corpora, language models, and smoothing

CSC401/2511 – Natural Language Computing – Spring 2023

Lecture 2 En-Shiun Annie Lee, Raeid Saqur, Zining Zhu

University of Toronto

Types vs. Tokens

The cat in *the* hat

- Token: instance of word (the: 2)
- Type: “kind” of word (the: 1)
- Not clear in other cases:
  - *run* vs. *runs*
  - *happy* vs. *happily*
  - *frágment* vs. *fragmént*
  - *email* vs. *e-mail*
  - *hat* vs. *hat,*
  - speech disfluencies?

Corpora

- **Corpus**: *n.* A body of language data of a particular sort (*pl.* **corpora**).
- The **best** corpora occur **naturally**.
  - e.g., newspaper articles, telephone conversations, multilingual transcripts of the United Nations, tweets.
  - Some question now as to utility of **synthetic** corpora.
- We use corpora to gather statistics.
  - More is better.
  - Beware of bias.
- Examples: *Canadian Hansards*, *Project Gutenberg* (e-books), web crawls (*Google N-Gram, Common Crawl)*

Corpus (*pl.* Corpora)

A corpus is a collection of text(s) or utterances

- 10^6: tiny
- 10^9: reasonable
- 10^{13}: GPT-3
- 10^{14}: GPT-4

Lexicon

A collection of word-types: like a dictionary, but not necessarily with meanings

ZIPF AND NATURAL DISTRIBUTIONS IN LANGUAGE

Frequency Statistics

(Term) Frequency

$$TF(w, S) = \text{ \# tokens of } w \text{ in corpus } S$$

Relative Frequency:

$$F_S(w) = \frac{TF(w, S)}{|S|}$$

What happens to $F_S(w)$ as $|S|$ grows?
Answer: $F_S(w)$ converges to $p(w)$
This is the *frequentist* view of probability theory.

Frequency Statistics

(Term) Frequency

$$TF(w, S) = \text{ \# tokens of } w \text{ in corpus } S$$

Relative Frequency:

$$F_S(w) = \frac{TF(w, S)}{|S|}$$

What happens to $F_S(w)$ as $|S|$ and lexicon $|V|$ grow?

Frequency Statistics

(Term) Frequency

$$TF(w, S) = \text{ \# tokens of } w \text{ in corpus } S$$

Relative Frequency:

$$F_S(w) = \frac{TF(w, S)}{|S|}$$

What happens to $F_S(w)$ as $|S|$ and lexicon $|V|$ grow?
Answer: Average rel. freq. converges to 0.
That means that there are more and more infrequent words.
Not at all unusual for a word to have prob. 10^{-7}.

Frequency Statistics

(Term) Frequency

$$TF(w, S) = \text{ \# tokens of } w \text{ in corpus } S$$

Relative Frequency:

$$F_S(w) = \frac{TF(w, S)}{|S|}$$

What happens to $F_S(w)$ as $|S|$ and lexicon $|V|$ grow?
But rel. freq. itself stabilizes — surprise!
Let $N = |S|$:

$$\log(F_r)_V + \log N \approx H_N - B_N \log(\frac{r}{|V|})$$

The Zipf-Mandelbrot Equation

$$\log(F_r)_V + \log N \approx H_N - B_N \log(\frac{r}{|V|})$$

Line up all of the word types by (rel.) frequency:

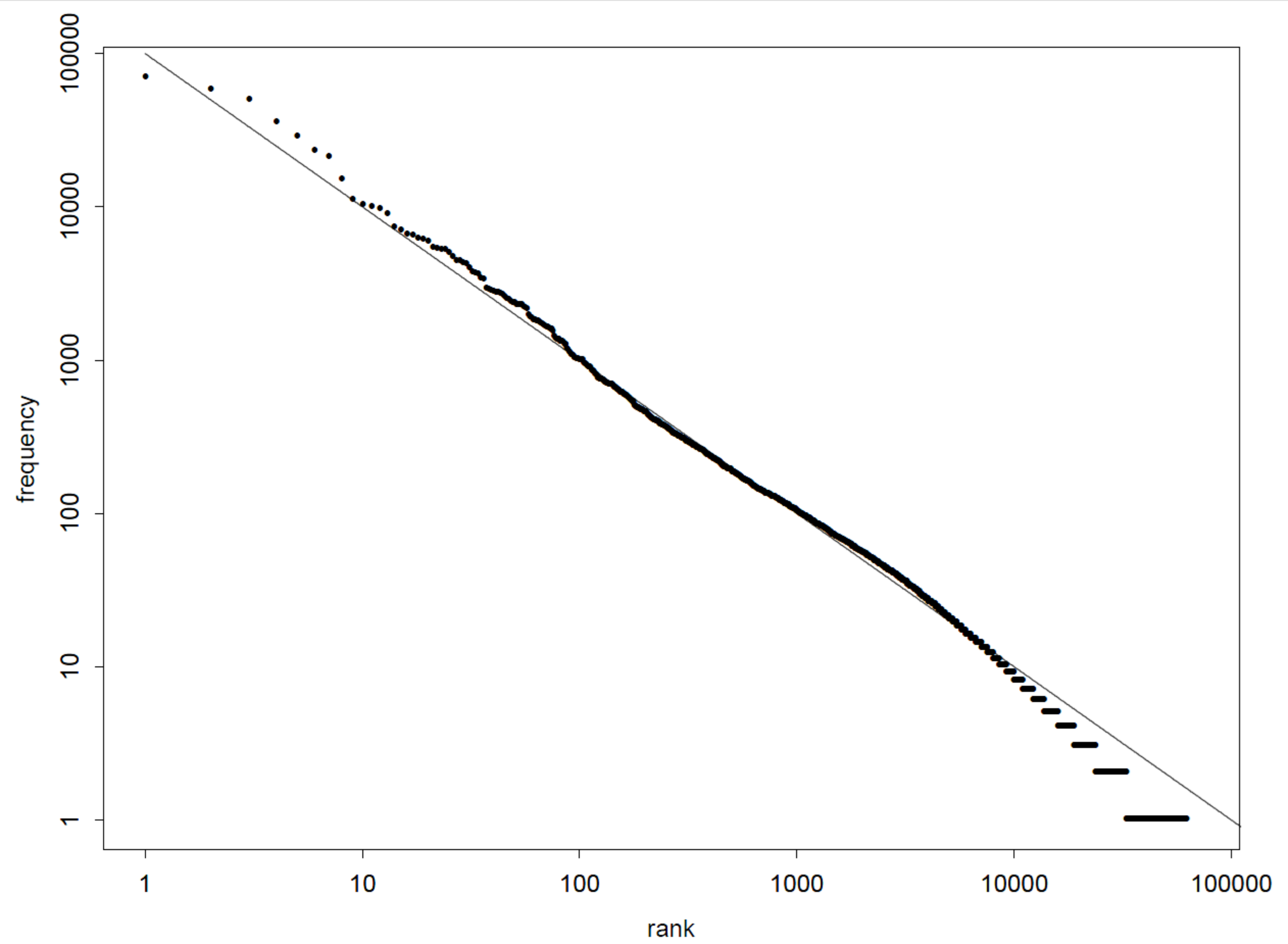
TF(w)	3000	2900	1750	1700	...
w	the	and	a	to	...
r	1	2	3	4	...

r: rank
F_r: the rel. freq. of the r^{th} ranked word.

$H_N \longrightarrow 0$ because lowest rank word should occur with rel. freq. $\frac{1}{N}$ (*hapax legomenon* — often typos)
But when $B_N \longrightarrow B \neq 0$, then we say that the population is *Zipfian*.

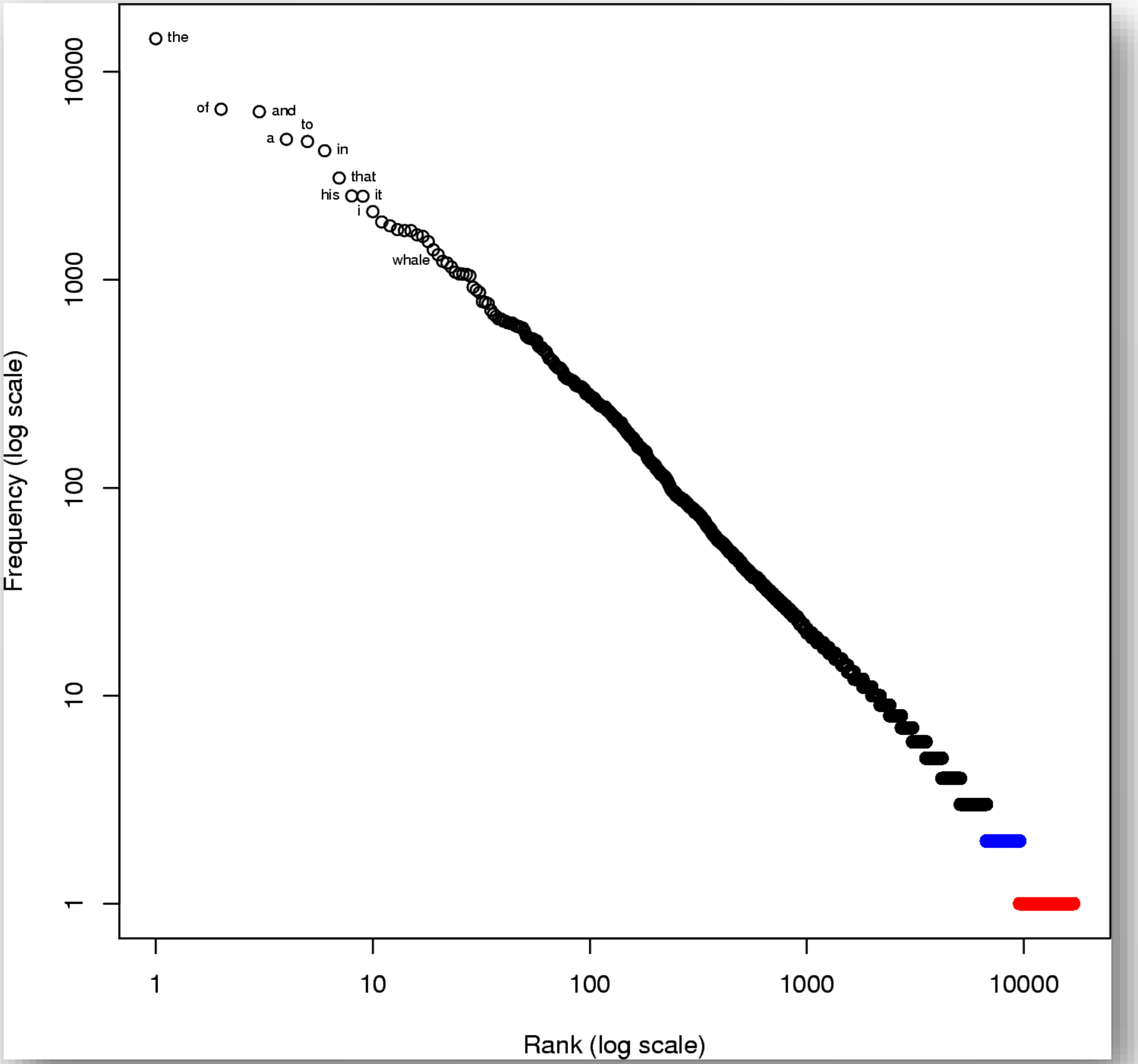
(This assumes N and $|V|$ grow independently.)

Zipf's Law on the Brown corpus



From Manning & Schütze

Zipf's Law on the novel *Moby Dick*



From Wikipedia

UNIVERSITY OF TORONTO

Zipf’s Law in perspective

- Zipf’s explanation for thisinvolved 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, e.g.
  - Yule’s Law: B = 1 +
    - *s:* probability of mutation becoming dominant in species
    - *g:* probability of mutation that expels species from genus
  - Pareto distributions
  - Champernowne’s Ergodic Wealth distribution
  - Mandelbrot’s (1961) monkey model.

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)

- There are a **lot** of infrequent words:

  English Top 31: 36%

  Top 150: 43%

  Top 256: 50%

  Hungarian Top 4096: 50% (why?)

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!

LANGUAGE MODELLING

Statistical modelling

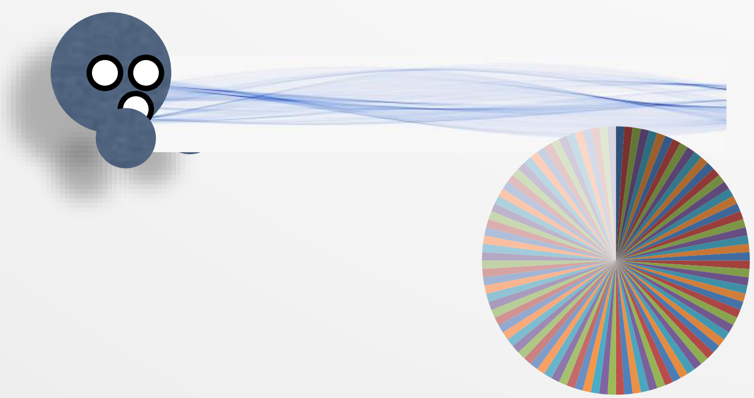
- Insofar as language can be modelled statistically, it might help to think of it in terms of dice.

Fair die

- Vocabulary: numbers
- Vocabulary size: 6

Language

- Vocabulary: words
- Vocabulary size: 2– 200,000



Learning probabilities

- What if the symbols are ***not*** equally likely?
  - We have to estimate the *bias* using training data.

Loaded die

- Observe many rolls of the die.
  - e.g.,

1,6,5,4,1,3,2,2,….

Language

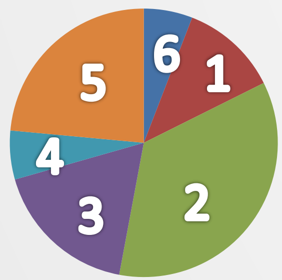
- Observe many words.
  - e.g.,

…and then I will…

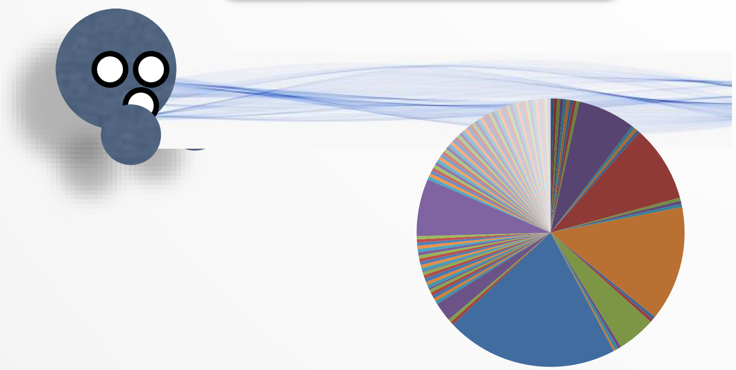
Training data

Training vs testing

Loaded die



Language



- So you've **learned** your **probabilities**.
  - Do they model **unseen** data from the **same** source well?

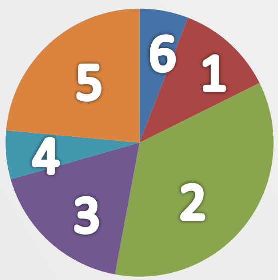
- **Keep rolling** the same dice.
- Do **sides** keep appearing in the **same proportion** as we expect?

- **Keep reading** words.
- Do **words** keep appearing in the **same proportion** as we expect?

Sequences with no dependencies

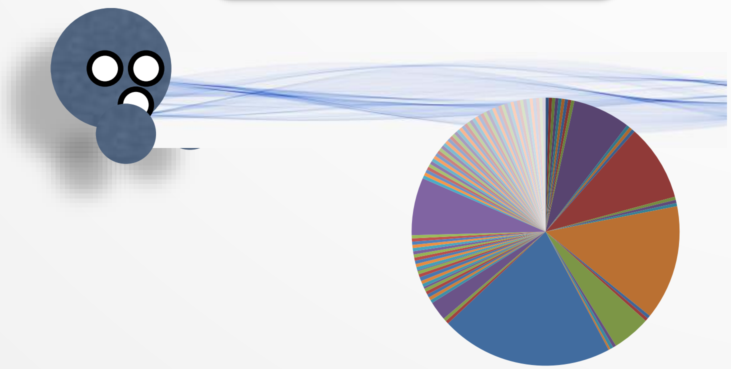
- If you ***ignore*** the past ***entirely***, the probability of a sequence is the product of **prior** probabilities.

Loaded die

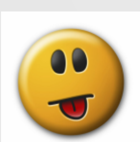


$$P(2,1,4) = P(2)P(1)P(4)$$

Language



$$P(\textit{the old car}) = P(\textit{the})P(\textit{old})P(\textit{car})$$



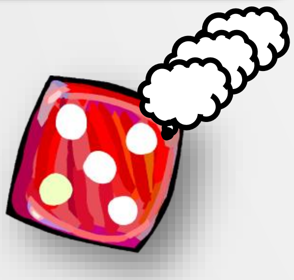
Language involves context. Ignoring that gives weird results, e.g.,

$$P(2,1,4) = P(2)P(1)P(4) \\ = P(2)P(4)P(1) = P(2,4,1)$$

$$P(\textit{the old car}) = P(\textit{the})P(\textit{old})P(\textit{car}) \\ = P(\textit{the})P(\textit{car})P(\textit{old}) \\ = P(\textit{the car old})$$

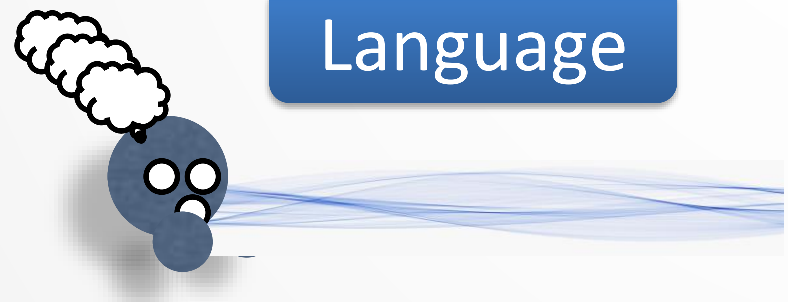
Sequences with full dependencies

Magic die
(with total memory)



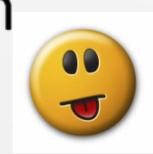
P(2,1,4)

Language



P(the,old,car)

- If you consider ***all*** of the past, you will **never** gather enough data in order to be **useful** in practice.
  - Imagine you've only seen the **Brown** corpus.
  - The sequence *'the old car'* **never appears** therein
  - $\therefore P(the\ old\ car) = 0$



UNIVERSITY OF TORONTO

Sequences with fewer dependencies?

Magic die
(with recent memory)

$$P(2,1,4) = P(2)P(1|2)P(4|1)$$

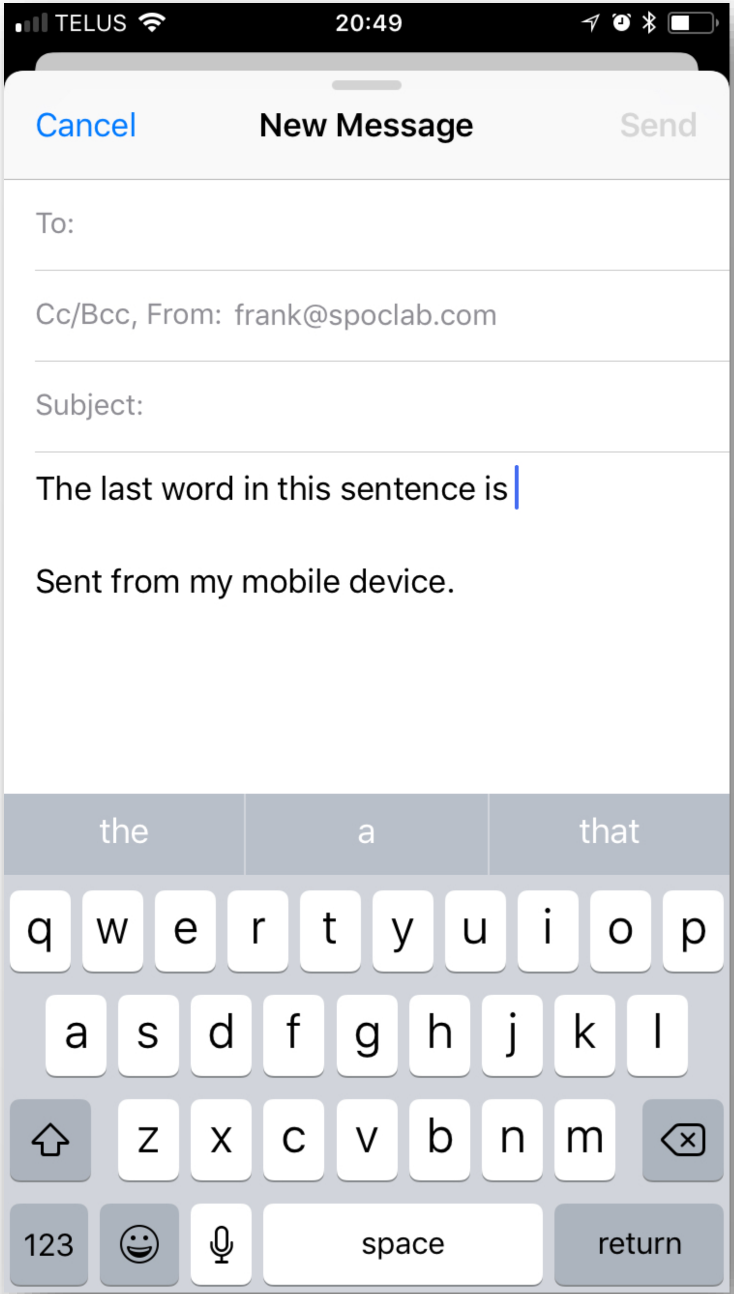
Language

$$P(the\ old\ car) = P(the)P(old|the) \cdot P(car|old)$$

- Only consider two words at a time…
  - Imagine you've only seen the Brown corpus.
  - The sequences *'the old'* & *'old car'* **do appear** therein!
  - $\boxed{P(old|the) > 0, P(car|old) > 0}\ \therefore P(the\ old\ car) > 0$
  - **Also**, $P(the\ old\ car) > P(the\ car\ old)$ 

Word prediction

- Guess the next word
- You can do quite well just with **limited extent**
  - E.g., $P(w_t \mid w_{t-1})$, just by counting (w_{t-1}, w_t) in a **representative** corpus



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*'?
  - Derived from bigrams (is,·)

Use of *N*-gram models

- Given the **probabilities** of *N*-grams, we can compute the **conditional probabilities** of possible subsequent words.

  - E.g., $P(is\ the) > P(is\ a)\ \therefore$
    $$P(the|is) > P(a|is)$$

    Then we would predict:

    'the last word in this sentence is ***the****.'*

(The last word in this sentence is missing.)

UNIVERSITY OF TORONTO

Language model usage

- Language models can **score** and **sort** sentences.
  e.g. P(*I like apples*) >> P(*I lick apples*)
  Commonly used to (re-)rank hypotheses in other tasks

- Infer properties about natural language
  e.g. P(*les pommes rouges*) > P(*les rouges pommes*)

- Infer embedding spaces

- Efficiently compress or repair text

- But how do we calculate P(…)?

The chain rule

- Recall,

$$P(A, B) = P(B|A)P(A) = P(A|B)P(B)$$

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

- This extends to longer sequences, e.g.,

$$P(A, B, C, D) = P(A)P(B|A)P(C|A, B)P(D|A, B, C)$$

- Or, in general,

$$P(w_1, w_2, \ldots, w_n) = P(w_1)P(w_2|w_1) \cdots P(w_n|w_1, w_2, \ldots, w_{n-1})$$

Very simple predictions

- Let's return to **word prediction**.
- We want to know the probability of the **next** word given the **previous** words in a sequence.

- We can **approximate** conditional probabilities by counting occurrences in large corpora of data.
  - E.g., $P(food \mid I\ like\ Chinese) = \dfrac{P(I\ like\ Chinese\ food)}{P(I\ like\ Chinese\ \cdot)}$

$$\approx \frac{Count(I\ like\ Chinese\ food)}{Count(I\ like\ Chinese)}$$

Problem with the chain rule

- There are **many** (∞?) possible sentences.
- In general, we **won't** have enough data to compute **reliable** statistics for **long** prefixes
  - E.g.,

$$P(pretty|I\ heard\ this\ guy\ talks\ too\ fast\ but\ at\ least\ his\ slides\ are) = \frac{P(I\ heard\ \ldots are\ pretty)}{P(I\ heard\ \ldots are)} = \frac{0}{0}$$

- How can we avoid $\{0, \infty\}$-probabilities?

Markov assumptions

1) **Limited extent**: assume each observation's dependence on history factors through a short recent history:

$$P(w_n \mid w_{1:(n-1)}) \approx P(w_n \mid w_{(n-L+1):(n-1)})$$

"Bigrams": $P(w_n \mid w_{1:(n-1)}) \approx P(w_n \mid w_{n-1})$

2) **Time invariance**

Berkeley Restaurant Project corpus

- Let's compute simple *N*-gram models of speech queries about restaurants in Berkeley California.
  - E.g.,
    - *can you tell me about any good cantonese restaurants close by*
    - *mid priced thai food is what i'm looking for*
    - *tell me about chez panisse*
    - *can you give me a listing of the kinds of food that are available*
    - *i'm looking for a good place to eat breakfast*
    - *when is caffe venezia open during the day*

UNIVERSITY OF TORONTO

Example bigram counts

- Out of 9222 sentences,
  - e.g., *"I want"* occurred 827 times

$Count(w_{t-1}, w_t)$		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

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

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

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

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

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

UNIVERSITY OF TORONTO

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

$P(w_t \vert w_{t-1})$	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

Bigram estimate of an unseen phrase

- We can **string** bigram probabilities together to estimate the probabilities of **whole sentences**.
  - We use the **start** (<s>) and **end** (</s>) tags here.

- E.g.,

$$P(<s>\ I\ want\ english\ food\ </s>) \approx$$
$$P(I \mid <s>)\ P(want \mid I) \cdot$$
$$\mathrm{P}(english \mid want)\ \mathrm{P}(food \mid english) \cdot$$
$$\mathrm{P}(</s> \mid food)$$
$$\approx 0.000031$$

- Where are the errors?

N-grams as linguistic knowledge

- Despite their simplicity, *N*-gram probabilities can **crudely** capture **interesting facts** about language and the world.

- E.g.,

$$P(english|want) = 0.0011$$
$$P(chinese|want) = 0.0065$$

World knowledge

$$P(to|want) = 0.66$$
$$P(eat|to) = 0.28$$
$$P(food|to) = 0$$

Syntax

$$P(i| <s>) = 0.25$$

Discourse

UNIVERSITY OF TORONTO

Probabilities of sentences

- The probability of a **sentence** s is defined as the **product** of the **conditional** probabilities of its ***N*-grams**:

$$P(s) = \prod_{i=2}^{t} P(w_i | w_{i-2} w_{i-1})$$ trigram

$$P(s) = \prod_{i=1}^{t} P(w_i | w_{i-1})$$ bigram

- *Which of these two models is better?*

Aside - are *N*-grams still relevant?

- Appropriately smoothed *N*-gram LMs: (Shareghi *et al*. 2019):
  - Are *invariably* <u>cheaper to train/query than neural LMs</u>
  - *Occasionally* outperform neural LMs
    - At least are a good baseline
  - *Usually* handle previously unseen tokens in a more principled (and fairer) way than neural LMs
- *N*-gram probabilities aren’t as deceptive to interpret
- *N*-grams are pervasively used in other tasks than LM
- Many neural language models do use limited extent.

EVALUATING LANGUAGE MODELS

Evaluating a language model

- How can we **quantify** the *goodness* of a model?
- How do we know whether one model is better than another?
  - There are 2 general ways of evaluating LMs:
    - **Extrinsic**: in terms of some external measure (this depends on some task or application).
    - **Intrinsic**: in terms of properties of the LM itself.

Extrinsic evaluation

- The **utility** of a **language model** is often determined *in situ* (i.e., in **practice**).
  - e.g.,
    1. **Alternately embed** LMs A and B into a **speech recognizer**.
    2. **Run** speech recognition using each model.
    3. **Compare** recognition rates between the system that uses LM A and the system that uses LM B.

Intrinsic evaluation

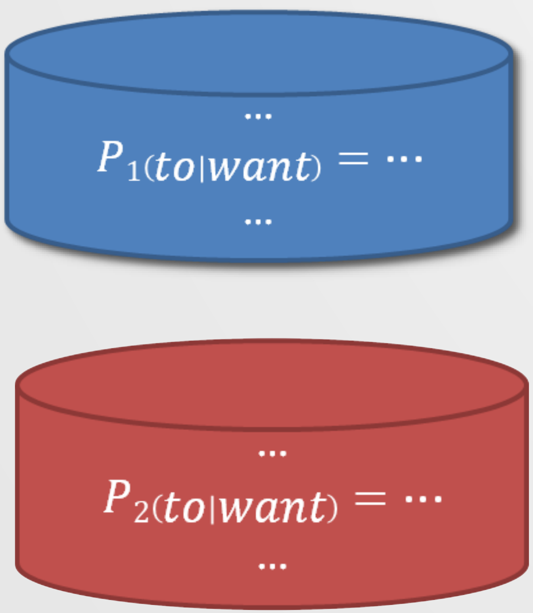
- To measure the **intrinsic value** of a language model, we first need to estimate the **probability of a corpus**, $P(C)$.
  - This will also let us **adjust/estimate** model **parameters** (e.g., $P(to|want)$) to maximize $P(Corpus)$.

- For a **corpus** of sentences, C, we sometimes make the assumption that the **sentences are conditionally independent**: $P(C) = \prod_i P(s_i)$

UNIVERSITY OF TORONTO

Intrinsic evaluation

- We **estimate** $P(\cdot)$ given a **particular** corpus, e.g., Brown.
  - A good model of the Brown corpus is one that makes Brown very likely *(even if that model is bad for other corpora).*



If

$$P_1(\text{Brown corpus}) \geq P_j(\text{Brown corpus}) \; \forall j$$

then

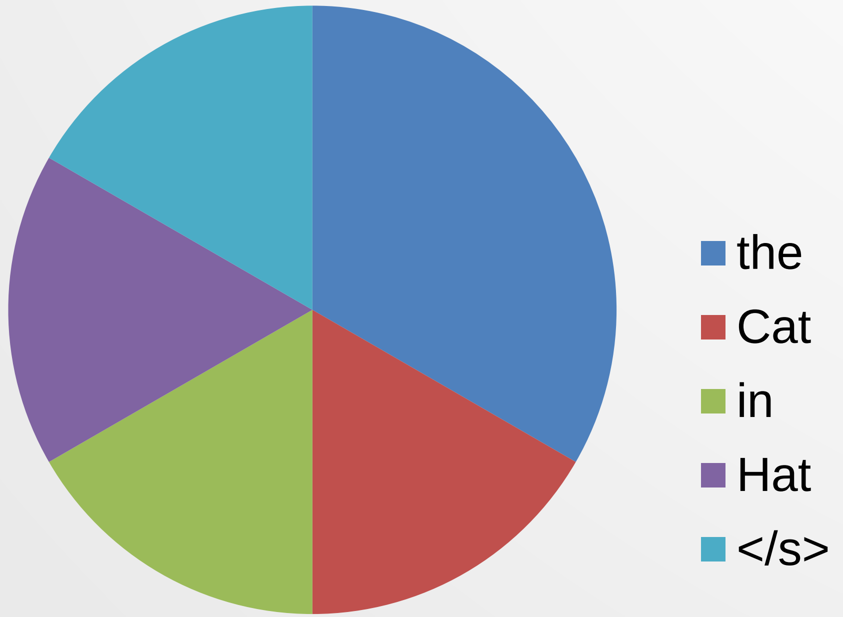
P_1 is the **best** model of the Brown corpus.

Shannon’s method

- We can use a language model to **generate** random sequences.
- We ought to see sequences that are **similar** to those we used for **training**.
- This approach is attributed to Claude Shannon.

Shannon’s method – unigrams

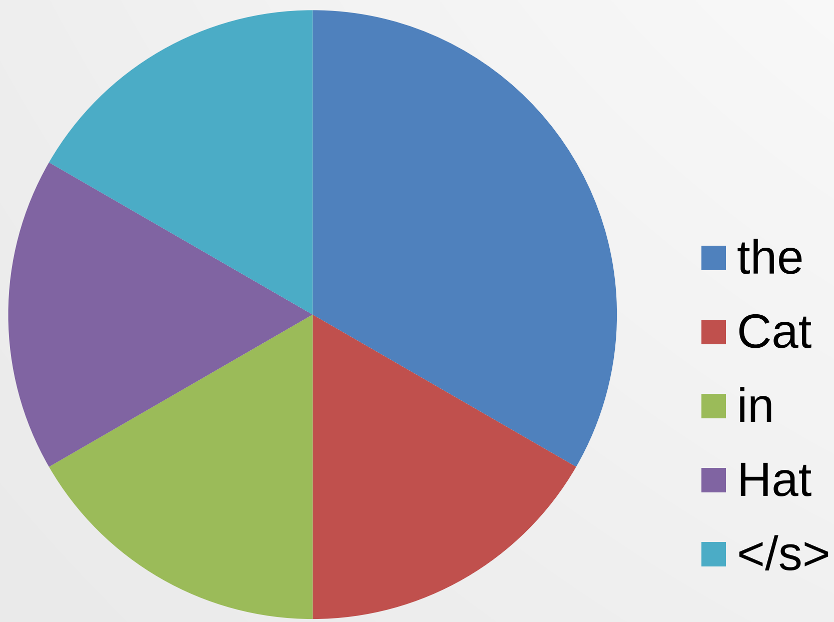
- **Sample** a model according to its probability.
  - For unigrams, keep picking tokens.
    - e.g., imagine throwing darts at this:



Problem with unigrams

- Unigrams give high probability to odd phrases.

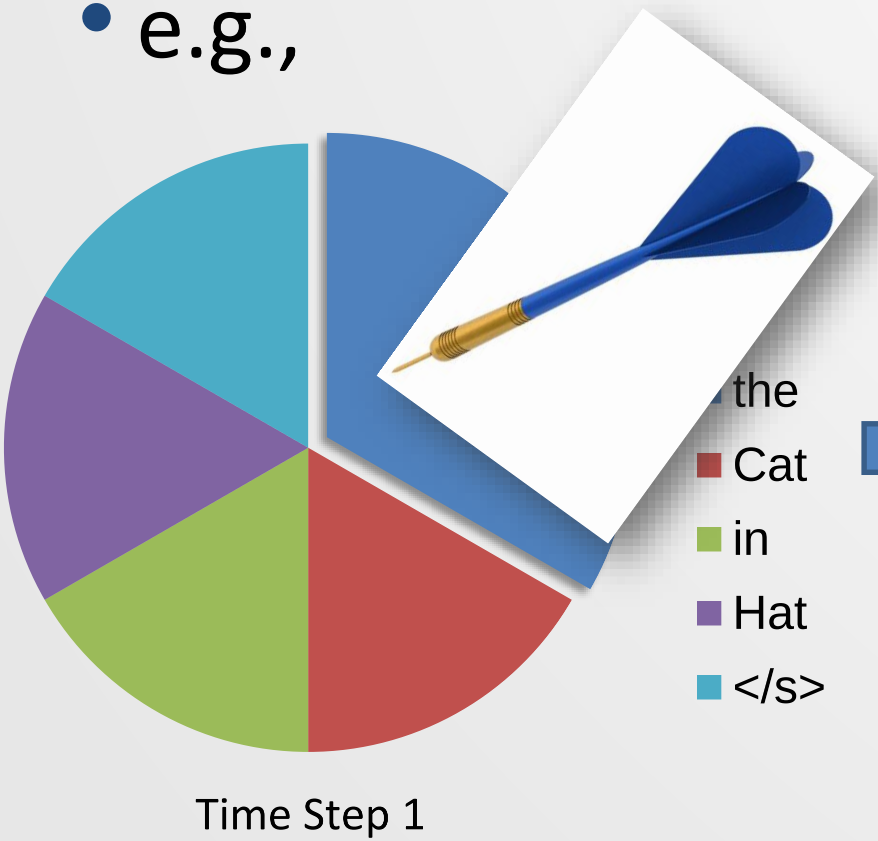
  e.g., $P(the\ the\ the\ the\ the\ </s>) = P(the)^5 \cdot P(</s>)$
  $> P(the\ Cat\ in\ the\ Hat\ </s>)$



UNIVERSITY OF TORONTO

Shannon's method – bigrams

- Bigrams have *fixed* context once that context has been sampled.
  - e.g.,



$$P(\cdot\,|the)$$

Shannon and the Wall Street Journal

Unigram	• Months the my and issue of year foreign new exchange’s September were recession exchange new endorsed a acquire to six executives.
Bigram	• Last December through the way to preserve the Hudson corporation N.B.E.C. Taylor would seem to complete the major central planners one point five percent of U.S.E. has already old M.X. corporation of living on information such as more frequently fishing to keep her.
Trigram	• They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions.

Shannon's method on Shakespeare

Unigram	• To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have • Hill he late speaks; or! A more to leg less first you enter • Are where exeunt and sighs have rise excellency took of.. Sleep knave we. Near; vile like.
Bigram	• What means, sir. I confess she? Then all sorts, he is trim, captain. • Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow. • What we, hat got so she that I rest and sent to scold and nature bankrupt nor the first gentleman?
Trigram	• Sweet prince, Falstaff shall die. Harry of Monmouth's grave. • This shall forbid it should be branded, if renown made it empty. • Indeed the duke; and had a very good friend.
Quadrigram	• King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. • Will you not tell me who I am? • It cannot be but so. • Indeed the short and the long. Marry. 'tis a noble Lepidus.

Shakespeare as a corpus

- 884,647 tokens, **vocabulary** of $V = 29{,}066$ types.
- Shakespeare produced about 300,000 bigram types out of $V^2 \approx 845M$ **possible** bigram types.
  - ∴ 99.96% of possible bigrams were **never** seen (i.e., they have 0 probability in the bigram table).

- **Quadrigrams** appear more **similar** to Shakespeare because, for **increasing context**, there are fewer possible next words, given the training data.
  - E.g., $P(Gloucester|seek\ the\ traitor) = 1$

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*.

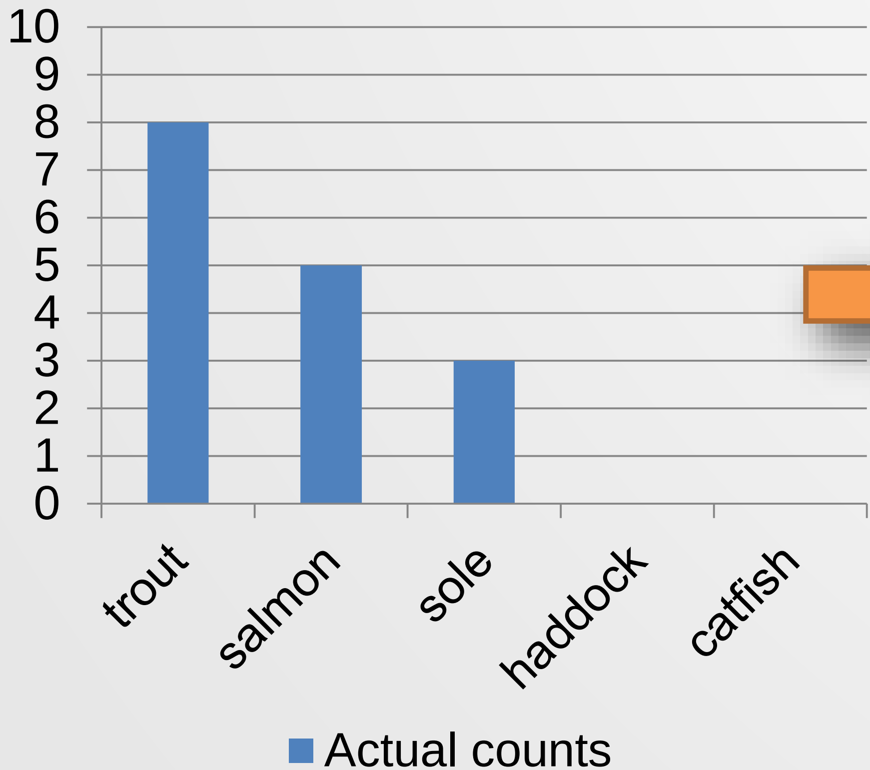
SMOOTHING

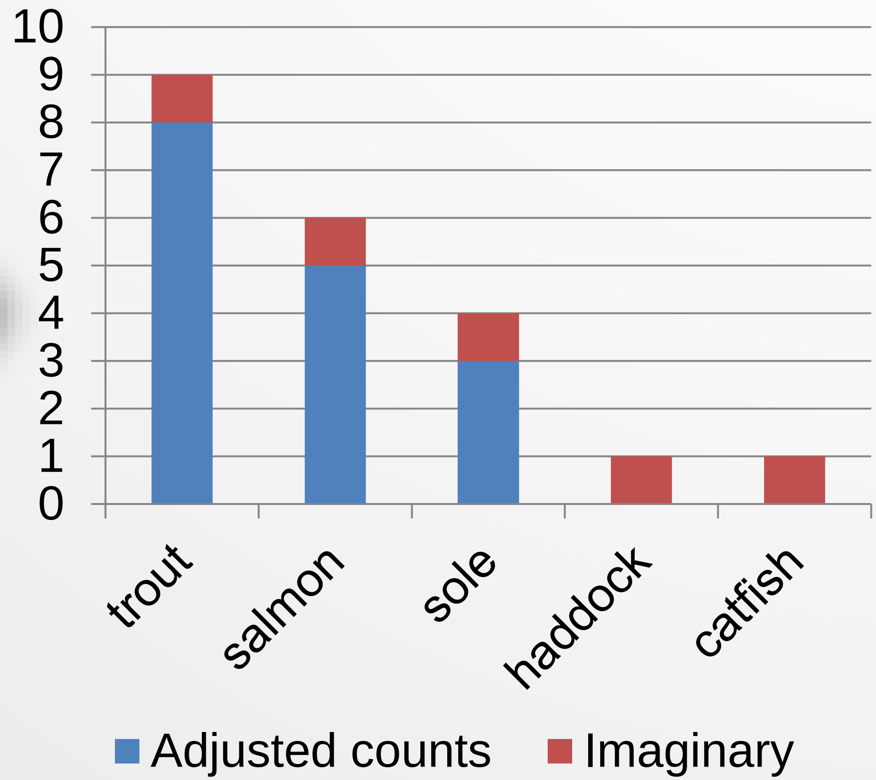
Zero probability in general

- 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 as redistribution

- Make the distribution more uniform.
- Move probability mass from ‘the rich’ towards ‘the poor’.



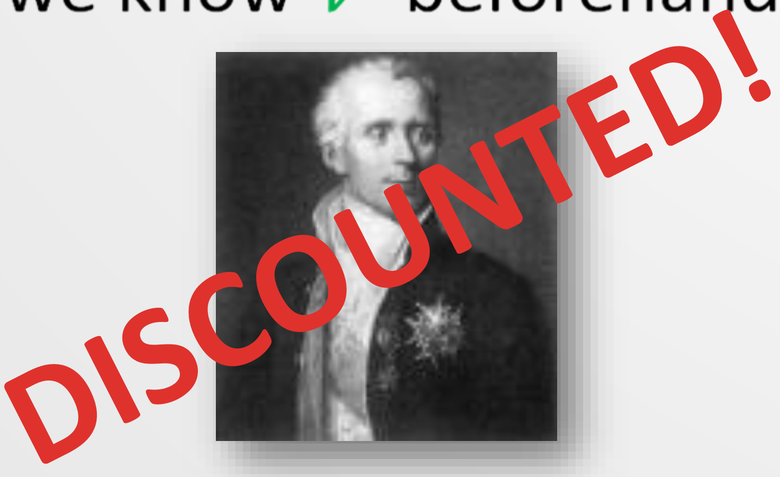


1. Add-1 smoothing

- 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 $\mathcal{V}$ beforehand.



1. Add-δ smoothing

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

- MLE : $P(w) = Count(w)/N$
- Add- δ estimate : $P_{add-\delta}(w) = \frac{Count(w)+\delta}{N+\delta\|\mathcal{V}\|}$

- Does this give a proper probability distribution? Yes:

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

$$= \frac{N+\delta\|\mathcal{V}\|}{N+\delta\|\mathcal{V}\|} = 1$$

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

Is there another way?

- Choice of δ is ad-hoc
- Has Zipf taught us *nothing*?
  - **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 '*zenzizenzizenzic*', I will have seen '*the*' a lot more than once more.

2. Good-Turing for N-grams?

- Q: What happens when:

  C(McGill genius) = C(McGill brainbox) = 0,

  and we smooth bigrams using Good-Turing?
- A: P(genius | McGill) = P(brainbox | McGill) > 0
- But really, we should expect

  P(genius | McGill) > P(brainbox | McGill)

  context-independently, because *genius* is simply more common than *brainbox.*
- So we would need to combine this approach with something else.

Readings

- Chen & Goodman (1998) "An Empirical Study of Smoothing Techniques for Language Modeling," Harvard Computer Science Technical Report
- Jurafsky & Martin (*2nd ed*): 4.1-4.7
- Manning & Schütze: 6.1-6.2.2, 6.2.5, 6.3
- Shareghi *et al* (2019): https://www.aclweb.org/anthology/N19-1417.pdf (From the aside – completely optional)