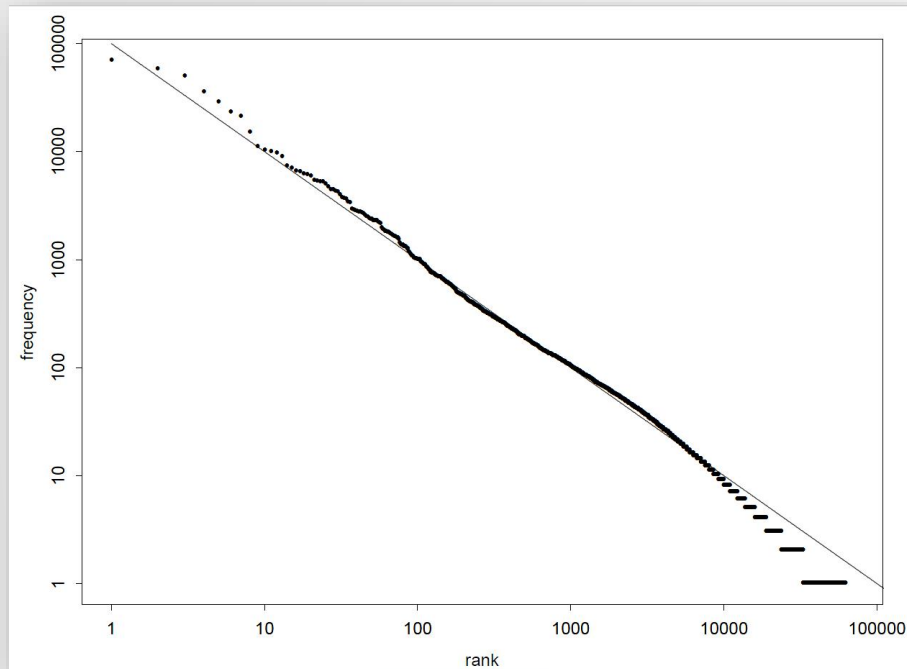
Mandelbrot



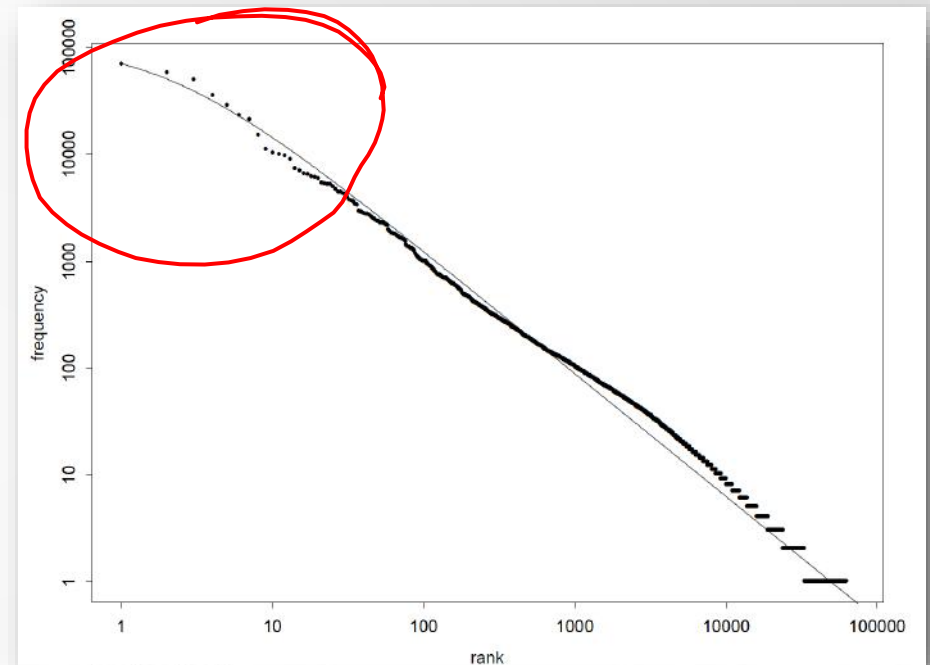
- In “Structure formelle des textes et communication”. *Word 10:1—27*, Benoit Mandelbrot claimed that Zipf lacks detail.
- With hand-tuneable parameters \mathcal{P} , B and ρ , he suggests

$$f = \mathcal{P} \cdot (r + \rho)^{-B}$$

Zipf vs. Mandelbrot on Brown corpus



Zipf



Mandlebrot

Zipf's Law in perspective

- Zipf's explanation of the **phenomenon** involved human laziness.
- Simon's discourse model (1956) argued that the phenomenon could equally be explained by two processes:
 - People imitate relative frequencies of words they hear
 - People innovate new words with small, constant probability
- There are other explanations.

→ many meanings Aside – Zipf's Law in perspective

- Zipf *also* observed that **frequency** *correlates* with several **other** properties of words, e.g.:
 - Age (frequent words are old)
 - Polysemy (frequent words often have many meanings or higher-order functions of meaning, e.g., *chair*)
 - Length (frequent words are spelled with few letters)
- He also showed that there are hyperbolic distributions in the world (crucially, they're not Gaussian), just like:
 - Yule's Law: $B = 1 + \frac{g}{s}$
 - s : probability of mutation becoming dominant in species
 - g : probability of mutation that expels species from genus
 - Pareto distributions (wealth distribution)

0-count

SMOOTHING

Zero probability in Shakespeare

- Shakespeare's collected writings account for about 300,000 bigrams out of a possible $V^2 \approx 845M$ bigrams, given his lexicon.
- So 99.96% of the possible bigrams were **never** seen.
- Now imagine that someone finds a **new play** and wants to know whether it is Shakespearean...
- Shakespeare isn't very predictable! Every time the play uses one of those 99.96% bigrams, the sentence that contains it (and the play!) gets **0 probability**.
- *This is bad.*

Zero probability in general

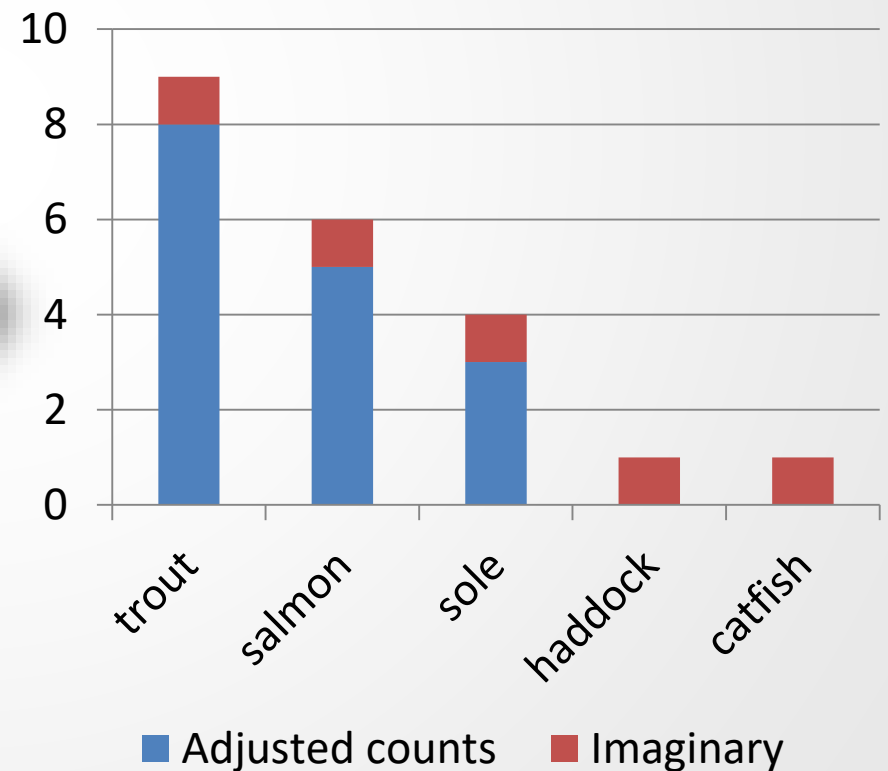
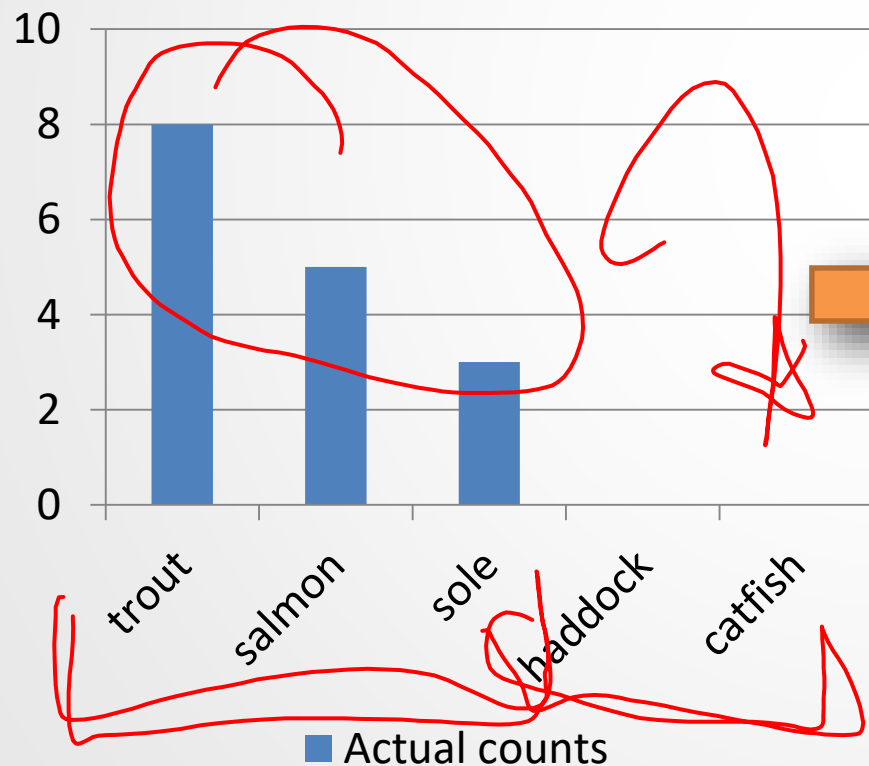
- Some N -grams are just *really rare*.
 - e.g., perhaps '*negative press covfefe*'
- If we had more data, *perhaps* we'd see them.
- If we have no way to determine the distribution of *unseen* N -grams, how can we estimate them?

Smoothing mechanisms

- Smoothing methods include:
 1. Add- δ smoothing (Laplace)
 2. Good-Turing smoothing
 3. Katz smoothing
 4. Simple interpolation (Jelinek-Mercer)
 5. Absolute discounting
 6. Kneser-Ney smoothing

Smoothing as redistribution

- Make the distribution more uniform.
- This moves the probability mass from 'the rich' towards 'the poor'.



1. Add-1 smoothing (“Laplace discounting”)

- Given vocab size $\|\mathcal{V}\|$ and corpus size $N = \|\mathcal{C}\|$.
- Just add 1 to all the counts! No more zeros!

- MLE

$$: P(w) = \text{Count}(w) / N$$

- Laplace estimate

$$: P_{Lap}(w) = \frac{\text{Count}(w) + 1}{N + \|\mathcal{V}\|}$$

- Does this give a proper probability distribution? Yes:

$$\sum_w P_{Lap}(w) = \sum_w \frac{\text{Count}(w) + 1}{N + \|\mathcal{V}\|} = \frac{\sum_w \text{Count}(w) + \sum_w 1}{N + \|\mathcal{V}\|} = \frac{N + \|\mathcal{V}\|}{N + \|\mathcal{V}\|} = 1$$

1. Add-1 smoothing for bigrams

- Same principle for bigrams:

$$P_{Lap}(w_t|w_{t-1}) = \frac{\text{Count}(w_{t-1}w_t) + 1}{\text{Count}(w_{t-1}) + \|\mathcal{V}\|}$$

- We are essentially holding out and spreading $\|\mathcal{V}\|/(N + \|\mathcal{V}\|)$ uniformly over “imaginary” events.
- Does this work?

1. Laplace smoothed bigram counts

- Out of 9222 sentences in Berkeley restaurant corpus,
 - e.g., “*I want*” occurred 827 times so Laplace gives 828

$Count(w_{t-1}, w_t)$		w_t							
		I	want	to	eat	Chinese	food	lunch	spend
w_{t-1}	I	5+1	827+1	1	9+1	1	1	1	2+1
	want	2+1	1	608+1	1+1	6+1	6+1	5+1	1+1
	to	2+1	1	4+1	686+1	2+1	1	6+1	211+1
	eat	1	1	2+1	1	16+1	2+1	42+1	1
	Chinese	1+1	1	1	1	1	82+1	1+1	1
	food	15+1	1	15+1	1	1+1	4+1	1	1
	lunch	2+1	1	1	1	1	1+1	1	1
	spend	1+1	1	1+1	1	1	1	1	1

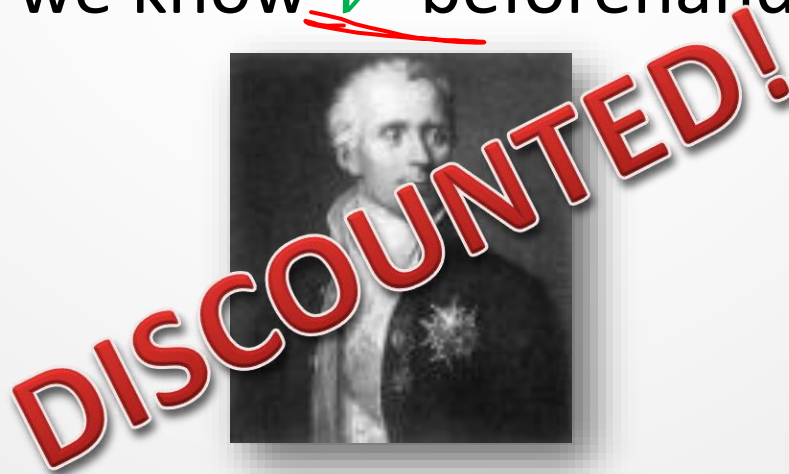
1. Laplace smoothed probabilities

$$P_{Lap}(w_t|w_{t-1}) = \frac{C(w_{t-1}w_t) + 1}{C(w_{t-1}) + \|\mathcal{V}\|}$$

$P(w_t w_{t-1})$	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	0.00083	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

1. Add-1 smoothing

- According to this method, $P(to|want)$ went from 0.66 to 0.26.
 - That's a huge change!
- In **extrinsic** evaluations, the results are **not great**.
- Sometimes ~90% of the probability mass is spread across unseen events.
- It only works if we know v beforehand.



1. Add- δ smoothing

Handwritten notes: w_0 , w_1 , w_2 and a table with $0+\delta$, $1+\delta$, $2+\delta$ in the second column, and a vertical line separating it from the first column.

- Generalize Laplace: Add $\delta < 1$ to be a bit less generous.

- MLE

$$: P(w) = \text{Count}(w)/N$$

- Add- δ estimate

$$: P_{\text{add-}\delta}(w) = \frac{\text{Count}(w) + \delta}{N + \delta \|\mathcal{V}\|}$$

- Does this give a proper probability distribution? Yes:

$$\begin{aligned} \sum_w P_{\text{add-}\delta}(w) &= \sum_w \frac{\text{Count}(w) + \delta}{N + \delta \|\mathcal{V}\|} = \frac{\sum_w \text{Count}(w) + \sum_w \delta}{N + \delta \|\mathcal{V}\|} \\ &= \frac{N + \delta \|\mathcal{V}\|}{N + \delta \|\mathcal{V}\|} = 1 \end{aligned}$$

This sometimes works empirically (e.g., in text categorization), sometimes not...

Is there another way?

words seen
once
↑

- Has Zipf taught us *nothing*?
 - We shouldn't adjust all words uniformly.
 - **Unseen** words should behave more like **hapax legomena**.
 - Words that occur **a lot** should behave like other words that occur **a lot**.
- If I keep reading from a corpus, by the time I see a new word like 'zenzizenzizenic', I will have seen 'the' a lot more than once more.

2. Good-Turing

$c \rightarrow 0$



- Define N_c as the number of N -grams that occur c times.

Word frequency	# of words (i.e., unigrams) with that frequency
1	$N_1 = 3993$
2	$N_2 = 1292$
3	$N_3 = 664$
...	...

(from *Tom Sawyer*)

$$p(w) = \frac{1}{N}$$

- For some word in 'bin' N_c , the MLEstimate is that I saw that word c times.
- Idea:** get rid of zeros by re-estimating c using the MLE estimate of words that occur $c + 1$ times.

2. Good-Turing intuition/example

- Imagine you have this toy scenario:

Word	ship	pass	camp	frock	soccer	mother	tops
Frequency	8	7	3	2	1	1	1

= 23 words total

- What is the MLE prior probability of hearing 'soccer'?
 - $P(\text{soccer}) = 1/23$
- What is the probability of seeing something **new**?
 - No way to tell, but $3/23$ words are hapax legomena ($N_1 = 3$).
 - If we use $3/23$ to approximate things we've never seen, then we have to also adjust other probabilities (e.g., $P_{GT}(\text{soccer}) < 1/23$).

$$P(\text{unseen-word}) = 3/23$$

2. Good-Turing adjustments

- $P_{GT}^*([unseen]) = N_1/N$
- Re-estimate count $c^* = \frac{(c+1)N_{c+1}}{N_c}$

• Unseen words

- $c = 0$
- MLE: $p = 0/23$
- $P_{GT}^*([unseen]) = \frac{N_1}{N}$
 $= 3/23$

• Seen once (e.g., *soccer*)

- $c = 1$
- MLE: $p = 1/23$
- $c^*(\textit{soccer}) = 2 \cdot \frac{N_2}{N_1}$
 $= 2 \cdot 1/3$
- $P_{GT}^*(\textit{soccer}) = (\frac{2}{3})/23$

2. Good-Turing limitations

- Q: What happens when you want to estimate $P(w)$ when w occurs c times, but no word occurs $c + 1$ times?
 - E.g., what is $P_{GT}^*(camp)$ since $N_4 = 0$?

Word	ship	pass	camp	frock	soccer	mother	tops
Frequency	8	7	3	2	1	1	1

- A1: We can **re-estimate** count $c^* = \frac{(c+1)E[N_{c+1}]}{E[N_c]}$.
 - We can use Expectation-Maximization, which we'll see later.
- A2: We can **interpolate linearly**, in log-log, between values of c that we *do* have.

2. Good-Turing limitations

- Q: What happens when $\text{Count}(\text{McGill genius}) = 0$ and $\text{Count}(\text{McGill brainbox}) = 0$, and we smooth bigrams?
- A: $P(\text{genius}|\text{McGill}) = P(\text{brainbox}|\text{McGill})$
 - But we'd expect $P(\text{genius}|\text{McGill}) > P(\text{brainbox}|\text{McGill})$ (context notwithstanding) because 'genius' is a more common word than 'brainbox').
- The solution may be to **combine** bigram and unigram models...

3. Katz backoff

- N -grams with non-zero count, e.g., $c = C(w_{t-1}w_t)$, are discounted according to a ratio d_c , similar to Good-Turing.
- 'Count mass' subtracted from existing N -grams are redistributed to $(N - 1)$ -grams.

$$C_{katz}(w_{t-1}w_t) = \begin{cases} d_c \cdot C(w_{t-1}w_t) & \text{if } C(w_{t-1}w_t) > 0 ; d_c \leq 1 \\ \alpha(w_{t-1})P(w_t) & \text{otherwise} \end{cases}$$

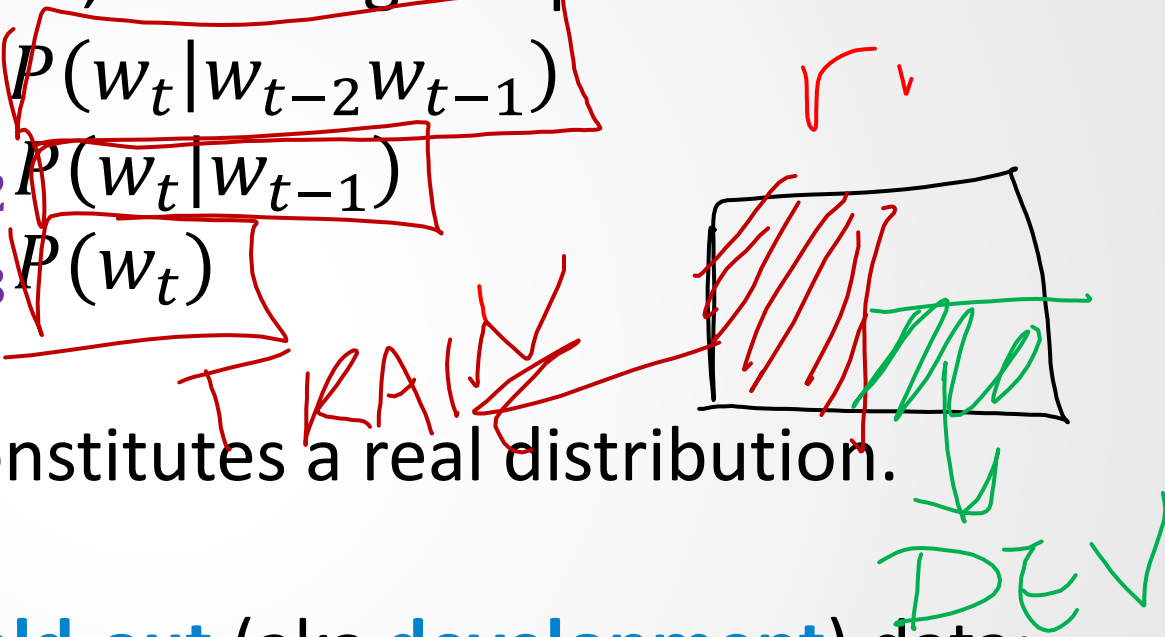
$$P_{katz}(w_t|w_{t-1}) = \frac{C_{katz}(w_{t-1}w_t)}{\sum_{w_i} C_{katz}(w_{i-1}w_i)}$$

3. Katz backoff

$$C_{katz}(w_{t-1}w_t) = \begin{cases} d_c \cdot C(w_{t-1}w_t) & \text{if } C(w_{t-1}w_t) > 0; d_c \leq 1 \\ \alpha(w_{t-1})P(w_t) & \text{otherwise} \end{cases}$$

- We set $\alpha(w_{t-1})$ so $\sum_{w_t} C_{katz}(w_{t-1}w_t) = \sum_{w_t} C(w_{t-1}w_t)$.
 - The solution is non-trivial (but close), and left as an exercise.
- Katz suggests 'large' counts ($c > 5$) are reliable; $d_{c>5} = 1$.
- Otherwise, we set d_c so that the total discount equals the fictional counts given by Good-Turing to unseen events.
 - I.e., solve for $\sum_{c=1}^k n_c(1 - d_c) \cdot c = n_1$
- Katz generalizes to higher-order N -grams, recursively.

4. Simple interpolation (Jelinek-Mercer)

- Combine trigram, bigram, and unigram probabilities.
- $\hat{P}(w_t|w_{t-2}w_{t-1}) = \lambda_1 P(w_t|w_{t-2}w_{t-1}) + \lambda_2 P(w_t|w_{t-1}) + \lambda_3 P(w_t)$ 
- With $\sum_i \lambda_i = 1$, this constitutes a real distribution.
- λ_i determined from **held-out** (aka **development**) data:
 - Fix N -gram probabilities on training set.
 - Adjust λ_i that give highest probability to held-out data.
 - (again, we can use “expectation-maximization”, to be discussed later)

held out = dev = validation

5. Absolute discounting

T	D
2	2 - δ
3	3 - δ

- Instead of multiplying highest N -gram by a λ_i , just subtract a fixed discount $0 \leq \delta \leq 1$ from each non-zero count.

$$P_{abs}(w_t | w_{t-n+1:t-1}) = \frac{\max(C(w_{t-n+1:t}) - \delta, 0)}{C(w_{t-n+1:t-1})} + (1 - \lambda_{w_{t-n+1:t-1}}) P_{abs}(w_t | w_{t-n+2:t-1})$$

The $n-1$ words of context
 The discounted ML estimate
 The weighting factor for the $n-1$ words of context
 And recurse using the $n-2$ words of context

- Once again, you need to learn λ and δ using held-out data.

6. Kneser-Ney smoothing

- In **interpolation**, lower-order (e.g., $N - 1$) models should only be useful if the N -gram counts are close to 0.
 - E.g., unigram models *should* be optimized for when bigrams are not sufficient.
- Imagine the bigram ‘San Francisco’ is common \therefore ‘Francisco’ has a very high unigram probability because it occurs a lot.
 - But ‘Francisco’ only occurs after ‘San’.
- **Idea:** We should give ‘Francisco’ a low unigram probability, because it only occurs within the well-modeled ‘San Francisco’.

6. Kneser-Ney smoothing

- Let the unigram count be the number of different words that it follows. I.e.:

$$N_{1+}(\bullet w_t) = |w_{t-1} : C(w_{t-1}w_t) > 0|$$

$$N_{1+}(\bullet\bullet) = \sum_{w_i} N_{1+}(\bullet w_i) \quad \leftarrow \text{The total number of bigram types.}$$

- So, the unigram probability is $P_{KN}(w_t) = \frac{N_{1+}(\bullet w_t)}{N_{1+}(\bullet\bullet)}$, and:

$$P_{KN}(w_t | w_{t-n+1:t-1}) = \frac{\max(C(w_{t-n+1:t}) - \delta, 0)}{\sum_i C(w_{i-n+1:i})} + \frac{\delta N_{1+}(w_{t-n+1:t-1}\bullet)}{\sum_{w_i} C(w_{i-n+1:i})} P_{KN}(w_t | w_{t-n+2:t-1})$$

Where $N_{1+}(w_{t-n+1:t-1}\bullet)$ is the number of possible words that follow the context.

Smoothing over smoothing

- **Interpolation** and **backoff** involve combining higher- and lower-order models.
- Only **interpolation** includes information from lower-order models when higher-order models have non-zero counts.
- Jelinek-Mercer performs better on small training sets; Katz performs better on large training sets.
- Katz smoothing performs well on N-grams with large counts; Kneser-Ney is best for small counts.
- Interpolated models are superior to backoff models for low (nonzero) counts.

Announcements and reading

- Chen & Goodman (1996) [An Empirical Study of Smoothing Techniques for Language Modeling](#), Proceedings of the 34th annual meeting of the Association for Computational Linguistics, Pages 310-318.
- Jurafsky & Martin (*2nd ed*): 4.1-4.7
- Manning & Schutze: 6.1-6.2.2, 6.2.5, 6.3