SCAN ME

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

Overview

- (Statistical) language models (*n*-gram models)
  - Counting
  - Data
  - Definitions
  - Evaluations
  - Distributions
  - 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.

Statistics: what are we counting?

- Statistical language models are based on simple counting.
- *What are we counting?*

> *First, we shape our tools and thereafter our tools shape us.*

- **Tokens**: *n.pl.* **instances** of words or punctuation (13).

- **Types**: *n.pl.* '**kinds**' of words or punctuation (10).

Confounding factors

- Are the following pairs **one** type or **two**?
  - *(run, runs)* (verb conjugation)
  - *(happy, happily)* (adjective vs. adverb)
  - *(fra$^{(1)}$gment, fragme$^{(1)}$nt)* (spoken stress)
  - *(realize, realise)* (spelling)
  - *(We, we)* (capitalization)

- How do we count **speech disfluencies**?
    - e.g., *I* ***uh*** ***main-****mainly do data processing*
  - Answer: It depends on your task.
    - e.g., if you're doing summarization, you usually don't care about '*uh*'.

UNIVERSITY OF TORONTO

Does it matter how we count things?

- Answer: See lecture on **feature extraction**.
- Preview: *yes, it matters*...(sometimes)
  - E.g., to **diagnose Alzheimer's disease** from a patient's speech, you may want to measure:
    - Excessive **pauses** (disfluencies),
    - Excessive word **type** repetition, and
    - Simplistic or **short** sentences.
- *Where* do we count things?

Corpora

- **Corpus**: *n.* A body of language data of a particular sort (*pl.* **corpora**).
- Most **useful** corpora occur **naturally**.
  - e.g., newspaper articles, telephone conversations, multilingual transcripts of the United Nations, tweets.
- We use corpora to gather statistics.
  - More is better (typically between 10M and 1T words).
  - Be aware of bias.
- Examples: *Canadian Hansards*, *Project Gutenberg* (e-books), web crawls (*Google N-Gram, Common Crawl)*

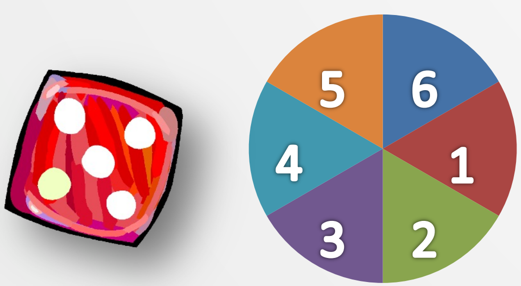
UNIVERSITY OF TORONTO

Statistical modelling

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

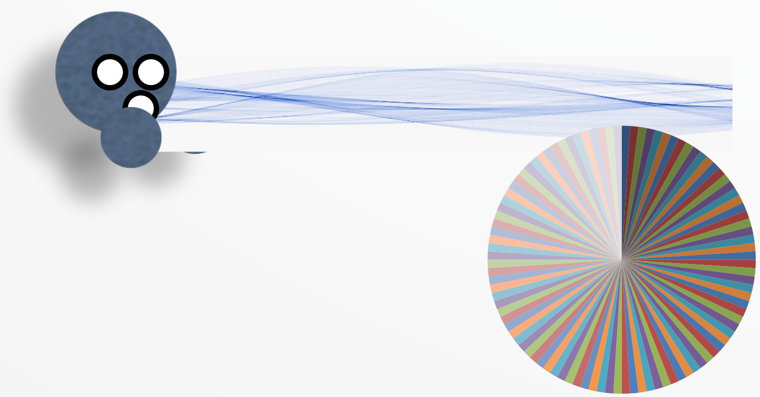
Fair die

- Vocabulary: numbers
- Vocabulary size: 6



Language

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



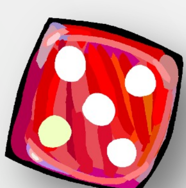
Learning probabilities

- 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

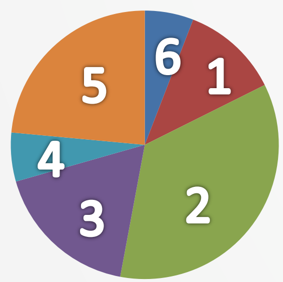
- Observe many words.
  - e.g.,

…and then I will…

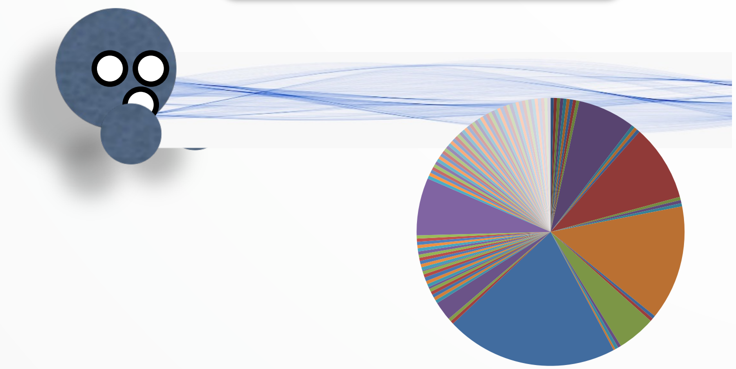
Training data

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?

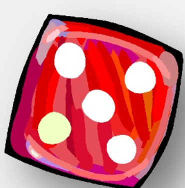
- **Keep reading** words.
- Do **words** keep appearing in the **same proportion** as we expect?

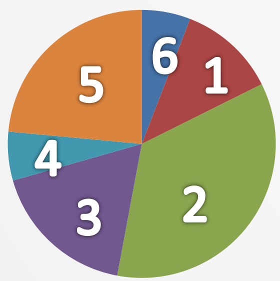
UNIVERSITY OF TORONTO

Sequences with no dependencies

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

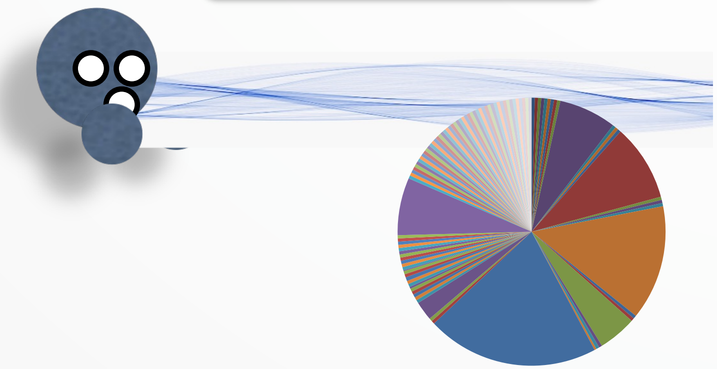
Loaded die



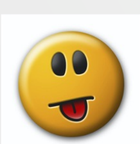


$$P(2,1,4) = P(2)P(1)P(4)$$

Language



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



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

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

$$P(the\ old\ car) = P(the)P(old)P(car) = P(the)P(car)P(old) = P(the\ car\ old)$$ ✘

Sequences with full dependencies

Magic die
(with total memory)

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

Language

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

- 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
  - $P(car|the\ old) = 0 \therefore P(the\ old\ car) = 0$

Sequences with fewer dependencies?

Magic die
(with recent memory)

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

Language

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

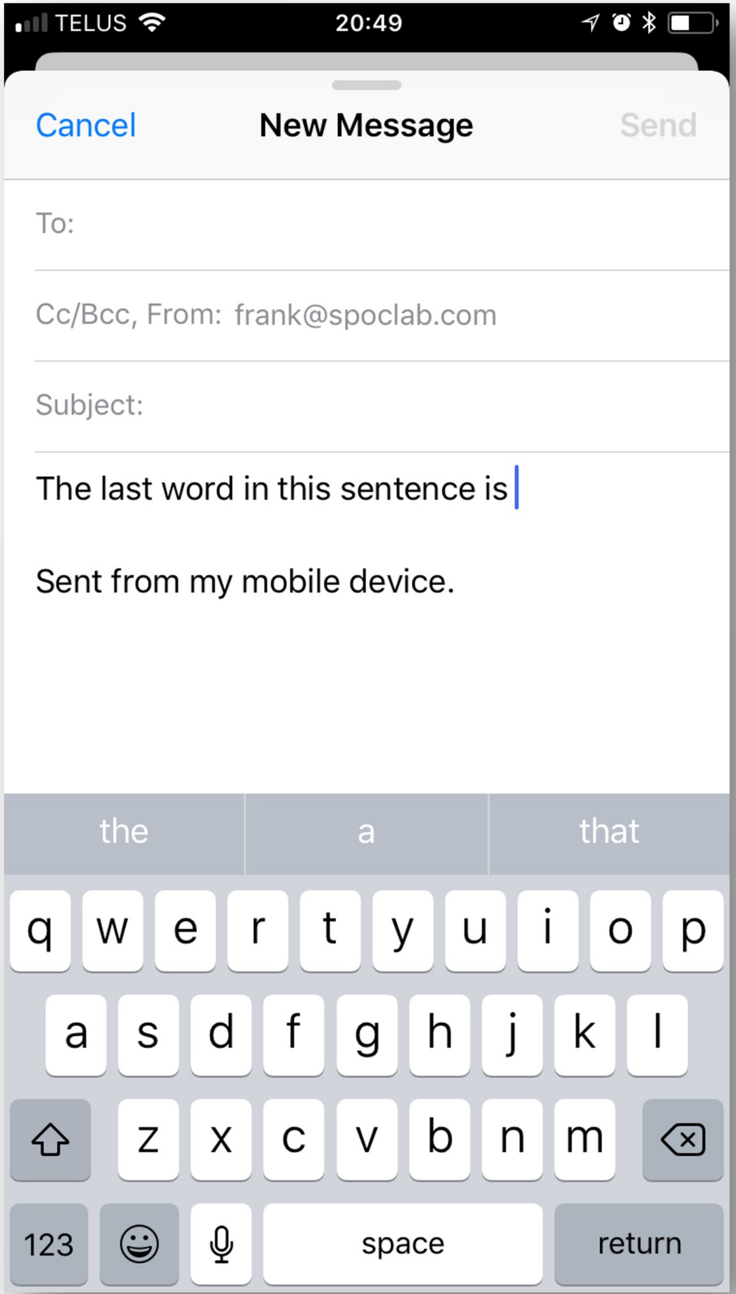
- Only consider two words at a time...
  - Imagine you've only seen the Brown corpus.
  - The sequences *'the old'* & *'old car'* **do appear** therein!
  - $P(old|the) > 0, P(car|old) > 0 \therefore P(the\ old\ car) > 0$
  - **Also**, $P(the\ old\ car) > P(the\ car\ old)$ 

SCAN ME

LANGUAGE MODELS

Word prediction

- Guess the next word...
- **Spoilers** You can do quite well by **counting** how often certain **tokens** occur given their **contexts**.
  - E.g., estimate $P(w_t|w_{t-1})$ from count of (w_{t-1}, w_t) in corpus



UNIVERSITY OF TORONTO

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.)

Language models

- **Language model**: *n.* The statistical model of a language (obviously).
  - e.g., probabilities of words in an *ordered* sequence.

    i.e., $P(w_1, w_2, \ldots, w_n)$

- Word prediction is at the heart of language modelling.

- What do we **do** with a language model?

Language model usage

- Language models can **score** and **sort** sentences.
  - e.g., $P(I\ like\ apples) \gg P(I\ lick\ apples)$
  - Commonly used to (re-)rank hypotheses in other tasks
- Infer properties of natural language
  - e.g., $P(les\ pommes\ rouges) > P(les\ rouges\ pommes)$
  - Embedding spaces
- Efficiently compress text
- *How do we calculate* $P(\ldots)$*?*

Frequency statistics

- **Term count ($Count$)** of term w in corpus C is the number of tokens of term w in C.

$$Count(w, C)$$

- **Relative frequency (F_C)** is defined **relative** to the **total number of tokens** in the corpus, $||C||$.

$$F_C(w) = \frac{Count(w, C)}{||C||}$$

  - In theory, $\lim_{||C|| \to \infty} F_C(w) = P(w)$. (the "*frequentist* view")

UNIVERSITY OF TORONTO

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,

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

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?

Independence!

- We can **simplify** things if we're willing to **break** from the **distant** past and focus on **recent** history.
  - e.g.,

$$P(pretty|I\ heard\ this\ guy\ talks\ too\ fast\ but\ at\ least\ his\ slides\ are)$$
$$\approx P(pretty|slides\ are)$$
$$\approx P(pretty|are)$$

- I.e., we assume **statistical independence**.

UNIVERSITY OF TORONTO

Markov assumption

- Assume each observation only depends on **a short linear history** of length *L*.

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

- **Bigram** version:

$$P(w_n|w_{1:(n-1)}) \approx P(w_n|w_{n-1})$$

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*

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

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

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

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

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

$P(w_t \mid w_{t-1})$	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 probability of **whole sentences**.
  - We use the **start** (<s>) and **end** (</s>) tags here.

- E.g.,

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

UNIVERSITY OF TORONTO

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

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?*

UNIVERSITY OF TORONTO

Aside - are *N*-grams still relevant?

- Appropriately smoothed *N*-gram LMs:
  (Shareghi *et al*. 2019):
  - Are *often* <u>cheaper to train/query than neural LMs</u>
  - Are interpolated with neural LMs to *often* achieve state-of-the-art performance
  - *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 are interpretable
- Convenient

SCAN ME

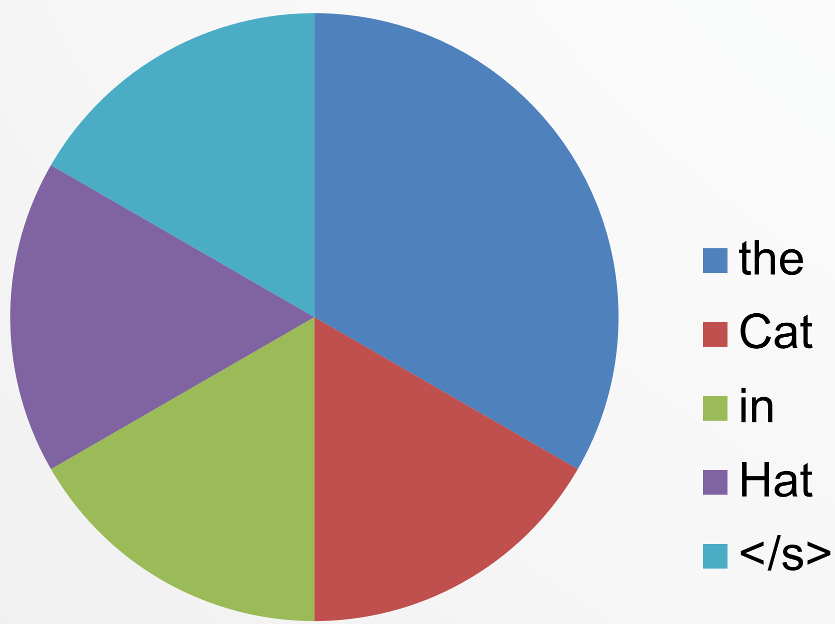
EVALUATING LANGUAGE MODELS

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:

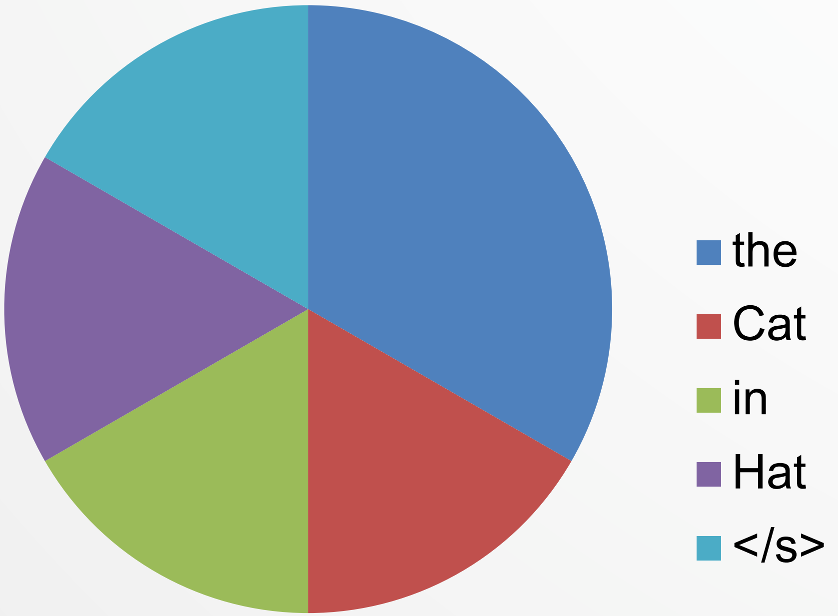


UNIVERSITY OF TORONTO

Problem with unigrams

- Unigrams give high probability to odd phrases.

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

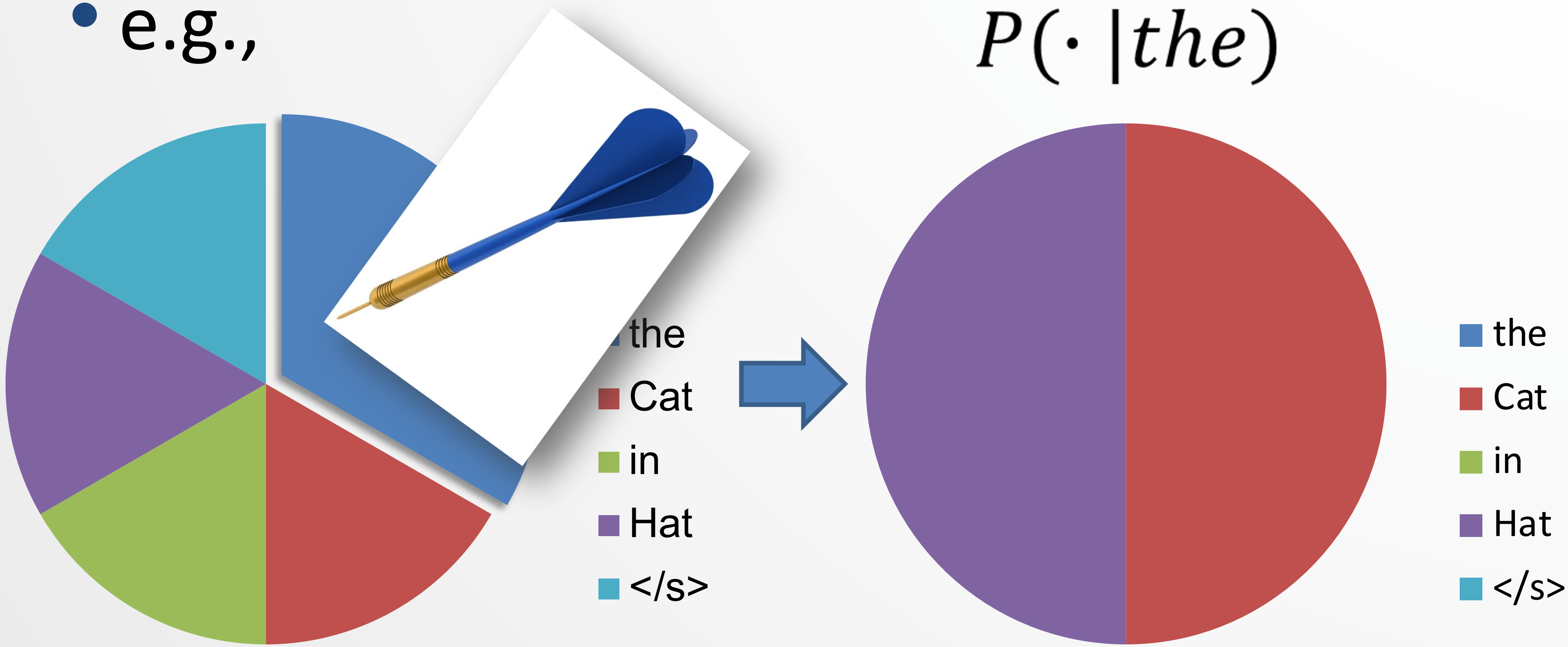


UNIVERSITY OF TORONTO

Shannon's method – bigrams

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

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



Time Step 1

Time Step 2

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., $P(Gloucester|seek\ the\ traitor) = 1$

UNIVERSITY OF TORONTO

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

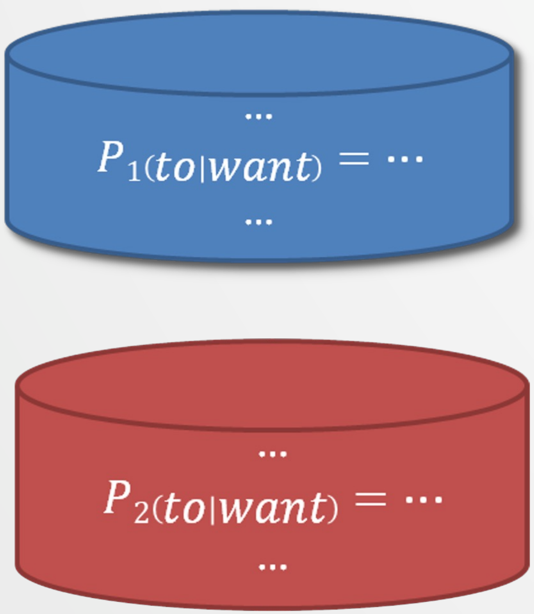
- To measure the **intrinsic value** of a language model, we first need to estimate the **probability of a corpus**, $P(C)$.
  - This will also let us **adjust/estimate** model **parameters** (e.g., $P(to|want)$) to maximize $P(Corpus)$.

- 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.

Maximum likelihood estimate

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

$$\theta^* = \text{argmax}_\theta L_M(\theta|T), \quad L_M(\theta|T) = P_{M(\theta)}(T)$$

  - e.g., $\boldsymbol{T}$ is the Brown corpus,
    $\boldsymbol{M}$ is the bigram and unigram tables
    $\boldsymbol{\theta}_{(to|want)}$ is $P(to|want)$.

- In fact, we have been doing MLE, within the *N*-gram context, **all along** with our simple counting*

*(assuming an end-of-sentence token)

MLE https://www.youtube.com/watch?v=XepXtl9YKwc
Probability versus Likelihood https://www.youtube.com/watch?v=pYxNSUDSFH4

UNIVERSITY OF TORONTO

Perplexity

- Perplexity corp. C, $PP(C) = 2^{-\left(\frac{\log_2 P(C)}{\|C\|}\right)} = P(C)^{-1/\|C\|}$
- **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**

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

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

- Minimizing perplexity ≡ maximizing probability of corpus

Perplexity as an evaluation metric

- **Lower** perplexity → a **better** model.
  - (more on this in the section on information theory)

- e.g., splitting WSJ corpus into a 38M word training set and a 1.5M word test set gives:

N-gram order	Unigram	Bigram	Trigram
Perplexity	962	170	109

UNIVERSITY OF TORONTO

Modelling language

- So far, we've modelled language as a **surface phenomenon** using only our **observations** (i.e., words).
- Language is hugely **complex** and involves **hidden** structure (recall: syntax, semantics, pragmatics).
- A '**true**' model of language would take into account **all** those things and the proper **relations** between them.
- Our first **hint** of modelling hidden structure will come with uncovering grammatical roles (i.e., **parts-of-speech**)

SCAN ME

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 ~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 *uni*gram, but still miss many *bi*grams:

	I	want	to	eat	Chinese	food	lunch	spend
Unigram counts:	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***.

UNIVERSITY OF TORONTO

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!

Ranking words in *Tom Sawyer*

- Rank word types in order of decreasing frequency.

Word	Freq. (f)	Rank (r)	$f \cdot r$
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. (f)	Rank (r)	$f \cdot r$
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, $f \cdot r$ is very consistent!

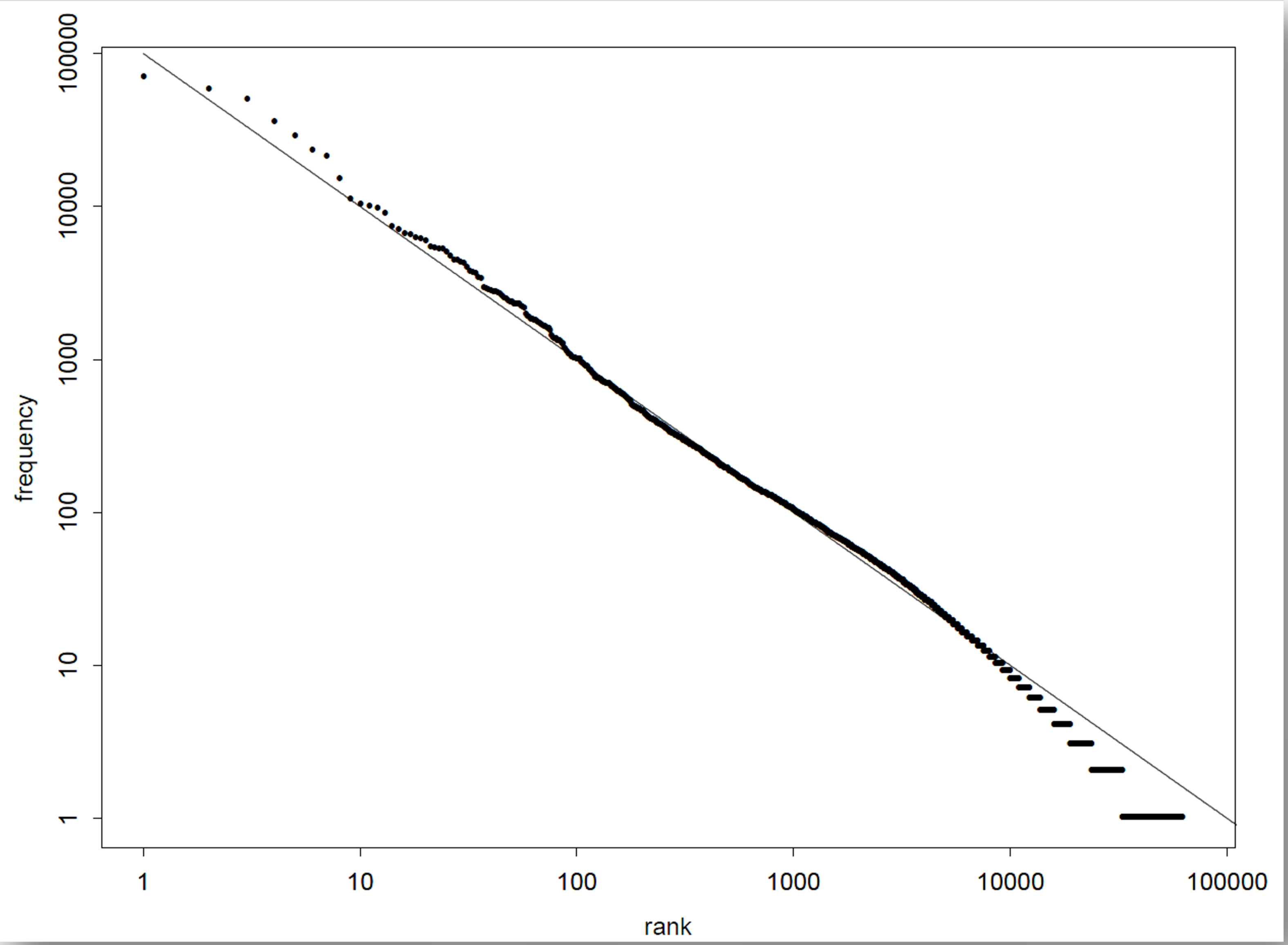
Zipf's Law

- 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.

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

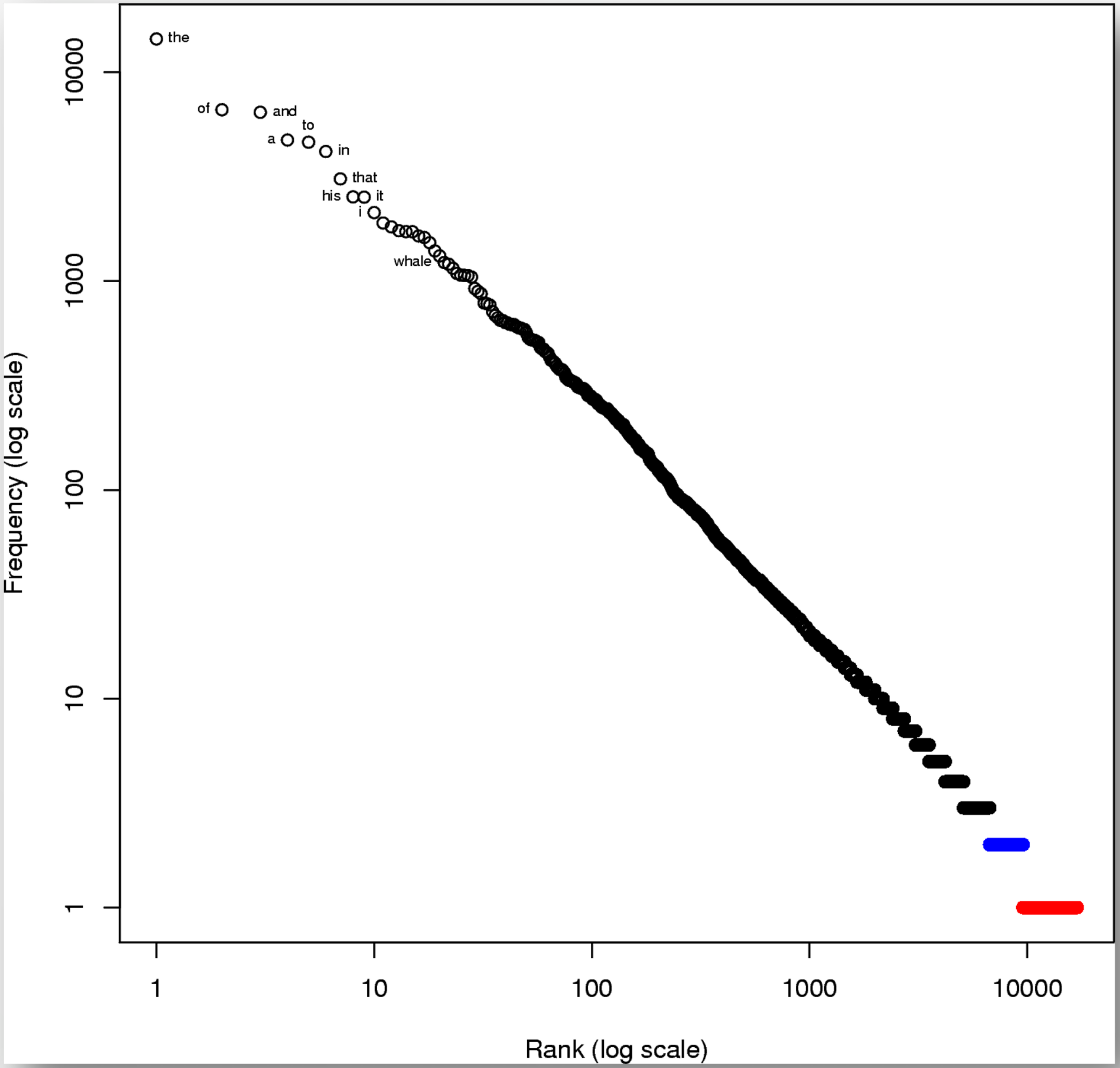
(*) This does not make it true.

Zipf's Law on the Brown corpus



From Manning & Schütze

Zipf's Law on the novel *Moby Dick*



From Wikipedia

UNIVERSITY OF TORONTO

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.

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)

UNIVERSITY OF TORONTO

SCAN ME

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.

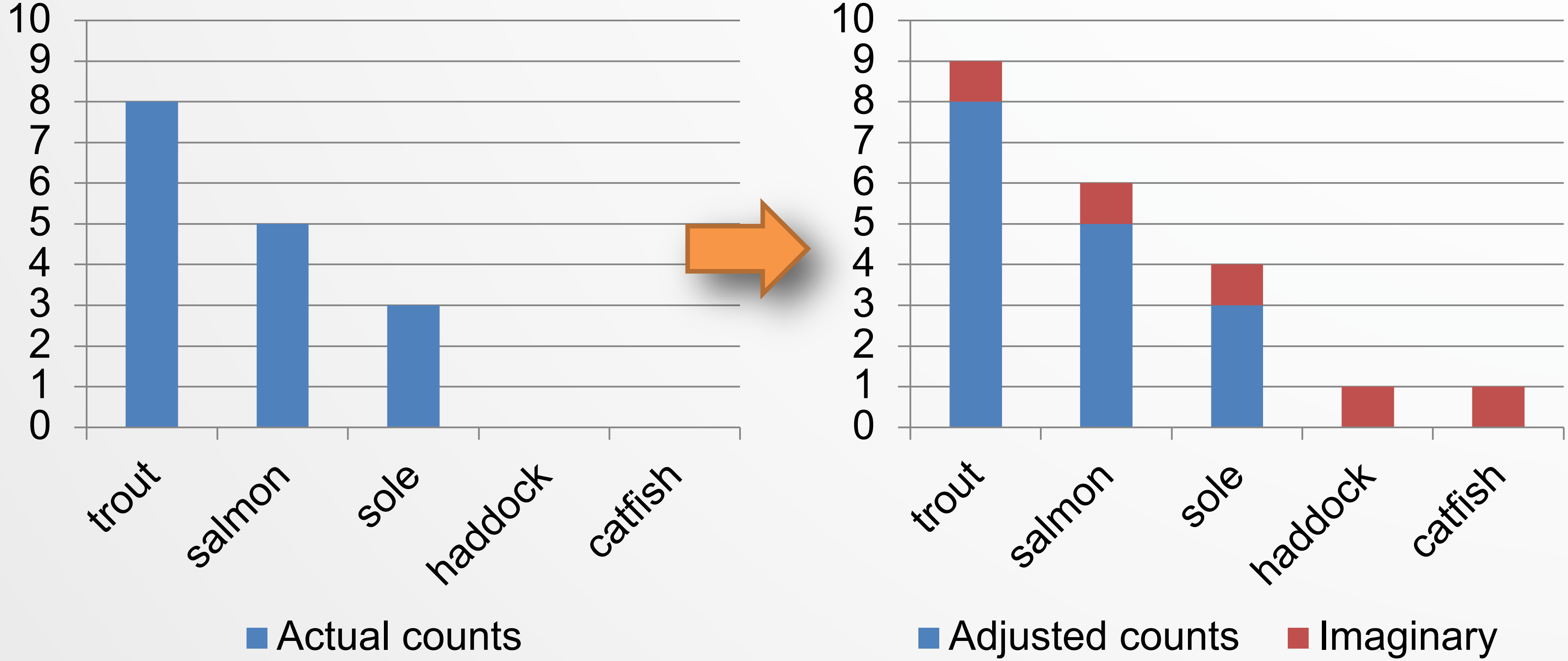
- If we have no way to determine the distribution of *unseen N*-grams, how can we estimate them?

Smoothing mechanisms

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

Smoothing as redistribution

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



UNIVERSITY OF TORONTO

1. Add-1 smoothing ("Laplace discounting")

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

- MLE : $P(w) = Count(w)/N$
- Laplace estimate : $P_{Lap}(w) = \frac{Count(w)+1}{N+\|\mathcal{V}\|}$

- Does this give a proper probability distribution? Yes:

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

1. Add-1 smoothing for bigrams

- Same principle for bigrams:

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

- 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

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

$P(w_t \mid w_{t-1})$	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 $\mathcal{V}$ 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.

UNIVERSITY OF TORONTO

2. Good-Turing

- Define N_c as the number of *N*-grams that occur *c* times.
  - "Count of counts"

Word frequency	# of words (i.e., unigrams) with that frequency
1	
2	
3	
…	…

(from *Tom Sawyer*)

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

Good Turing https://www.youtube.com/watch?v=GwP8gKa-ij8&list=PL8FFE3F391203C98C&index=18

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*'?
  - $P(soccer) = 1/23$

- 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., $P_{GT}(soccer) < 1/23$).

MLE https://www.youtube.com/watch?v=XepXtl9YKwc

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: $p = 1/23$
  - $c^*(soccer) = 2 \cdot \frac{N_2}{N_1}$
    $= 2 \cdot 1/3$
  - $P^*_{GT}(soccer) = \left(\frac{2}{3}\right)/23$

UNIVERSITY OF TORONTO

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 $c^* = \frac{(c+1)E[N_{c+1}]}{E[N_c]}$.
  - Uses Expectation-Maximization (method used 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 $Count(McGill\ genius) = 0$ and $Count(McGill\ brainbox) = 0$, and we smooth bigrams?
- A: $P(genius|McGill) = P(brainbox|McGill)$
  - But we'd expect $P(genius|McGill) > P(brainbox|McGill)$ (context notwithstanding) because '*genius*' is a more common word than '*brainbox*'*).*

- The solution may be to **combine** bigram and unigram models...

UNIVERSITY OF TORONTO

BONUS: Extra

3. 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
  - Expectation maximization

Interpolation https://www.youtube.com/watch?v=HInzCyXqgsI&list=PL8FFE3F391203C98C&index=17

4. Absolute discounting

- 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

- $\lambda_{w_{t-n+1:t-1}}$ are chosen s.t. $\sum_{w_t} P_{abs}(w_t | \ldots) = 1$
- You *can* learn δ using held-out data.

4. Why absolute discounting?

- Both simple interpolation and absolute discounting redistribute probability mass, why absolute discounting?
- Compare GT counts to observed counts on this database:

	0	1	2	3	4	5	6	7	8	9
	0.0000270	0.446	1.26	2.24	3.24	4.22	5.19	6.21	7.24	8.25

AP newswire, J&M 2nd Ed.

- As c increases, $(c - c^*) \rightarrow 0.75$. Good δ!
- Similar trend observed when comparing counts of training set (c) vs. held-out set ($\approx c^*$)

5. 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 ∴ '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'.

UNIVERSITY OF TORONTO
Kneser-Ney Smoothing https://www.youtube.com/watch?v=ody1ysUTD7o&list=PL8FFE3F391203C98C&index=19

5. Kneser-Ney smoothing

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

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

$$N_{1+}(\bullet\bullet) = \sum_{w_i} N_{1+}(\bullet w_i)$$ ←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:

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

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

5. Modified Kneser-Ney smoothing

- Use different absolute discounts δ depending on the n-gram count s.t. $C(w_{t-n+1:t}) \geq \delta_{C(w_{t-n+1:t})} \geq 0$

$$P_{MKN}(w_t|w_{t-n+1:t-1}) = \frac{C(w_{t-n+1:t}) - \delta_{C(w_{t-n+1:t})}}{\sum_{w_t} C(w_{t-n+1:t})} + (1 - \lambda_{w_{t-n+1:t-1}})P_{MKN}(w_t|w_{t-n+2:t-1})$$

- $\delta_{C(w_{t-n+1:t})}$ could be learned or approximated, usually aggregated for counts above 3
- λ chosen to sum to one

	0	1	2	3
	0.0000270	0.446	1.26	2.24

Smoothing over smoothing

- Modified Kneser-Ney is arguably the most popular choice for *n*-gram language modelling
  - Popular open-source toolkits like *KenLM, SRILM,* and *IRSTLM* all implement it
- New smoothing methods are occasionally published
  - Huang et al., Interspeech 2020
- While *n*-gram LMs are still around, most interest in language modelling research has shifted to neural networks
- We will discuss neural language modelling a few weeks from now

UNIVERSITY OF TORONTO

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)