Ngrams
+
Smoothing

Hit record

UROP

Undergraduate Fellowships for Clinical and Translational Research (CCTR)

The VCU Center for Clinical and Translational Research (CCTR) will fund undergraduate research fellowship awards for clinical translational research projects focused on human health and mentored by a VCU faculty member. A clinical translational research project is one that aims to translate scientific discoveries into improved human health and wellness. Successful proposals must discuss how the project will increase the student researcher's knowledge, skills and experience while simultaneously attempting to advance human health through clinical research. Each fellowship award includes $1500 in funding for the student and $500 for the faculty mentor. Eligibility: Undergraduates pursuing research in any discipline(s). For details and to apply visit: https://provost.vcu.edu/initiatives/urop/cctr/

Undergraduate Fellowships for Community Engaged Research (CEnR)

The Center for Community Engagement and Impact will fund undergraduate community-engaged research fellowship awards for research projects mentored by VCU faculty and carried out in collaboration with a community partner. Proposals for this fellowship should include a community-engaged research project that creates and disseminates knowledge or creative expression with the goal of contributing to the discipline and strengthening the well-being of the community. Each fellowship award includes $1500 in funding for the student, $500 for the faculty mentor, and $500 for the community partner. Eligibility: Undergraduates pursuing research in any discipline(s). For details and to apply visit: https://provost.vcu.edu/initiatives/urop/community/

Undergraduate Fellowships for Inclusion, Inquiry, and Innovation (iCubed)

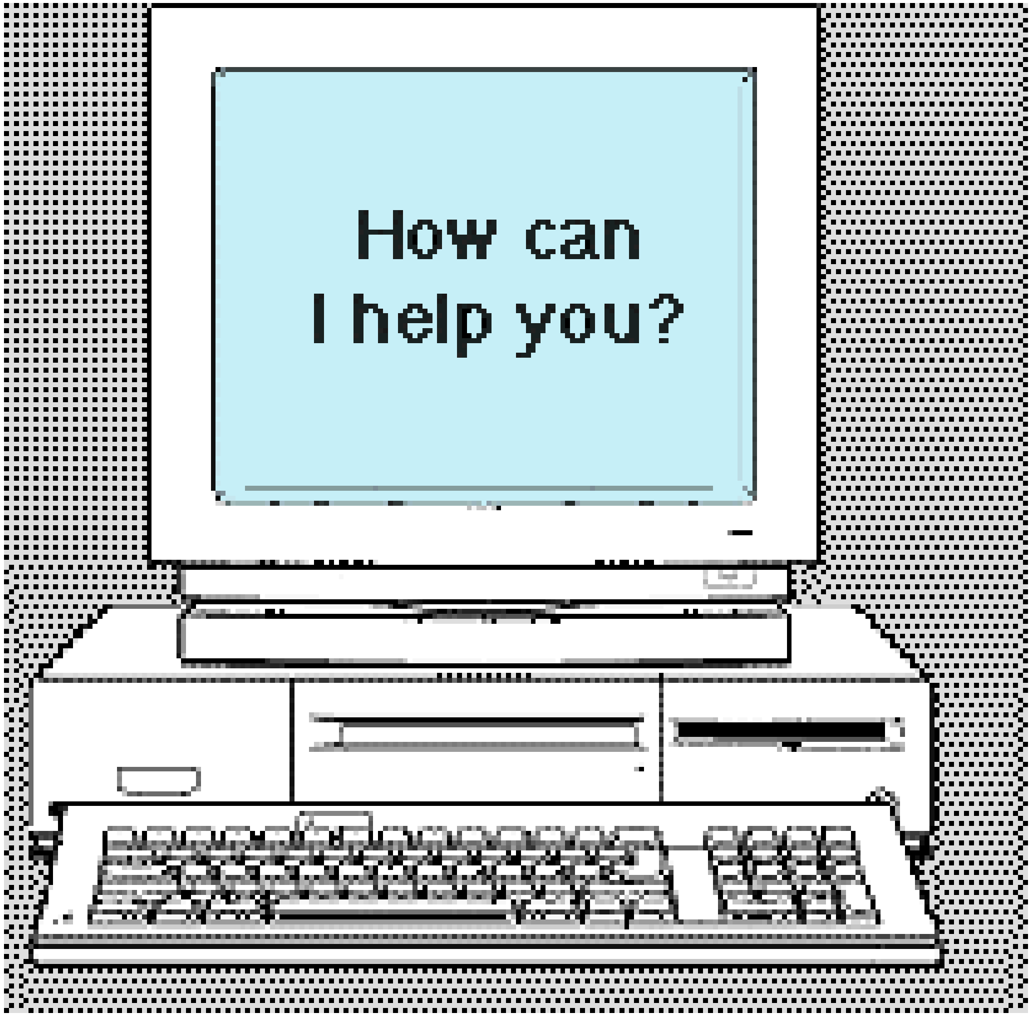
The Institute for Inclusion, Inquiry, and Innovation (iCubed) will fund research fellowship awards for projects mentored by VCU faculty who fall within the eight iCubed Cores: 1. Oral Health Equity 2. Sustainable Food Access 3. Intersections in the Lives of LGBTQIA+ Communities 4. Urban Education and Family 5. Disrupting Criminalization in Education 6. Health and Wellness in Aging Populations 7. Racial Equity, Arts, and Culture 8. Culture, Race, and Health. Each fellowship award includes $1500 in funding for the student and $500 for the faculty mentor. Eligibility: Work-Study eligible undergraduates pursuing research in any discipline(s) which aligns with the eight iCubed Cores. For details and to apply visit: https://provost.vcu.edu/initiatives/urop/icubed/

Undergraduate Research and Creative Scholarship Summer Fellowships (UROP)

The Undergraduate Research Opportunities Program (UROP) will fund a variety of undergraduate student fellowship awards for projects in any discipline, mentored by VCU faculty. Successful student applicants will receive a cash stipend of $1,500 and $500 for the faculty mentor. Applicants must submit an online application no later than March 10, 2021 for review. Eligibility: Undergraduates pursuing research in any discipline(s). For details and to apply visit: https://provost.vcu.edu/initiatives/urop/urop-fellowship/

Citations

/[Qq]uestions? (with|about) [Ee]liza?



Programming assignment rubric

1) Overall comment at start of program – a clear and concise introduction that has three components : 1) describe the problem to be solved well enough so that someone not familiar with our class could understand, 2) give actual examples of program input and output, along with usage instructions, and 3) describe the algorithm you have used to solve the problem, specified in a stepwise or point by point fashion. Your introduction should also include identifying information (your name, date, etc.)

________ 2 points max.

2) Detailed comments throughout code that fully explain details of algorithm. Implementation must work as described and solve problem correctly to get credit for detailed comments.

________ 3 points max.

3) Correct results automatically produced by running the program as specified in the assignment.

________ 5 points max.

Other comments may be found on the back – please turn over to check.

Ngram Modeling

Talked about: Probability

- $P(e)$ = *a priori probability*
  - the chance that *e* happens.

- $P(f|e)$ = conditional probability
  - the chance of *f* given *e*

- $P(f, e)$ = joint probability
  - the chance of both *e* and *f* happening
  - If *e* and *f* are independent, we can write
    - $P(e, f) = P(e) * P(f)$

Using the matrices which contain the frequency of the bigrams and unigrams for the words: *it, was, a cold, dark* and *night* from a corpus containing 50,000 tokens (N) and a vocabulary size (V) of 1,000.

$$P(b|a) = \frac{frequency\ (a\ b)}{frequency\ (a)}$$

w_i (b)

w_i-1 (a)

	it	was	a	cold	dark	night
it	0	5	2	0	1	0
was	0	0	2	3	5	0
a	0	0	0	5	2	1
cold	0	1	2	0	1	2
dark	1	2	2	1	0	5
night	0	0	0	1	0	5

it	was	a	cold	dark	night
100	10	20	15	25	20

What is the P(dark)?

Using the matrices which contain the frequency of the bigrams and unigrams for the words: *it, was, a cold, dark* and *night* from a corpus containing 50,000 tokens (N) and a vocabulary size (V) of 1,000.

$$P(b|a) = \frac{frequency\ (a\ b)}{frequency\ (a)}$$

w_i (b)

w_i-1 (a)		it	was	a	cold	dark	night
	it	0	5	2	0	1	0
	was	0	0	2	3	5	0
	a	0	0	0	5	2	1
	cold	0	1	2	0	1	2
	dark	1	2	2	1	0	5
	night	0	0	0	1	0	5

it	was	a	cold	dark	night
100	10	20	15	25	20

What is the P(cold)?

Using the matrices which contain the frequency of the bigrams and unigrams for the words: *it, was, a cold, dark* and *night* from a corpus containing 50,000 tokens (N) and a vocabulary size (V) of 1,000.

$$P(b|a) = \frac{frequency\ (a\ b)}{frequency\ (a)}$$

w_i (b)

w_i-1 (a)		it	was	a	cold	dark	night
	it	0	5	2	0	1	0
	was	0	0	2	3	5	0
	a	0	0	0	5	2	1
	cold	0	1	2	0	1	2
	dark	1	2	2	1	0	5
	night	0	0	0	1	0	5

it	was	a	cold	dark	night
100	10	20	15	25	20

What is the P(night|dark)?

Using the matrices which contain the frequency of the bigrams and unigrams for the words: *it, was, a cold, dark* and *night* from a corpus containing 50,000 tokens (N) and a vocabulary size (V) of 1,000.

$$P(b|a) = \frac{frequency\ (a\ b)}{frequency\ (a)}$$

w_i (b)

w_i-1 (a)		it	was	a	cold	dark	night
	it	0	5	2	0	1	0
	was	0	0	2	3	5	0
	a	0	0	0	5	2	1
	cold	0	1	2	0	1	2
	dark	1	2	2	1	0	5
	night	0	0	0	1	0	5

it	was	a	cold	dark	night
100	10	20	15	25	20

What is the P(night|cold)?

Talked about the: Chain Rule

Chain rule:

allows us to decompose the probability into a product of component conditional probabilities

P(the mythical unicorn) = P(the) * P(mythical|the) * P(unicorn|the mythical)

$$P(X_1 \ldots X_n) = P(X_1)\mathrm{P}(X_2|X_1)P(X_3|X_1^2) \ldots P(X_n|X_1^{n-1})$$

$$= \prod_{k=1}^{n} P(x_k|x_1^{k-1})$$

$$P(x_1^n) = \prod_{k=1}^{n} P\left(x_k \middle| x_1^{k-1}\right) = \prod_{k=1}^{n} P(x_k | x_{k-1})$$

Talked about : Markov Assummption:

Markov Assumption:

word is only dependent on its limit history

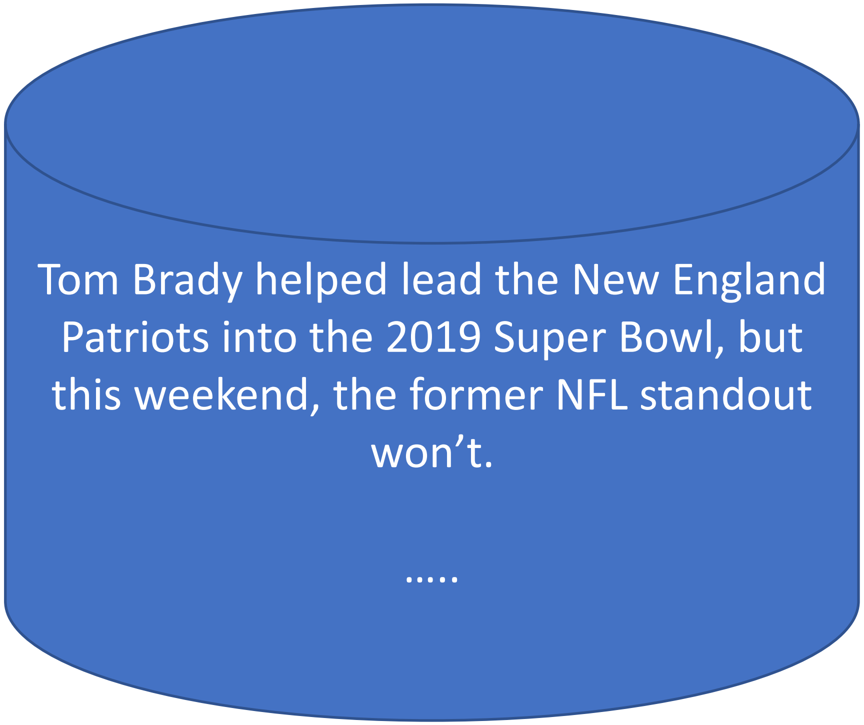
This allows us only go back *k*-1 words

Estimate:

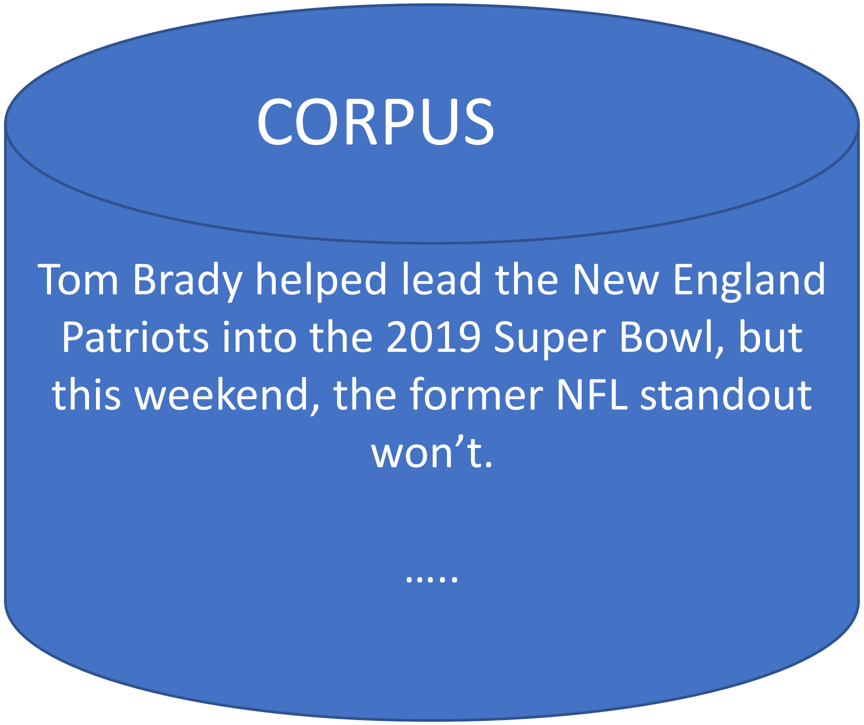
P(unicorn|the mythical)
using
(unicorn|mythical)

What you need to review and know

- Calculate the probability and conditional probability of words
- Understand the Chain Rule
- Understand the Markov assumption


Tom Brady helped lead the New England Patriots into the 2019 Super Bowl, but this weekend, the former NFL standout won’t.
.....

Unigram RAW FREQUENCIES



Unigram	Frequency
tom	1
brady	1
helped	1
lead	1
the	3
new	1
england	1
patriots	1
into	1
2019	1
super	1
bowl	1
but	1
this	1
weekend	1
Former	1
NFL	1
standout	1
wo	1
n‘t	1
.	1

	tom	brady	helped	lead	the	new	england	patriots	into	2019	super	bowl	but	this	weekend	former	nfl	standout	wo	n‘t	.
tom	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
brady	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
helped	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
lead	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
the	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0
new	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
england	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
patriots	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
into	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2019	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
super	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
bowl	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
but	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
this	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
weekend	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Former	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
NFL	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
standout	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
wo	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
n‘t	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bigram RAW FREQUENCIES

	tom	brady	helped	lead	the	new	england	patriots	into	2019	super	bowl	but	this	weekend	former	nfl	standout	won	‘t	.
tom	0	1	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0
brady	0	0	1	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0
helped	0	0	0	1	0	0	0				0	0	0	0	0	0	0	0	0	0	0
lead	0	0	0	0	1	0	0				0	0	0	0	0	0	0	0	0	0	0
the	0	0	0	0	0	1	0				0	0	0	0	0	1	0	0	0	0	0
new	0	0	0	0	0	0	1				0	0	0	0	0	0	0	0	0	0	0
england	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0
patriots	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0
into	0	0	0	0	1	0	0				0	0	0	0	0	0	0	0	0	0	0
2019	0	0	0	0	0	0	0				1	0	0	0	0	0	0	0	0	0	0
super	0	0	0	0	0	0	0				0	1	0	0	0	0	0	0	0	0	0
bowl	0	0	0	0	0	0	0				0	0	1	0	0	0	0	0	0	0	0
but	0	0	0	0	0	0	0				0	0	0	1	0	0	0	0	0	0	0
this	0	0	0	0	0	0	0				0	0	0	0	1	0	0	0	0	0	0
weekend	0	0	0	0	1	0	0				0	0	0	0	0	0	0	0	0	0	0
Former	0	0	0	0	0	0	0				0	0	0	0	0	0	1	0	0	0	0
NFL	0	0	0	0	0	0	0				0	0	0	0	0	0	0	1	0	0	0
standout	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	1	0	0
won	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	1	0
‘t	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	1
.	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0

Unigram	Frequency
tom	1
brady	1
helped	1
lead	1
the	3
new	1
england	1
patriots	1
into	1
2019	1
super	1
bowl	1
but	1
this	1
weekend	1
Former	1
NFL	1
standout	1
won	1
‘t	1
.	1

WE WANT THE RELATIVE FREQUENCIES

$$P(w2 \mid w1) = Freq(w1\ w2) / Freq\ (w1)$$

	tom	brady	helped	lead	the	new	england	patriots	into	2019	super	bowl	but	this	weekend	former	nfl	standout	wo	n‘t	.
tom	0	1	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0
brady	0	0	1	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0
helped											0	0	0	0	0	0	0	0	0	0	0
lead	0	0	0	0	1	0	0				0	0	0	0	0	0	0	0	0	0	0
the	0	0	0	0	0	1	0				0	0	0	0	0	1	0	0	0	0	0
new	0	0	0	0	0	0	1				0	0	0	0	0	0	0	0	0	0	0
england	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0
patriots	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0
into	0	0	0	0	1	0	0				0	0	0	0	0	0	0	0	0	0	0
2019	0	0	0	0	0	0	0				1	0	0	0	0	0	0	0	0	0	0
super	0	0	0	0	0	0	0				0	1	0	0	0	0	0	0	0	0	0
bowl	0	0	0	0	0	0	0				0	0	1	0	0	0	0	0	0	0	0
but	0	0	0	0	0	0	0				0	0	0	1	0	0	0	0	0	0	0
this	0	0	0	0	0	0	0				0	0	0	0	1	0	0	0	0	0	0
weekend	0	0	0	0	1	0	0				0	0	0	0	0	0	0	0	0	0	0
Former	0	0	0	0	0	0	0				0	0	0	0	0	0	1	0	0	0	0
NFL	0	0	0	0	0	0	0				0	0	0	0	0	0	0	1	0	0	0
standout	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	1	0	0
wo	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	1	0
n‘t	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	1
.	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0

Unigram	Frequency
tom	1
brady	1
helped	1
lead	1
the	3
new	1
england	1
patriots	1
into	1
2019	1
super	1
bowl	1
but	1
this	1
weekend	1
Former	1
NFL	1
standout	1
won	1
‘t	1
.	1

P(new | the) = Freq(the new)/Freq(the)

1/3

WE WANT THE RELATIVE FREQUENCIES

P(w2 | w1) = Freq(w1 w2) / Freq (w1)

	tom	brady	helped	lead	the	new	england	patriots	into	2019	super	bowl	but	this	weekend	former	nfl	standout	wo	n‘t	.
tom	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
brady	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
helped	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
lead	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
the	0	0	0	0	0	1/3	0	0	0	1/3	0	0	0	0	0	1	0	0	0	0	0
new	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
england	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
patriots	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
into	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2019	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
super	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
bowl	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
but	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
this	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
weekend	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Former	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
NFL	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
standout	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
wo	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
n‘t	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RELATIVE FREQUENCIES

P(Tom Brady helped lead the New England Patriots into the 2019 Super Bowl but this weekend the former NFL standout won't) =

P(tom) *
P(brady|tom) *
P(helped | brady) *
P(lead | helped) *
P(the | lead) *
P(new | the) *
P(england | new) *
….
P(n't | wo)

P(Tom Brady helped lead the New England Patriots into the 2019 Super Bowl but this weekend the former NFL standout won’t) =

P(tom) *
P(brady|tom) *
P(helped | brady) *
P(lead | helped) *
P(the | lead) *
P(new | the) *
P(england | new) *
….
P(n’t | wo)

chain rule of probability
&
the markov assumption

Chain rule of probability

$$P(X_1 \ldots X_n) = P(X_1)\mathrm{P}(X_2|X_1)P\left(X_3\middle|X_1^2\right) \ldots P(X_n|X_1^{n-1})$$

P(Tom Brady helped lead the New England Patriots into the 2019 Super Bowl but this weekend the former NFL standout won’t) =

P(tom) *
P(brady|tom) *
P(helped | tom brady) *
P(lead | tom brady helped) *
P(the | tom brady helped lead) *
….
P(n’t| tom brady … standout wo)

Chain rule of probability

$$P(X_1 \ldots X_n) = P(X_1)\mathrm{P}(X_2|X_1)P\left(X_3\middle|X_1^2\right) \ldots P(X_n|X_1^{n-1})$$

P(Tom Brady helped lead the New England Patriots into the 2019 Super Bowl but this weekend the former NFL standout won’t) =

P(tom) *
P(brady|tom) *
P(helped | tom brady) *
P(lead | tom brady helped) *
P(the | tom brady helped lead) *
….
P(n’t| tom brady … standout wo)

Why is it difficult to calculate this?

Chain rule of probability

$$P(X_1 \ldots X_n) = P(X_1)\mathrm{P}(X_2|X_1)P\left(X_3\middle|X_1^2\right) \ldots P(X_n|X_1^{n-1})$$

P(tom brady helped lead the New England Patriots into the 2019 Super Bowl but this weekend the former NFL standout won’t) =

P(tom) *
P(brady|tom) *
P(helped | tom brady) *
P(lead | tom brady helped) *
P(the | tom brady helped lead) *
….
P(n’t| tom brady … wo)

calculating this is
tough due to sparseness:

The chances of seeing
“Tom Brady helped lead the New England Patriots into the 2019 Super Bowl but this weekend the former NFL standout won’t” is slim

Markov assumption

Estimate the conditional probability of the next word without looking too far in the past

$$P\left(w_n \middle| w_1^{n-1}\right) \approx P\left(w_n \middle| w_{n-N+1}^{n-1}\right)$$

Markov assumption

Estimate the conditional probability of the next word without looking too far in the past

$$P\left(w_n \middle| w_1^{n-1}\right) \approx P\left(w_n \middle| w_{n-N+1}^{n-1}\right)$$

P(n’t| tom brady …standout wo) = P(n’t | wo) **Using a bigram model**

P(n’t| tom brady …standout wo) = P(n’t | standout wo) **Using a trigram model**

P(n’t| tom brady …standout wo) = P(n’t | nfl standout wo) **Using a 4-gram model**

etc …

P(Tom brady helped lead the New England Patriots into the 2019 Super Bowl, but this weekend, the former NFL standout) =

P(tom) *
P(brady|tom) *
P(helped | brady) *
P(lead | helped) *
P(the | lead) *
P(new | the) *
P(england | new) *
....
P(n't| wo)

Bigram Relative Frequency Table

	aaron	hernandez	helped	lead	the	new	england	patriots	into	2011	super	bowl	but	this	weekend	former	nfl	standout	wo	n't	.
aaron	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
hernandez	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
helped	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
lead	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
the	0	0	0	0	0	1/3	0	0	0	1/3	0	0	0	0	0	1	0	0	0	0	0
new	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
england	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
patriots	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
into	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2011	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
super	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
bowl	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
but	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
this	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
weekend	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Former	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
NFL	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
standout	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
wo	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
n't	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

P(Tom brady helped lead the New England Patriots into the 2019 Super Bowl, but this weekend, the former NFL standout) =

P(tom) *
P(brady|tom) *
P(helped | brady) *
P(lead | helped) *
P(the | lead) *
P(new | the) *
P(england | new) *
….
P(n’t| wo)

Bigram Relative Frequency Table

	aaron	hernandez	helped	lead	the	new	england	patriots	into	2011	super	bowl	but	this	weekend	former	nfl	standout	wo	n't	.
aaron	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
hernandez	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
helped	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
lead	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
the	0	0	0	0	0	1/3	0	0	0	1/3	0	0	0	0	0	1	0	0	0	0	0
new	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
england	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
patriots	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
into	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2011	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
super	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
bowl	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
but	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
this	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
weekend	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Former	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
NFL	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
standout	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
wo	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
n't	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

How do we begin and end our sentences?

- Now let’s look at a different example and introduce
  - <start>
  - <end>

	i	want	to	eat	chinese	food	lunch	spend	<start>	<end>
i	5	827	0	9	0	0	0	2	0	0
want	2	0	608	1	6	6	5	1	0	0
to	2	0	4	686	2	0	6	211	0	0
eat	0	0	2	0	16	2	42	0	0	34
chinese	1	0	0	0	0	82	1	0	0	23
food	15	0	15	0	1	1	0	0	0	12
lunch	2	0	0	0	0	0	0	0	0	9
spend	1	0	1	0	0	0	0	0	1	17
<start>	45	0	30	0	15	10	3	0	0	0
<end>	0	0	0	0	3	23	6	34	0	0

Bigram table of raw frequency's

i	want	to	eat	chinese	food	lunch	spend	<start>	<end>
2533	927	2417	746	158	1093	341	278	3000	3000

Unigram table of raw frequency's

$$P(w2|w1) = \frac{frequency(w1\ w2)}{frequency(w1)}$$

$$P(i \mid < start >) = ?$$

	i	want	to	eat	chinese	food	lunch	spend	<start>	<end>
i	5	827	0	9	0	0	0	2	0	0
want	2	0	608	1	6	6	5	1	0	0
to	2	0	4	686	2	0	6	211	0	0
eat	0	0	2	0	16	2	42	0	0	34
chinese	1	0	0	0	0	82	1	0	0	23
food	15	0	15	0	1	1	0	0	0	12
lunch	2	0	0	0	0	0	0	0	0	9
spend	1	0	1	0	0	0	0	0	1	17
<start>	45	0	30	0	15	10	3	0	0	0
<end>	0	0	0	0	3	23	6	34	0	0

i	want	to	eat	chinese	food	lunch	spend	<start>	<end>
2533	927	2417	746	158	1093	341	278	3000	3000

$$P(w2|w1) = \frac{frequency(w1\ w2)}{frequency(w1)}$$

	i	want	to	eat	chinese	food	lunch	spend	<start>	<end>
i	5	827	0	9	0	0	0	2	0	0
want	2	0	608	1	6	6	5	1	0	0
to	2	0	4	686	2	0	6	211	0	0
eat	0	0	2	0	16	2	42	0	0	34
chinese	1	0	0	0	0	82	1	0	0	23
food	15	0	15	0	1	1	0	0	0	12
lunch	2	0	0	0	0	0	0	0	0	9
spend	1	0	1	0	0	0	0	0	1	17
<start>	45	0	30	0	15	10	3	0	0	0
<end>	0	0	0	0	3	23	6	34	0	0

$$P(i \mid < start >) = \frac{45}{3000}$$

$$P(< end > \mid spend) = ?$$

i	want	to	eat	chinese	food	lunch	spend	<start>	<end>
2533	927	2417	746	158	1093	341	278	3000	3000

$$P(w2|w1) = \frac{frequency(w1\, w2)}{frequency(w1)}$$

	i	want	to	eat	chinese	food	lunch	spend	<start>	<end>
i	5	827	0	9	0	0	0	2	0	0
want	2	0	608	1	6	6	5	1	0	0
to	2	0	4	686	2	0	6	211	0	0
eat	0	0	2	0	16	2	42	0	0	34
chinese	1	0	0	0	0	82	1	0	0	23
food	15	0	15	0	1	1	0	0	0	12
lunch	2	0	0	0	0	0	0	0	0	9
spend	1	0	1	0	0	0	0	0	1	17
<start>	45	0	30	0	15	10	3	0	0	0
<end>	0	0	0	0	3	23	6	34	0	0

$$P(i \mid < start >) = \frac{45}{3000}$$

$$P(< end > \mid spend) = \frac{17}{278}$$

i	want	to	eat	chinese	food	lunch	spend	<start>	<end>
2533	927	2417	746	158	1093	341	278	3000	3000

Any questions?

What does this allow you to do

Calculate the probability of arbitrarily long sentences

Generate arbitrarily long sentences

Let's look closer on how you can do this.

Programming assignment 2

How to Generate Sentences

We know to begin the sentences we want to use the <start> tag

	i	want	to	eat	chinese	food	lunch	spend	<start>	<end>
i	0.002	0.33	0	0.0036	0	0	0	0.00079	0	0
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0064	0.0011	0	0
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087	0	0
eat	0	0	0.0027	0	0.021	0.0027	0.056	0	0	0.011
chinese	0.0063	0	0	0	0	0.52	0.0063	0	0	0.008
food	0.014	0	0.014	0	0.00092	0.0037	0	0	0	0.004
lunch	0.0059	0	0	0	0	0.0029	0	0	0	0.003
spend	0.0036	0	0.0036	0	0	0	0	0	1	0.006
<start>	0.015	0	0.01	0	0.005	0.003	0.001	0	0	0
<end>	0	0	0	0	0.001	0.007	0.002	0.011	0	0

How to Generate Sentences

We know to begin the sentences we want to use the <start> tag

	i	want	to	eat	chinese	food	lunch	spend	<start>	<end>
<start>	0.15	0	0.1	0	0.4	0.3	0.05	0	0	0

How to Generate Sentences

	i	want	to	eat	chinese	food	lunch	spend	<start>	<end>
<start>	0.15	0	0.1	0	0.4	0.3	0.05	0	0	0

We know to begin the sentences
we want to use the <start> tag

Now we are only interested in those
Words that follow <start>
(the non zero elements)

Why?

Because we are using
our language model
(the relative frequency table)
To generate the words in our sentence

Which of the five choices do we choose

	i	to	chinese	food	lunch
<start>	0.15	0.1	0.4	0.3	0.05

If we pick the one with the highest probability our sentences are not going to change very much

So

randomly pick one based on its distribution

Which of the five choices do we choose

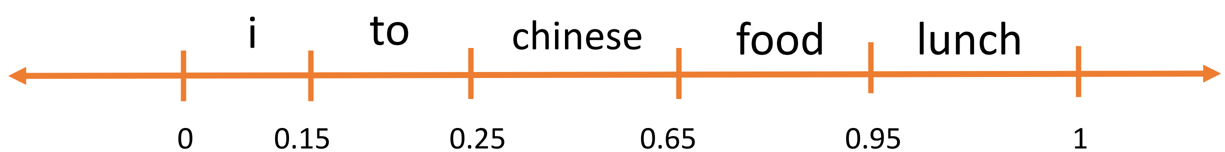
	i	to	chinese	food	lunch
<start>	0.15	0.1	0.4	0.3	0.05

Randomly pick one based on its distribution

Which of the five choices do we choose

	i	to	chinese	food	lunch
<start>	0.15	0.1	0.4	0.3	0.05

$$0.15 + 0.10 + 0.40 + 0.30 + 0.05 = 1$$

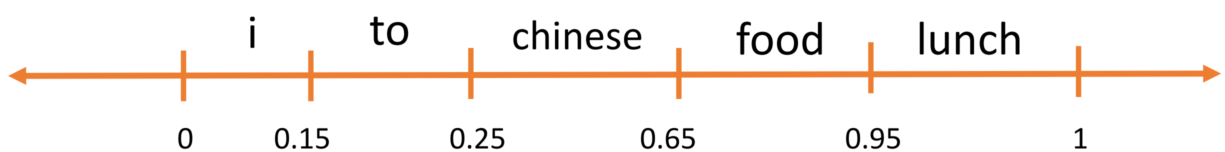


We can plot these probabilities
a line from 0 to 1

Which of the five choices do we choose

	i	to	chinese	food	lunch
<start>	0.15	0.1	0.4	0.3	0.05

$$0.15 + 0.10 + 0.40 + 0.30 + 0.05 = 1$$



We can plot these probabilities
a line from 0 to 1

Now we can randomly pick what word
Follows <start> given the distribution
of them occurring within the text

Which of the five choices do we choose

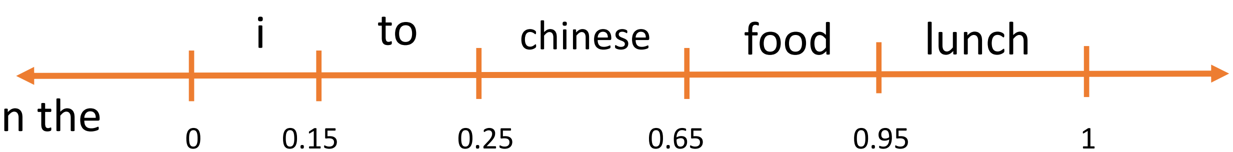
	i	to	chinese	food	lunch
<start>	0.15	0.1	0.4	0.3	0.05

Pick a random number between zero and one

$$0.15 + 0.10 + 0.40 + 0.30 + 0.05 = 1$$

```
my $r = rand();
```

And then see where it falls on the distribution:




```
if($r <= 0.44)     { $next_word = “i”; }
elsif($r <= 0.73) { $next_word = “to”; }
elsif($r <= 0.88){ $next_word = “chinese”; }
elsif($r <= 0.97) { $next_word = “food”; }
else  { $next_word = “lunch”; }
```

We can plot these probabilities
a line from 0 to 1

Now we can randomly pick what word
Follows <start> given the distribution
of them occurring within the text

So say our rand returned the value 0.2 what is our next word?

Which of the five choices do we choose

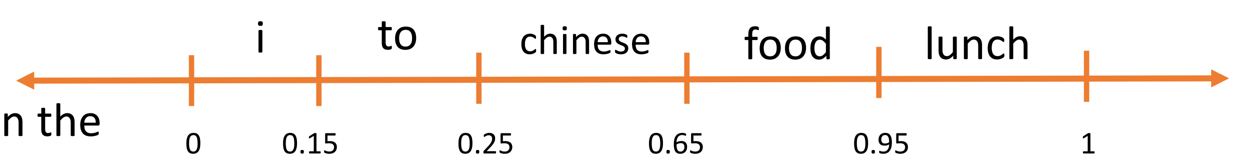
	i	to	chinese	food	lunch
<start>	0.15	0.1	0.4	0.3	0.05

Pick a random number between zero and one

$$0.15 + 0.10 + 0.40 + 0.30 + 0.05 = 1$$

```
my $r = rand();
```

And then see where it falls on the distribution:




```
if($r <= 0.15)    { $next_word = “i”; }
elsif($r <= 0.25) { $next_word = “to”; }
elsif($r <= 0.65){ $next_word = “chinese”; }
elsif($r <= 0.95) { $next_word = “food”; }
else  { $next_word = “lunch”; }
```

We can plot these probabilities a line from 0 to 1

Now we can randomly pick what word Follows <start> given the distribution of them occurring within the text

So then we start the process again with ‘to’

	i	want	to	eat	chinese	food	lunch	spend	<start>	<end>
i	0.002	0.33	0	0.0036	0	0	0	0.00079	0	0
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0064	0.0011	0	0
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087	0	0
eat	0	0	0.0027	0	0.021	0.0027	0.056	0	0	0.011
chinese	0.0063	0	0	0	0	0.52	0.0063	0	0	0.008
food	0.014	0	0.014	0	0.00092	0.0037	0	0	0	0.004
lunch	0.0059	0	0	0	0	0.0029	0	0	0	0.003
spend	0.0036	0	0.0036	0	0	0	0	0	1	0.006
<start>	0.015	0	0.01	0	0.005	0.003	0.001	0	0	0
<end>	0	0	0	0	0.001	0.007	0.002	0.011	0	0

Do you see how this extends any n-gram?

	am	eat	dish	…	box
<start> i	0.44	0.29	0.15	…	0.03

Questions?

Generation versus Accepting

- We have been talking about using Language Models to Generate sentences

- But what about calculating the probability of a sentence occurring?

Scarcity

As N increases the accuracy of our model increase

But

As N increases the sparsely of our model increases

	i	want	to	eat	chinese	food	lunch	spend	<start>	<end>
i	0.002	0.33	0	0.0036	0	0	0	0.00079	0	0
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0064	0.0011	0	0
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087	0	0
eat	0	0	0.0027	0	0.021	0.0027	0.056	0	0	0.011
chinese	0.0063	0	0	0	0	0.52	0.0063	0	0	0.008
food	0.014	0	0.014	0	0.00092	0.0037	0	0	0	0.004
lunch	0.0059	0	0	0	0	0.0029	0	0	0	0.003
spend	0.0036	0	0.0036	0	0	0	0	0	1	0.006
<start>	0.015	0	0.01	0	0.005	0.003	0.001	0	0	0
<end>	0	0	0	0	0.001	0.007	0.002	0.011	0	0

LOOK AT ALL THE ZERO’S

	i	want	to	eat	chinese	food	lunch	spend	<start>	<end>
i	0.002	0.33	0	0.0036	0	0	0	0.00079	0	0
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0064	0.0011	0	0
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087	0	0
eat	0	0	0.0027	0	0.021	0.0027	0.056	0	0	0.011
chinese	0.0063	0	0	0	0	0.52	0.0063	0	0	0.008
food	0.014	0	0.014	0	0.00092	0.0037	0	0	0	0.004
lunch	0.0059	0	0	0	0	0.0029	0	0	0	[illegible]
spend	0.0036	0	0.0036	0	0	0	0	0	1	0.006
<start>	0.015	0	0.01	0	0.005	0.003	0.001	0	0	0
<end>	0	0	0	0	0.001	0.007	0.002	0.011	0	0

Does this mean that P(want|spend) = 0?

With the model, yes but in real life?

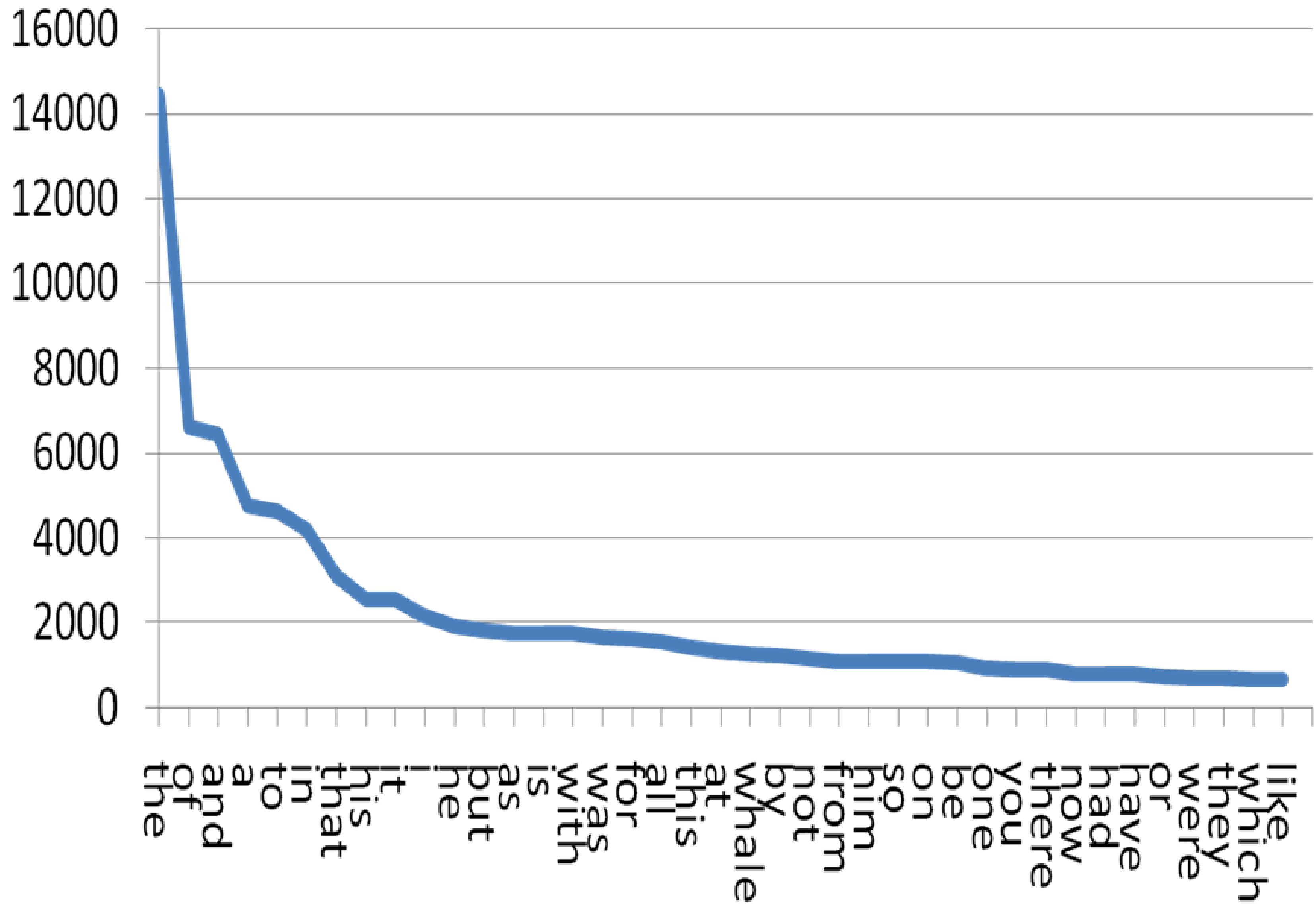
Smoothing

- Exploit the Zipfian distribution of words
- Two smoothing methods:
  - Laplace Smoothing
  - Good Turning
- The basic idea is that we take a little from everything we see and give it to what we don't see
- Robin Hood: stealing from the rich and giving to the poor

Zipfian Distribution

- Words follow a Zipfian Distribution
  - Small number of words occur very frequently
  - A large number of words are only seen once

  - Zipf’s Law: A word’s frequency is approximately inversely proportional to its rank in the word distribution list



16000
14000
12000
10000
8000
6000
4000
2000
0
the
of
and
a
to
in
that
his
it
i
he
but
as
is
with
was
for
all
this
at
whale
by
not
from
him
so
on
be
one
you
there
now
had
have
or
were
they
which
like

Great Video on Zipf

https://www.youtube.com/watch?v=fCn8zs912OE

Laplace Smoothing

- Simple metric : adds one to each count

- $P(w_i) = \frac{frequency(w_i)}{N}$

- $P_{Laplace}(w_i) = \frac{frequency(w_i)+1}{N+V}$

- N = the number of tokens in our corpus
- V = the number of types in our corpus

Laplace Smoothing

- Simple metric : adds one to each count
- $P(w_i) = \frac{frequency(w_i)}{N}$
- $P_{Laplace}(w_i) = \frac{frequency(w_i)+1}{N+V}$
- N = the number of tokens in our corpus
- V = the number of types in our corpus

Adding V because you've added one to each *w* seen in your corpus

Sometimes referred to as "adjusted count"

$$frequency^*(w_i) = (frequency(w_i) + 1)\frac{N}{N + V}$$

$$P(w_i) = \frac{frequency^*(w_i)}{N}$$

Adjusted count

$$frequency^*(w_i) = (frequency(w_i) + 1)\,\frac{N}{N+V}$$

$$P(w_i) = \frac{frequency^*(w_i)}{N}$$

1: $$P(w_i) = \frac{(frequency(w_i) + 1)\,\frac{N}{N+V}}{N}$$

3: $$P(w_i) = \frac{N(frequency(w_i)+1)}{N+V} * \frac{1}{N}$$

2: $$P(w_i) = \frac{\frac{N(frequency(w_i) + 1)}{N+V}}{N}$$

4: $$P(w_i) = \frac{(frequency(w_i) + 1)}{N+V}$$

Adjusted count

$$frequency^*(w_i) = (frequency(w_i) + 1)\frac{N}{N+V}$$

$$P(w_i) = \frac{frequency^*(w_i)}{N}$$

$$1\colon\ P(w_i) = \frac{(frequency(w_i) + 1)\frac{N}{N+V}}{N}$$

$$2\colon\ P(w_i) = \frac{\frac{N(frequency(w_i) + 1)}{N+V}}{N}$$

$$3\colon\ P(w_i) = \frac{N(frequency(w_i)+1)}{N+V} * \frac{1}{N}$$

$$4\colon\ P(w_i) = \frac{(frequency(w_i) + 1)}{N+V}$$

$$P_{Laplace}(w_i) = \frac{frequency(w_i) + 1}{N+V}$$

Laplace Smoothing on Conditional Probabilities

$$P(w_i) = \frac{frequency(w_i)}{N} \Rightarrow P_{Laplace}(w_i) = \frac{frequency(w_i) + 1}{N + V}$$

$$P(w1|w2) = \frac{frequency(w1,w2)}{frequency(w2)} \Rightarrow P_{Laplace}(w1|w2) = ?$$

V = the number of types in our corpus

Laplace Smoothing on Conditional Probabilities

$$P(w_i) = \frac{frequency(w_i)}{N} \Rightarrow P_{Laplace}(w_i) = \frac{frequency(w_i) + 1}{N + V}$$

$$P(w1|w2) = \frac{frequency(w1,w2)}{frequency(w1)} \Rightarrow P_{Laplace}(w1|w2) = ?$$

$$P_{Laplace}(w_n|w_{n-1}) = \frac{frequency(w_{n-1}w_n) + 1}{frequency(w_{n-1}) + V}$$

V = the number of types in our corpus

	i	want	to	eat	chinese	food	lunch	spend	<start>	<end>
i	5	827	0	9	0	0	0	2	0	0
want	2	0	608	1	6	6	5	1	0	0
to	2	0	4	686	2	0	6	211	0	0
eat	0	0	2	0	16	2	42	0	0	34
chinese	1	0	0	0	0	82	1	0	0	23
food	15	0	15	0	1	1	0	0	0	12
lunch	2	0	0	0	0	0	0	0	0	9
spend	1	0	1	0	0	0	0	0	1	17
<start>	45	0	30	0	15	10	3	0	0	0
<end>	0	0	0	0	3	23	6	34	0	0

i	want	to	eat	chinese	food	lunch	spend	<start>	<end>
2533	927	2417	746	158	1093	341	278	3000	3000

$$P_{Laplace}(w_n|w_{n-1}) = \frac{frequency(w_{n-1}w_n) + 1}{frequency(w_{n-1}) + V}$$

$$P_{Laplace}(want|i) = \frac{frequency(i\ want) + 1}{frequency(i) + V}$$

$\boxed{V = 1446}$

	i	want	to	eat	chinese	food	lunch	spend	<start>	<end>
i	5	827	0	9	0	0	0	2	0	0
want	2	0	608	1	6	6	5	1	0	0
to	2	0	4	686	2	0	6	211	0	0
eat	0	0	2	0	16	2	42	0	0	34
chinese	1	0	0	0	0	82	1	0	0	23
food	15	0	15	0	1	1	0	0	0	12
lunch	2	0	0	0	0	0	0	0	0	9
spend	1	0	1	0	0	0	0	0	1	17
<start>	45	0	30	0	15	10	3	0	0	0
<end>	0	0	0	0	3	23	6	34	0	0

i	want	to	eat	chinese	food	lunch	spend	<start>	<end>
2533	927	2417	746	158	1093	341	278	3000	3000

$$P_{Laplace}(w_n|w_{n-1}) = \frac{frequency(w_{n-1}w_n) + 1}{frequency(w_{n-1}) + V}$$

$$P_{Laplace}(want|i) = \frac{frequency(i\ want) + 1}{frequency(i) + V}$$

$$\frac{827+1}{2533+1446} = 0.21$$

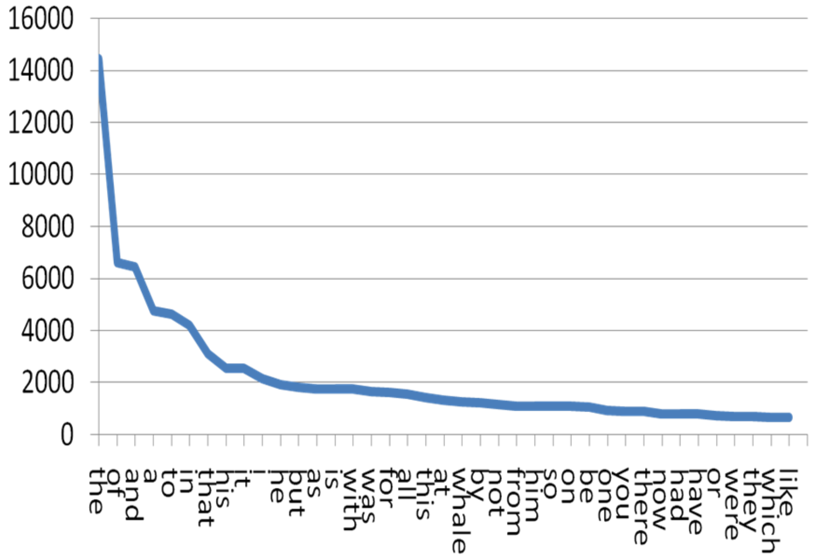
$\boxed{V = 1446}$

Good Turing

- Add-1 smoothing (Laplace smoothing) is a bit brute force
  - Few more elegant ways to smooth
    - **Good Turning**
    - Witten-Bell
    - Kneser-Ney

Good Turing

- Intuition
  - Use the count of things you have seen once to help estimate the count of things you've never seen



Good Turing

Based on computing N_c which is the number of N-grams that occur c times

frequency of frequency

$$N_o = \#\ of\ bigrams\ with\ count\ 0$$
$$N_1 = \#\ of\ bigrams\ with\ count\ 1$$
$$\ldots$$
$$N_c = \#\ of\ bigrams\ with\ count\ c$$

Redefine frequency

$$\left.frequency^*(w_i) = (frequency(w_i) + 1)\frac{N}{N+V}\right\} \text{Laplace Smoothing}$$

Redefine frequency

$$frequency^*(w_i) = (frequency(w_i) + 1)\frac{N}{N + V}$$

Laplace Smoothing

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

Good Turing Smoothing

Redefine frequency

$$frequency^*(w_i) = (frequency(w_i) + 1)\frac{N}{N+V}$$

Laplace Smoothing

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

Good Turing Smoothing

$$P_{smoothing}(w_n|w_{n-1}) = \frac{frequency^*(w_{n-1}w_n)}{frequency^*(w_{n-1})}$$

But what about unseen bigrams

$$P_{gt}(unseen) = \frac{N_1}{N_o}$$

How do we know what N_o is given
we don't know the number unseen events?

N_1 = number of bigrams seen 1 time
N_o = total number of bigrams in the corpus

But what about unseen bigrams

$$P_{gt}(unseen) = \frac{N_1}{N_o}$$

How do we know what N is given
we don't know the number unseen events?

N_1 = number of bigrams seen 1 time
N_o = total number of bigrams in the corpus

Guesstimate

We know V (the vocabulary size), therefore
The total number of bigrams = V^2
So

$$N_0 = V^2 - \# \, seen \, bigrams$$

Frequency	Frequency(Frequency)
0	2081496
1	5315
2	1419
3	642
4	381
5	311
6	196
…	…
M	…

N = 2,081,496

	i	want	to	eat	chinese	food	lunch	spend	<start>	<end>
i	5	827	0	9	0	0	0	2	0	0

i spend occurs twice in our corpus

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

$$frequency^*(i\ spend) = (frequency(i\ spend) + 1)\frac{N_3}{N_2}$$

$$frequency^*(i\ spend) = (2 + \ 1)\frac{642}{1419} = 1.36$$

$$\boxed{P_{gt}(spend \mid i) = \frac{frequency^*(i\,spend)}{frequency^*(i)} = \frac{1.36}{2534} = 0.00054\ (versus\ 0.00039)}$$

Frequency	Frequency(Frequency)
0	2081496
1	5315
2	1419
3	642
4	381
5	311
6	196
…	…
M	…

N = 2,081,496

	i	want	to	eat	chinese	food	lunch	spend	<start>	<end>
i	5	827	0	9	0	0	0	2	0	0

i spend occurs twice in our corpus

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

$$frequency^*(i\ spend) = (frequency(i\ spend) + 1)\frac{N_3}{N_2}$$

$$frequency^*(i\ spend) = (2 + \ 1)\frac{642}{1419} = 1.36$$

$$\boxed{P_{gt}(spend \mid i) = \frac{frequency^*(i\, spend)}{frequency^*(i)} = \frac{1.36}{2534} = 0.00054\ (versus\ 0.00039)}$$

How do we know this?

Frequency	Frequency(Frequency)
0	2081496
1	5315
2	1419
3	642
4	381
5	311
6	196
…	…
M	…

N = 2,081,496

	i	want	to	eat	chinese	food	lunch	spend	<start>	<end>
i	5	827	0	9	0	0	0	2	0	0

i spend occurs twice in our corpus

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

$$frequency^*(i\ spend) = (frequency(i\ spend) + 1)\frac{N_3}{N_2}$$

$$frequency^*(i\ spend) = (2 + \ 1)\frac{642}{1419} = 1.36$$

$$\boxed{P_{gt}(spend \mid i) = \frac{frequency^*(i\ spend)}{frequency^*(i)} = \frac{1.36}{2534} = 0.00054\ (versus\ 0.00039)}$$

Frequency	Frequency(Frequency)
…	…
2533	2
2534	2
…	…

How do we know this?

i	want	to	eat	chinese	food	lunch	spend	<start>	<end>
2533	927	2417	746	158	1093	341	278	3000	3000

Frequency	Frequency(Frequency)
0	2081496
1	5315
2	1419
3	642
4	381
5	311
6	196
…	…
M	…

N = 2,081,496

	i	want	to	eat	chinese	food	lunch	spend	<start>	<end>
i	5	827	0	9	0	0	0	2	0	0

i spend occurs twice in our corpus

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

$$frequency^*(i\ spend) = (frequency(i\ spend) + 1)\frac{N_3}{N_2}$$

$$frequency^*(i\ spend) = (2 + \ 1)\frac{642}{1419} = 1.36$$

$$\boxed{P_{gt}(spend \mid i) = \frac{frequency^*(i\ spend)}{frequency^*(i)} = \frac{1.36}{2534} = 0.00054\ (versus\ 0.00039)}$$

Frequency	Frequency(Frequency)
…	…
2533	2
2534	2
…	…

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

$$frequency^*(i) = (2533 + \ 1)\frac{2}{2} = 2534$$

i	want	to	eat	chinese	food	lunch	spend	<start>	<end>
2533	927	2417	746	158	1093	341	278	3000	3000

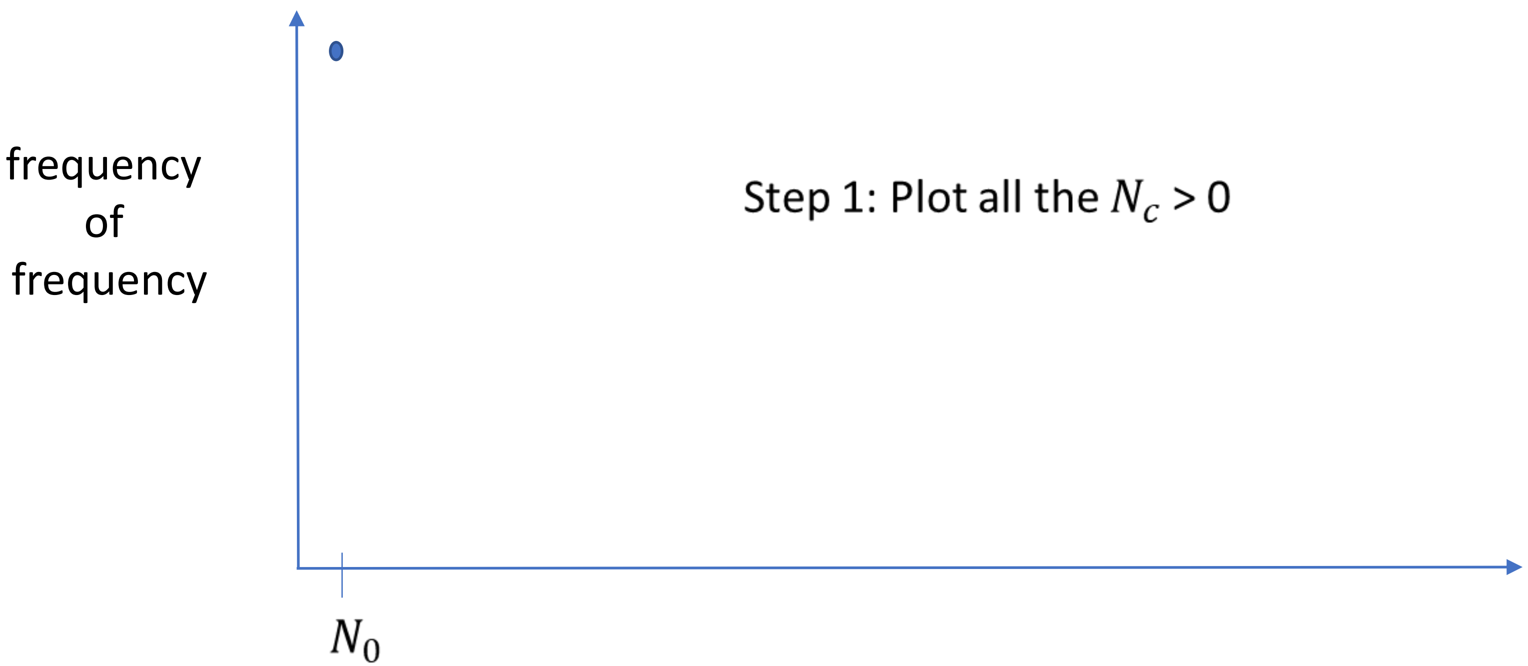
What happens when $N_{c+1} = 0$

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

Simplest thing is to perform linear regressions and replace the value of N_{c+1} with regression value whenever N_{c+1} = 0

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

Frequency of frequency: is the number of ngrams that occurred N_{c+1} times

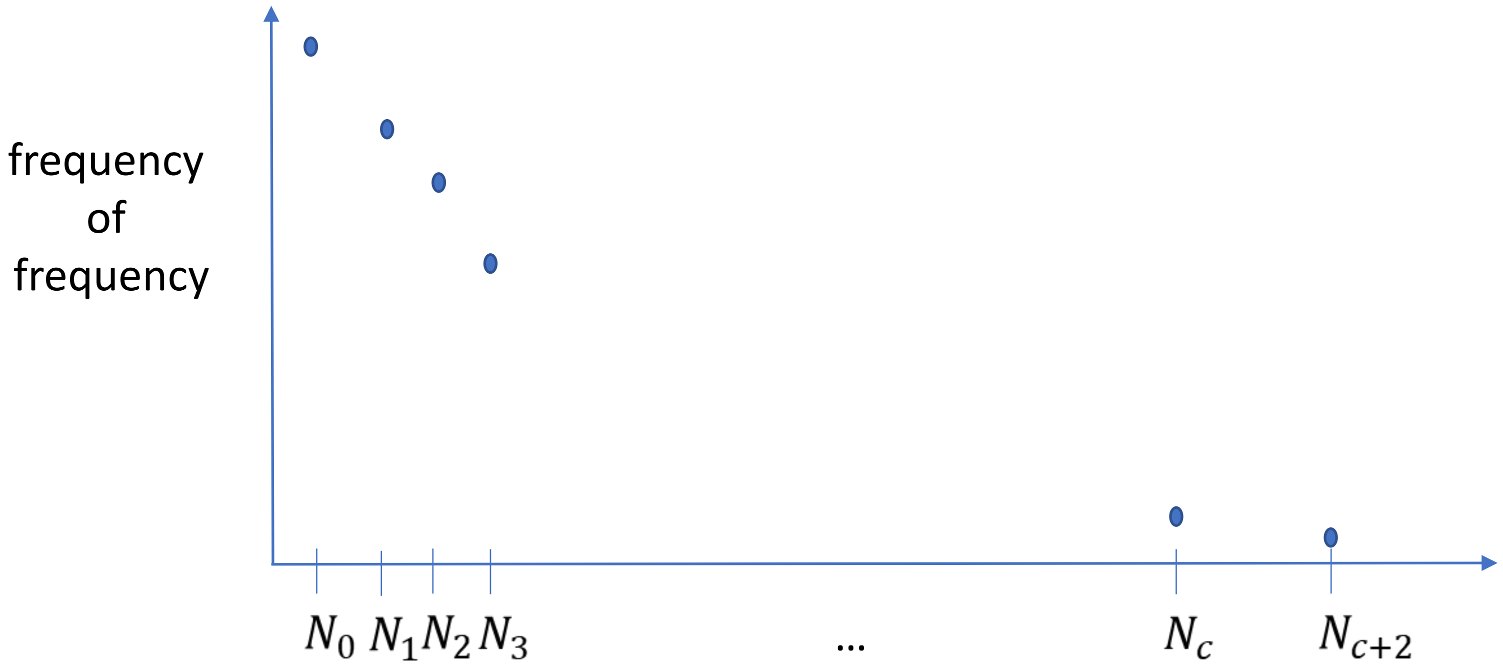


Estimating when $N_{c+1} = 0$

Frequency of frequency: is the number of ngrams that occurred N_{c+1} times

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

Frequency of frequency: is the number of ngrams that occurred N_{c+1} times

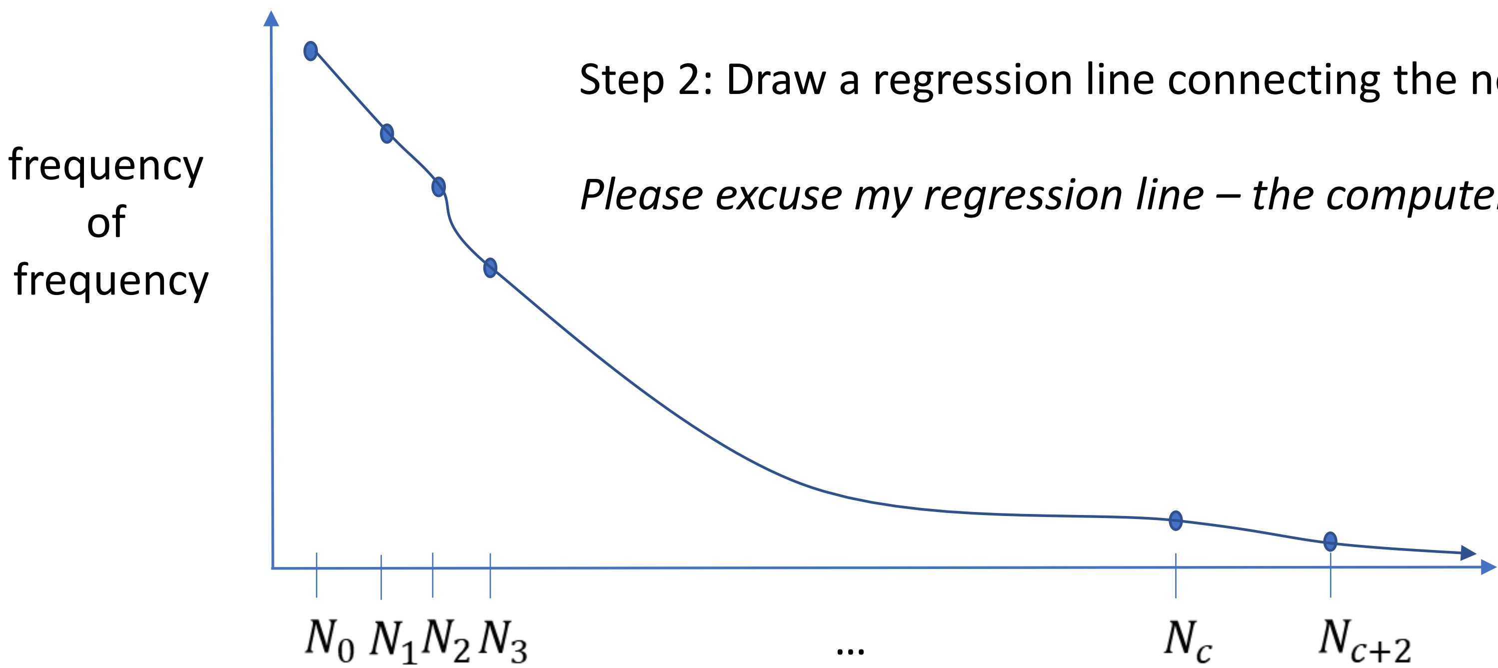


Estimating when $N_{c+1} = 0$

Frequency of frequency: is the number of ngrams that occurred N_{c+1} times

Step 2: Draw a regression line connecting the non-zero points

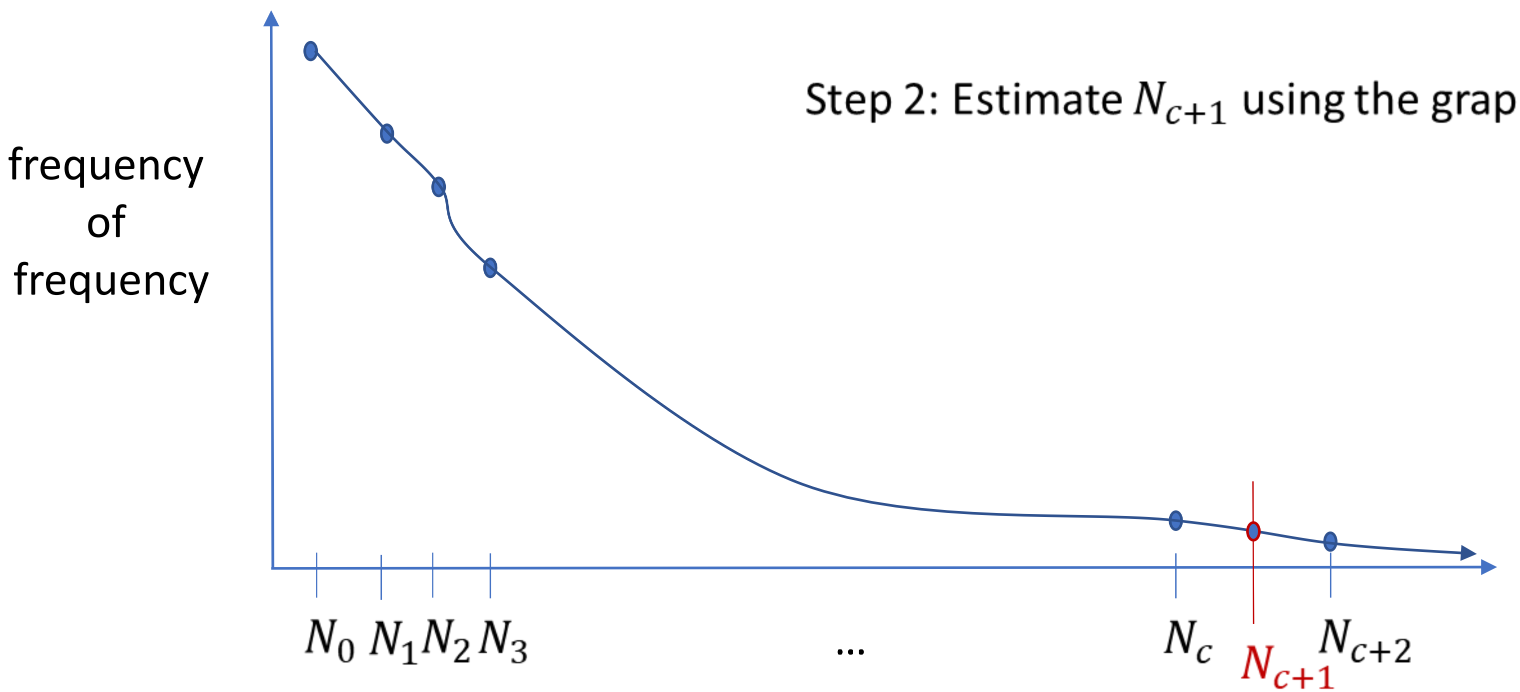
Please excuse my regression line – the computer does this better



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

Frequency of frequency: is the number of ngrams that occurred N_{c+1} times

Step 2: Estimate N_{c+1} using the graph



Do you need smoothing, for sentence generation?

Questions?

Evaluating Ngrams

- Perplexity
  - intrinsic evaluation metric for N-gram language models
  - the idea: given two probabilistic models, the better model:
    - is the one that has a tighter fit to the test data
    - that better predicts the details of the test data
  - measure this: looking at the probability the model assigns to the test data
    - the better model will assign a high probability to the test data.

Perplexity (PP)

$$W = w_1 \, w_2 \, w_3 \; \ldots \; w_n$$

Perplexity = probability of the test set normalized by the number of words

$$PP(W) = P(w_1 w_2 \; \ldots \; w_n)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1 \, w_2 \; \ldots \; w_n)}}$$

Perplexity (PP)

$$W = w_1 \, w_2 \, w_3 \; \dots \; w_n$$

Perplexity = probability of the test set normalized by the number of words

$$PP(W) = P(w_1 w_2 \; \dots \; w_n)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1 \, w_2 \; \dots \; w_n)}}$$

What is the difficulty here?

Perplexity (PP)

$$W = w_1\, w_2\, w_3\ \dots\ w_n$$

Perplexity = probability of the test set normalized by the number of words

$$PP(W) = P(w_1 w_2\ \dots\ w_n)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1\, w_2\ \dots\ w_n)}}$$

And what Rule did we learn with Language Modeling that we can apply here?

Perplexity: Chain Rule

$$PP(W) = P(w_1 w_2 \ \dots \ w_n)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1 \ w_2 \ \dots \ w_n)}}$$

$$= \sqrt[N]{\prod_{1}^{N} \frac{1}{P(w_i | w_1 w_2 \ \dots w_{i-1})}}$$

Chain rule: allows us to decompose the probability into a product of component conditional probabilities

Perplexity: Chain Rule

$$PP(W) = P(w_1 w_2 \ \dots \ w_n)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1 \, w_2 \ \dots \ w_n)}}$$

$$= \sqrt[N]{\prod_{1}^{N} \frac{1}{P(w_i | w_1 w_2 \ \dots w_{i-1})}}$$

Chain rule: allows us to decompose the probability into a product of component conditional probabilities

Perplexity: Chain Rule

$$PP(W) = P(w_1 w_2 \ \dots \ w_n)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1 \, w_2 \ \dots \ w_n)}}$$

$$= \sqrt[N]{\prod_{1}^{N} \frac{1}{P(w_i | w_1 w_2 \ \dots w_{i-1})}}$$

Chain rule: allows us to decompose the probability into a product of component conditional probabilities

And now what other Rule did we learn with
Language
Modeling that we can apply here?

Perplexity: Markov Assumption

$$PP(W) = P(w_1 w_2 \ \dots \ w_n)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1 \ w_2 \ \dots \ w_n)}}$$

$$= \sqrt[N]{\prod_{1}^{N} \frac{1}{P(w_i | w_1 w_2 \ \dots w_{i-1})}} = \sqrt[N]{\prod_{1}^{N} \frac{1}{P(w_i | w_{i-1})}}$$

Note

Because of the inverse: The higher the conditional probability of the word sequence the lower the perplexity

Therefore

Minimizing perplexity is equivalent to maximizing the test set probability according to the language model

Entropy

- Perplexity is based on the information-theoretic notion of cross-entropy
- Entropy =
  - measure of how much information there is in a particular grammar
  - how much information is there to predict a given langyage
- for a given N-gram grammar, how predictive is it about what the next word will be?

Entropy

$$H(X) = -\sum_{x \in X} p(x) log_2(p(x))$$

Technically, you can use any base, but with log base 2, resulting entropy is measured in bits

Entropy

$$H(X) = -\sum_{x \in X} p(x) log_2(p(x))$$

basic idea: the lower bound on the number of bits it would take to encode a certain piece of information

Horse example

- You are at Colonial Downs

- as you are walking around you want
- to send a short message to your bookie
- to tell him which horse to bet on

- you are paranoid that our wife/husband (who works for NSA) will find out … especially if you lose … which you won't … hopefully

Horse example

You could just use the binary representation of the horse's number:

Horse 1	000	Horse 5	100
Horse 2	001	Horse 6	101
Horse 3	010	Horse 7	110
Horse 4	011	Horse 8	111

If we spent the entire day at the track, it would be sending 3 bits per race

Horse example

You could just use the binary representation of the horse's number:

Horse 1	000	Horse 5	100
Horse 2	001	Horse 6	101
Horse 3	010	Horse 7	110
Horse 4	011	Horse 8	111

If we spent the entire day at the track,

it would be sending 3 bits per race

But you have not been doing your texting exercises

and worry that you will end up with a crippled thumb before the day is over

Horse example

Say we know the distribution of the bets placed on the horses and that we represent it as the prior probability of each horse

Horse 1	1/2	Horse 5	1/64
Horse 2	1/4	Horse 6	1/64
Horse 3	1/8	Horse 7	1/64
Horse 4	1/16	Horse 8	1/64

$$H(x) = -\sum_{x \in X} P(x) log_2 P(x) = -\frac{1}{2}\log\frac{1}{2} - \frac{1}{4} log\ \frac{1}{4} - \frac{1}{8} log \frac{1}{8} - \frac{1}{16} log \frac{1}{16} - 4\left(\frac{1}{64} log\ \frac{1}{64}\right) = 2\ bits$$

Horse example

Say we know the distribution of the bets placed on the horses and that we represent it as the prior probability of each horse

Horse 1	0	Horse 5	111100
Horse 2	10	Horse 6	111101
Horse 3	110	Horse 7	111110
Horse 4	1110	Horse 8	111111

$$H(x) = -\sum_{x\,\in X} P(x) log_2 P(x) = -\frac{1}{2}\log\frac{1}{2} - \frac{1}{4} log\ \frac{1}{4} - \frac{1}{8} log \frac{1}{8} - \frac{1}{16} log \frac{1}{16} - 4\left(\frac{1}{64} log\ \frac{1}{64}\right) = 2\ bits$$

Shannon Game

- Per letter entropy
  - Shannon reported 1.3 bits for 27 characters (26 letters + space)
    - Based on Jefferson the Virginian by Dumas Malone
  - Considered a bit low due to the corpus chosen

http://math.ucsd.edu/~crypto/java/ENTROPY/

Let's play a game

? <- What is the first letter?

What is the next letter?

B?

What is the next letter?

Be?

What is the next letter?

Bef?

What is the next letter?

Befo?

What is the next letter?

Befor?

What is the next letter?

Before?

What is the next letter?

Before ?

What is the next letter?

Before s?

What is the next letter?

Before sh?

What is the next letter?

Before she?

What is the next letter?

Before she ?

What is the next letter?

Before she w?

What is the next letter?

Before she wa?

What is the next letter?

Before she was?

What is the next letter?

Before she was ?

What is the next letter?

Before she was W?

What is the next letter?

Before she was Wo?

What is the next letter?

Before she was Won?

What is the next letter?

Before she was Wond?

What is the next letter?

Before she was Wonde?

What is the next letter?

Before she was Wonder?

What is the next letter?

Before she was Wonder ?

What is the next letter?

Before she was Wonder W?

What is the next letter?

Before she was Wonder Wo?

What is the next letter?

Before she was Wonder Wom?

What is the next letter?

Before she was Wonder Woma?

What is the next letter?

Before she was Wonder Woman?

What is the next letter?

Before she was Wonder Woman

?

What is the next letter?

Before she was Wonder Woman

s?

What is the next letter?

Before she was Wonder Woman

sh?

What is the next letter?

Before she was Wonder Woman

she?

What is the next letter?

Before she was Wonder Woman

she ?

What is the next letter?

Before she was Wonder Woman

she w?

What is the next letter?

Before she was Wonder Woman

she wa?

What is the next letter?

Before she was Wonder Woman

she was?

What is the next letter?

Before she was Wonder Woman

she was D?

What is the next letter?

Before she was Wonder Woman

she was Di?

What is the next letter?

Before she was Wonder Woman

she was Dia?

What is the next letter?

Before she was Wonder Woman

she was Dian?

What is the next letter?

Before she was Wonder Woman

she was Diana?

What is the next letter?

Before she was Wonder Woman

she was Diana

?

What is the next letter?

Before she was Wonder Woman

she was Diana

P?

What is the next letter?

Before she was Wonder Woman

she was Diana

Pr?

What is the next letter?

Before she was Wonder Woman

she was Diana

Pri?

What is the next letter?

Before she was Wonder Woman

she was Diana

Prin?

What is the next letter?

Before she was Wonder Woman

she was Diana

Princ?

What is the next letter?

Before she was Wonder Woman

she was Diana

Prince?

What is the next letter?

Before she was Wonder Woman

she was Diana

Princes?

What is the next letter?

Before she was Wonder Woman

she was Diana

Princess?

What is the next letter?

Before she was Wonder Woman

she was Diana

Princess ?

What is the next letter?

Before she was Wonder Woman

she was Diana

Princess o?

What is the next letter?

Before she was Wonder Woman

she was Diana

Princess of?

What is the next letter?

Before she was Wonder Woman

she was Diana

Princess of t?

What is the next letter?

Before she was Wonder Woman

she was Diana

Princess of th?

What is the next letter?

Before she was Wonder Woman

she was Diana

Princess of the?

What is the next letter?

Before she was Wonder Woman

she was Diana

Princess of the ?

What is the next letter?

Before she was Wonder Woman

she was Diana

Princess of the A?

What is the next letter?

Before she was Wonder Woman

she was Diana

Princess of the Am?

What is the next letter?

Before she was Wonder Woman

she was Diana

Princess of the Ama?

What is the next letter?

Before she was Wonder Woman

she was Diana

Princess of the Amaz?

What is the next letter?

Before she was Wonder Woman

she was Diana

Princess of the Amazo?

What is the next letter?

Before she was Wonder Woman

she was Diana

Princess of the Amazon

Probability information allows us to model language

Questions?