Ngrams …
+
Smoothing

Lecture # 5

7 February 2018

Recap of Ngram Modeling Example

Creating our Relative Frequency Tables

We are going to look at creating a bigram model

CORPUS

Aaron Hernandez helped lead the New England Patriots into the 2011 Super Bowl, but this weekend, the former NFL standout won’t.

.....

Unigram	Frequency
aaron	1
hernandez	1
helped	1
lead	1
the	3
new	1
england	1
patriots	1
into	1
2011	1
super	1
bowl	1
but	1
this	1
weekend	1
Former	1
NFL	1
standout	1
wo	1
n‘t	1
.	1

Unigram RAW FREQUENCIES

	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	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
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

Bigram RAW FREQUENCIES

	Aaron	hernandez	helped	lead	the	new	england	patriots	into	2011	super	bowl	but	this	weekend	former	nfl	standout	won	‘t	.
aaron	0	1	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
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
2011	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
aaron	1
hernandez	1
helped	1
lead	1
the	3
new	1
england	1
patriots	1
into	1
2011	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 | w1) = Freq(w1 w2) / Freq (w1)

	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
hernandez	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	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
2011	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

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

1/3

Unigram	Frequency
aaron	1
hernandez	1
helped	1
lead	1
the	**3**
new	1
england	1
patriots	1
into	1
2011	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 | w1) = Freq(w1 w2) / Freq (w1)

	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

RELATIVE FREQUENCIES

P(Aaron Hernandez helped lead the New England Patriots into the 2011 Super Bowl but this weekend the former NFL standout won't) =

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

P(Aaron Hernandez helped lead the New England Patriots into the 2011 Super Bowl but this weekend the former NFL standout won’t) =

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

This is called the *chain rule of probability* using *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(Aaron Hernandez helped lead the New England Patriots into the 2011 Super Bowl but this weekend the former NFL standout won't) =

P(aaron) *
P(hernandez|aaron) *
P(helped | aaron hernandez) *
P(lead | aaron hernadez helped) *
P(the | aaron hernadez helped lead) *
….
P(n't| aaron hernadez … standout wo)

Chain rule of probability

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

P(Aaron Hernandez helped lead the New England Patriots into the 2011 Super Bowl but this weekend the former NFL standout won’t) =

P(aaron) *
P(hernandez|aaron) *
P(helped | aaron hernandez) *
P(lead | aaron hernadez helped) *
P(the | aaron hernadez helped lead) *
….

calculating this is tough due to sparseness:

P(n’t| aaron hernadez … wo)

The chances of seeing “Aaron Hernandez helped lead the New England Patriots into the 2011 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

Aaron Hernandez helped lead the New England Patriots into the 2011 Super Bowl but this weekend the former NFL standout won't

- 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| aaron hernadez …standout wo) = P(n't | wo) **Using a bigram model**

P(n't| aaron hernadez …standout wo) = P(n't | standout wo) **Using a trigram model**

P(n't| aaron hernadez …standout wo) = P(n't | nfl standout wo) **Using a 4-gram model**

etc …

P(Aaron Hernandez helped lead the New England Patriots into the 2011 Super Bowl, but this weekend, the former NFL standout) =

P(aaron) *
P(hernandez|aaron) *
P(helped | Hernandez) *
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(Aaron Hernandez helped lead the New England Patriots into the 2011 Super Bowl, but this weekend, the former NFL standout) =

P(aaron) *
P(hernandez|aaron) *
P(helped | Hernandez) *
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

My Tokenization

Unigram	Frequency
aaron	1
hernandez	1
helped	1
lead	1
the	3
new	1
england	1
patriots	1
into	1
2011	1
super	1
bowl	1
but	1
this	1
weekend	1
Former	1
nfl	1
standout	1
Wo	1
n't	1
.	1

N = 23

P(unigram) = Freq(unigram) / N

N = 23

Unigram	Frequency
aaron	1
hernandez	1
helped	1
lead	1
the	3
new	1
england	1
patriots	1
into	1
2011	1
super	1
bowl	1
but	1
this	1
weekend	1
Former	1
nfl	1
standout	1
won	1
‘t	1
.	1

→

Unigram	P(unigram)
aaron	1/23 = 0.04
hernandez	1/23 = 0.04
helped	1/23 = 0.04
lead	1/23 = 0.04
the	3/23 = 0.13
new	1/23 = 0.04
england	1/23 = 0.04
patriots	1/23 = 0.04
into	1/23 = 0.04
2011	1/23 = 0.04
super	1/23 = 0.04
bowl	1/23 = 0.04
but	1/23 = 0.04
this	1/23 = 0.04
weekend	1/23 = 0.04
Former	1/23 = 0.04
nfl	1/23 = 0.04
standout	1/23 = 0.04
won	1/23 = 0.04
‘t	1/23 = 0.04
.	1/23 = 0.04

P(Aaron Hernandez helped lead the New England Patriots into the 2011 Super Bowl, but this weekend, the former NFL standout) =

P(aaron) *
P(Hernandez|aaron) *
P(helped | Hernandez) *
P(lead | helped) *
P(the | lead) *
P(new | the) *
P(england | new) *
….
P(standout| NFL)

Get this from our relative frequency tables

Does that make sense?

- 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

	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

Relative Frequency

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

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

	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

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

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

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

$$P(want \mid i) = \frac{827}{2533} = 0.33$$

	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

Relative Frequency Table

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

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

For bigrams

$How\ do\ we\ generalize\ this\ for\ all\ n-grams$

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

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

For bigrams

Any n-gram

$$P\left(w_n|w_{n-N+1}^{n-1}\right) = \frac{frequency(w_{n-N+1}^{n-1}, w_n)}{frequency(w_{n-N+1}^{n-1})}$$

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

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

For bigrams

Any n-gram

$$P\left(w_n|w_{n-N+1}^{n-1}\right) = \frac{frequency(w_{n-N+1}^{n-1}, w_n)}{frequency(w_{n-N+1}^{n-1})}$$

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

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

For bigrams

Any n-gram

$$P\left(w_n|w_{n-N+1}^{n-1}\right) = \frac{frequency(w_{n-N+1}^{n-1}, w_n)}{frequency(w_{n-N+1}^{n-1})}$$

Trigram Model: $P(unicorns|the\ magical) = \dfrac{frequency(the\ magical\ unicorns)}{frequency(the\ magical)}$

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

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

For bigrams

Any n-gram

$$P\left(w_n | w_{n-N+1}^{n-1}\right) = \frac{frequency(w_{n-N+1}^{n-1}, w_n)}{frequency(w_{n-N+1}^{n-1})}$$

Trigram Model: $P(unicorns|the\ magical) = \dfrac{frequency(the\ magical\ unicorns)}{frequency(the\ magical)}$

4-gram Model: $P(unicorns|the\ heroic\ magical) = \dfrac{frequency(the\ heroic\ magical\ unicorns)}{frequency(the\ heroic\ magical)}$

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

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

For bigrams

Any n-gram

$$P\left(w_n|w_{n-N+1}^{n-1}\right) = \frac{frequency(w_{n-N+1}^{n-1}, w_n)}{frequency(w_{n-N+1}^{n-1})}$$

Trigram Model: $P(unicorns|the\ magical) = \dfrac{frequency(the\ magical\ unicorns)}{frequency(the\ magical)}$

4-gram Model: $P(unicorns|the\ heroic\ magical) = \dfrac{frequency(the\ heroic\ magical\ unicorns)}{frequency(the\ heroic\ magical)}$

5-gram Model: $P(unicorns|the\ heroic\ unseen\ magical) = \dfrac{frequency(the\ heroic\ unseen\ magical\ unicorns)}{frequency(the\ heroic\ unseen\ magical)}$

To calculate these we use: two tables

N-gram table

and

(N-1)-gram tables

Any questions?

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

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)

	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

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

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

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

Which of the five choices do we choose

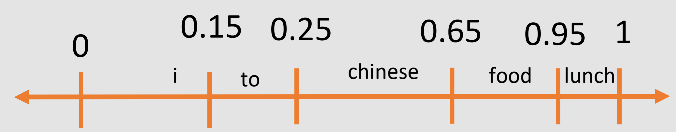
Randomly pick one based on its distribution

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

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 + 0.1 + 0.4 + 0.3 + 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 + 0.1 + 0.4 + 0.3 + 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

Pick a random number between zero and one

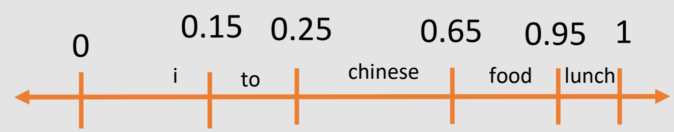
```
my $r = rand();
```

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"; }
```

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

$0.15 + 0.1 + 0.4 + 0.3 + 0.05 = 1$



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

Pick a random number between zero and one

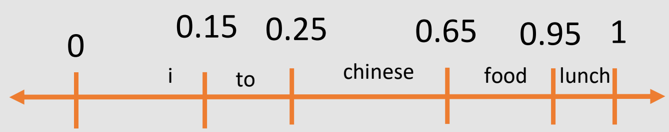
```
my $r = rand();
```

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"; }
```

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

0.15 + 0.1 + 0.4 + 0.3 + 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

So say our rand returned the value 0.6 what is our next word?

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?

	I am	to eat	chinese dish	…	lunch box
<start>	0.44	0.29	0.15	…	0.03

And before we move on
Perl’s Hashes of Hashes

```
my $n = shift;
my %hash = ();
while(<>) {
    chomp;
    my @array = split/\s+/;
    for my $i(0..$#array) {
        my $j = $i + $n - 1;
        my $first = $array[$i];
        my $ngram = "";
        if($j > $#array) { next; }
        for my $k ($i..$j) {
            $ngram .= "$array[$k] ";
        }
        chomp $ngram;
        $hash{$first}{$ngram}++;
    }
}
```

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	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

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

With the model, yes but in real life?

	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

$$P(I\ want\ to\ eat\ English\ Food) =$$

	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

$P(I\ want\ to\ eat\ English\ Food) =$

$P(i|<start>)$ *

$P(want|i)$ *

$P(to\ |\ want)$ *

$P(eat|to)$ *

$P(english|eat)$ *

$P(food|english)$ *

$P(<end>|food) = ?$

	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

$P(I\ want\ to\ eat\ English\ Food) =$

*P(i|<start>) **
*P(want|i) **
*P(to | want) **
*P(eat|to) **
*P(english|eat) **
*P(food|english)**
P(<end>|food) = ?

P(i|<start>) = 0.015
P(want|i) = 0.33
P(to | want) = 0.66
P(eat|to) = 0.28
P(english|eat) = 0
P(food|english) = 0
P(<end>|food) = 0.007

	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

$P(I\ want\ to\ eat\ English\ Food) =$

P(i|<start>) *
P(want|i) *
P(to | want) *
P(eat|to) *
P(english|eat) *
*P(food|english)**
P(<end>|food) = ?

P(i|<start>) = 0.015
P(want|i) = 0.33
P(to | want) = 0.66
P(eat|to) = 0.28
P(english|eat) = 0
P(food|english) = 0
P(<end>|food) = 0.007

Uh oh!

$P(I\ want\ to\ eat\ English\ Food) =$ *0*
.... ?

Sparicity

- Sparcity is a major problem for Maximum Likelihood Estimation

Recap MLE

- Sparsity is a major problem for Maximum Likelihood Estimation

This is MLE => $P(w_1^n) = \prod_{k=1}^{n} P(w_k|w_{k-1})$

MLE with code

- Sparsity is a major problem for Maximum Likelihood Estimation

This is MLE => $P(w_1^n) = \prod_{k=1}^{n} P(w_k | w_{k-1})$

```
my @w = {set of n words}
my $P_w = 1;
for my $k (1…n) {
     $P_w *= $rel_freq_table{$w[$k]}{$w[$k-1]}
}
```

MLE with example

- Sparsity is a major problem for Maximum Likelihood Estimation

This is MLE => $P(w_1^n) = \prod_{k=1}^{n} P(w_k | w_{k-1})$

$$
\begin{aligned}
P(the\ magical\ unicorn) = \\
&P(the) * \\
&P(magical|the) * \\
&P(unicorn|magical)
\end{aligned}
$$

MLE with example

- Sparsity is a major problem for Maximum Likelihood Estimation

This is MLE => $P(w_1^n) = \prod_{k=1}^{n} P(w_k|w_{k-1})$

$$P(the\ magical\ unicorn) =$$
$$P(the) *$$
$$P(magical|the) *$$
$$P(unicorn|magical)$$

These probabilities are referred to as Relative Frequency

Relative Frequency

- Sparsity is a major problem for Maximum Likelihood Estimation

This is MLE => $P(w_1^n) = \prod_{k=1}^{n} P(w_k|w_{k-1})$

This is Relative Frequency => $P(w_k|w_{k-1})$

$$P(unicorn|magical) = \frac{Frequency(magical\ unicorn)}{Frequency(magical)}$$

```
$rel_freq_table{$w[$k]}{$w[$k-1]}
```

Sparcity

	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

Because we don’t see <start> eat in the text does this mean it doesn’t occur ever

Is P(*<start> eat)* really zero

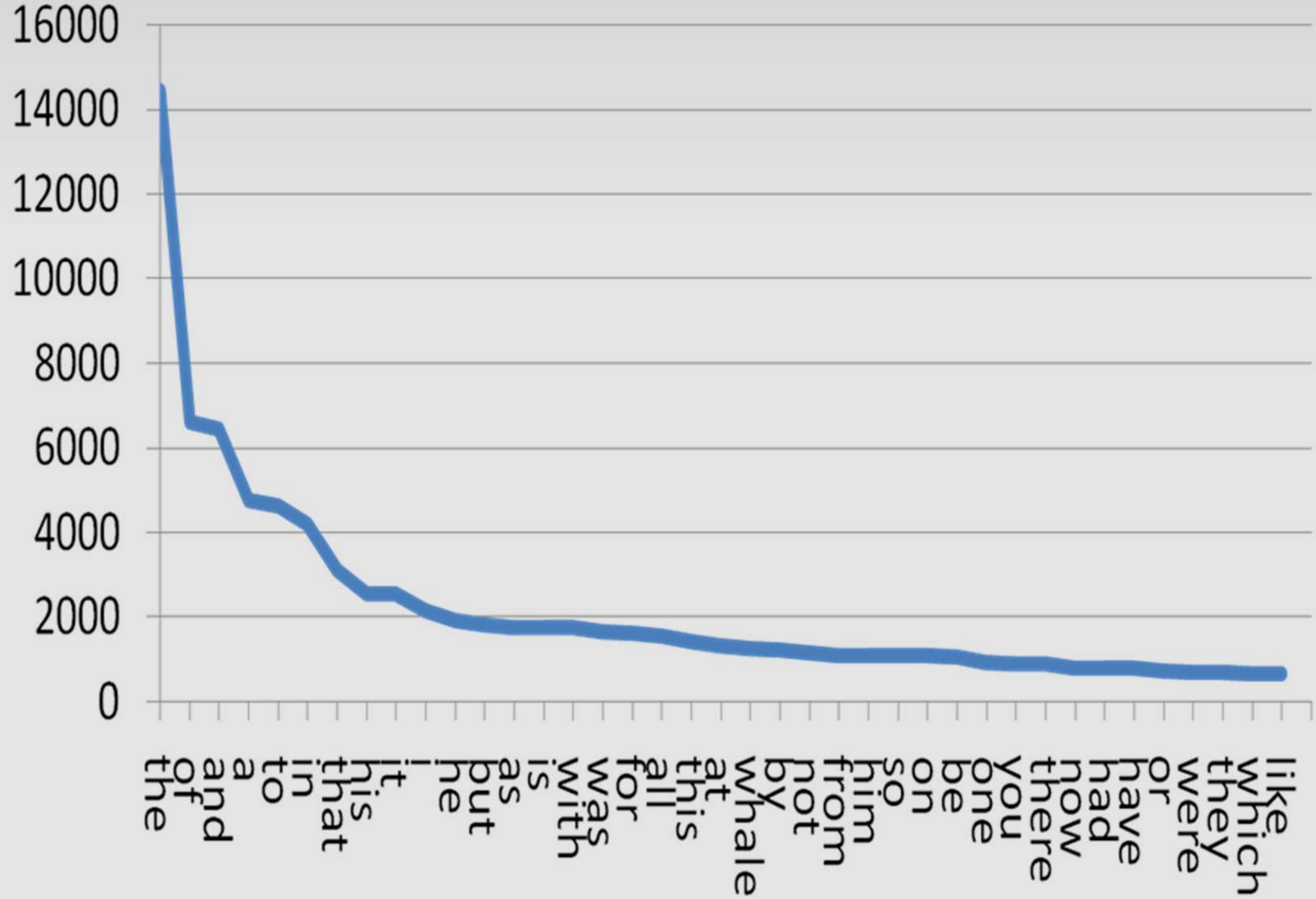
Smoothing

- 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

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

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

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

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

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

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

The book refers to “adjusted count”

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

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

Adjusted count

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

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

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

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

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

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

Adjusted count

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

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

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

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

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

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

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

Laplace Smoothing on Conditional Probabilities

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

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

V = the number of types in our corpus

Laplace Smoothing on Conditional Probabilities

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

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

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

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

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

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

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

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

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

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

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

V = 1446

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

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

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

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

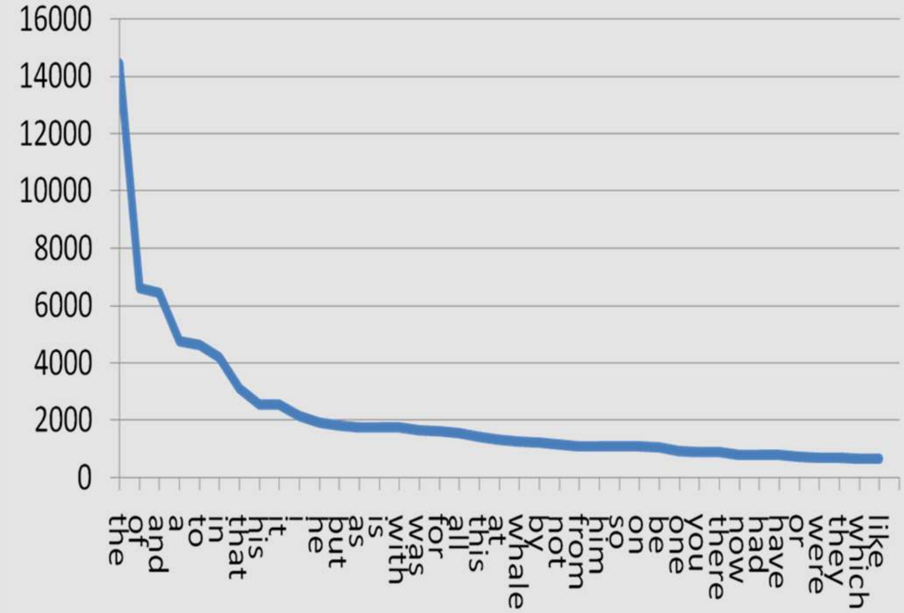
V = 1446

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

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

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

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

$$\left. frequency^*(w_i) = (frequency(w_i) + 1)\frac{N_{c+1}}{N_c} \right\} \text{Good Turing Smoothing}$$

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

But what about unseen bigrams

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

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

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

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?

Guesstimate

We know V (the vocabulary size), therefore
The total number of bigrams = V^2
So
$N_0 = V^2 - \#\ seen\ bigrams$

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

Frequency	Frequency(Frequency)
0	2081496
1	5315
2	1419
3	642
4	381
5	311
6	196
…	
2533	2
2534	2
….	….
M	1

	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
…	
2533	2
2534	2
….	….
M	1

	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

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

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

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

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

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

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
…	
2533	2
2534	2
….	….
M	1

	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

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

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

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

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

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

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

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
…	
2533	2
2534	2
….	….
M	1

	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

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

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

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

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

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

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

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

Frequency	Frequency(Frequency)
0	2081496
1	5315
2	1419
3	642
4	381
5	311
6	196
…	
2533	2
2534	2
….	….
M	1

	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

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

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

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

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

$$P^*(i\ to) = \frac{N_1}{N_0} = \frac{5315}{2081496} = 0.003$$

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

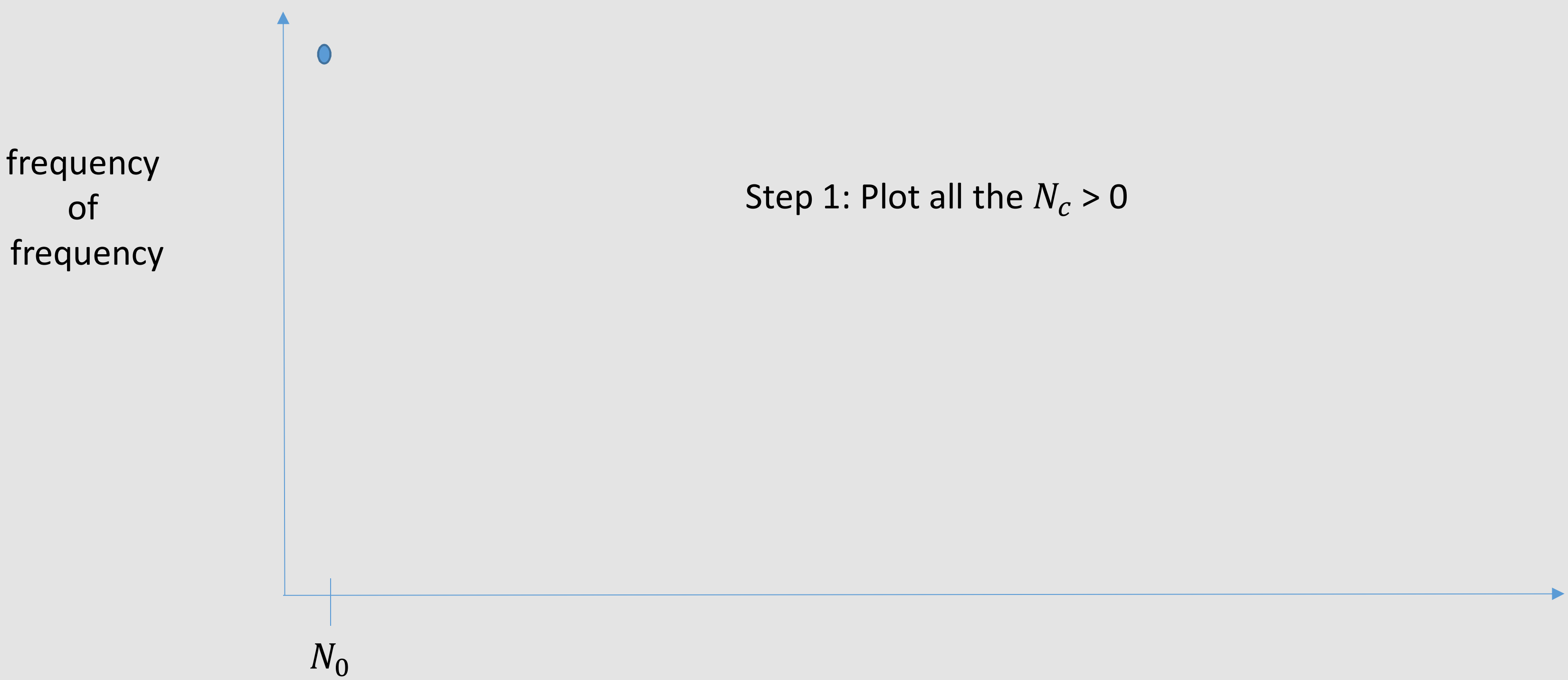
Frequency	Frequency(Frequency)
0	2081496
1	5315
2	1419
3	642
4	381
5	311
6	196
…	….
2533	2
2534	2
2535	0
….	….
M	1

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

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 $N_{c+1} = 0$

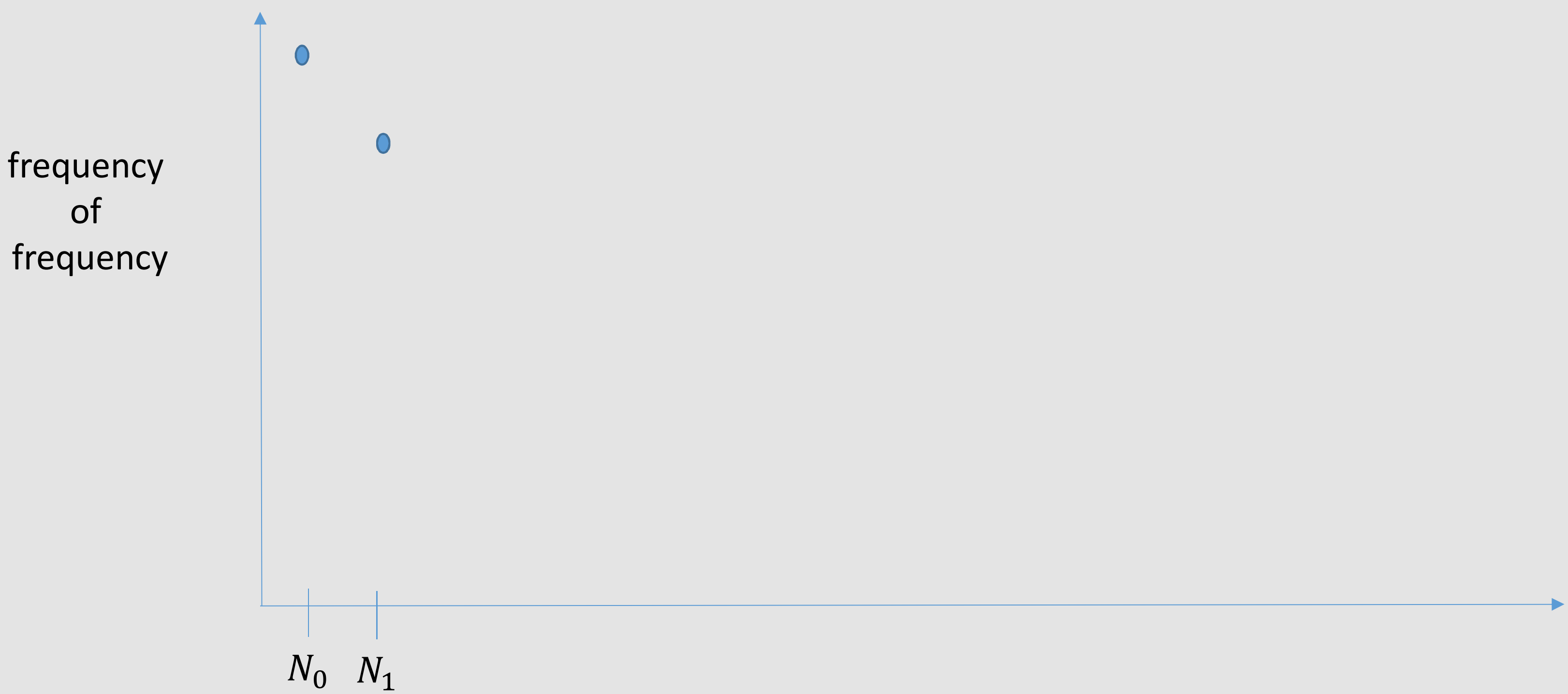
Frequency of frequency
Is the number of n-grams
That occurred N_{c+1} times



Step 1: Plot all the $N_c > 0$

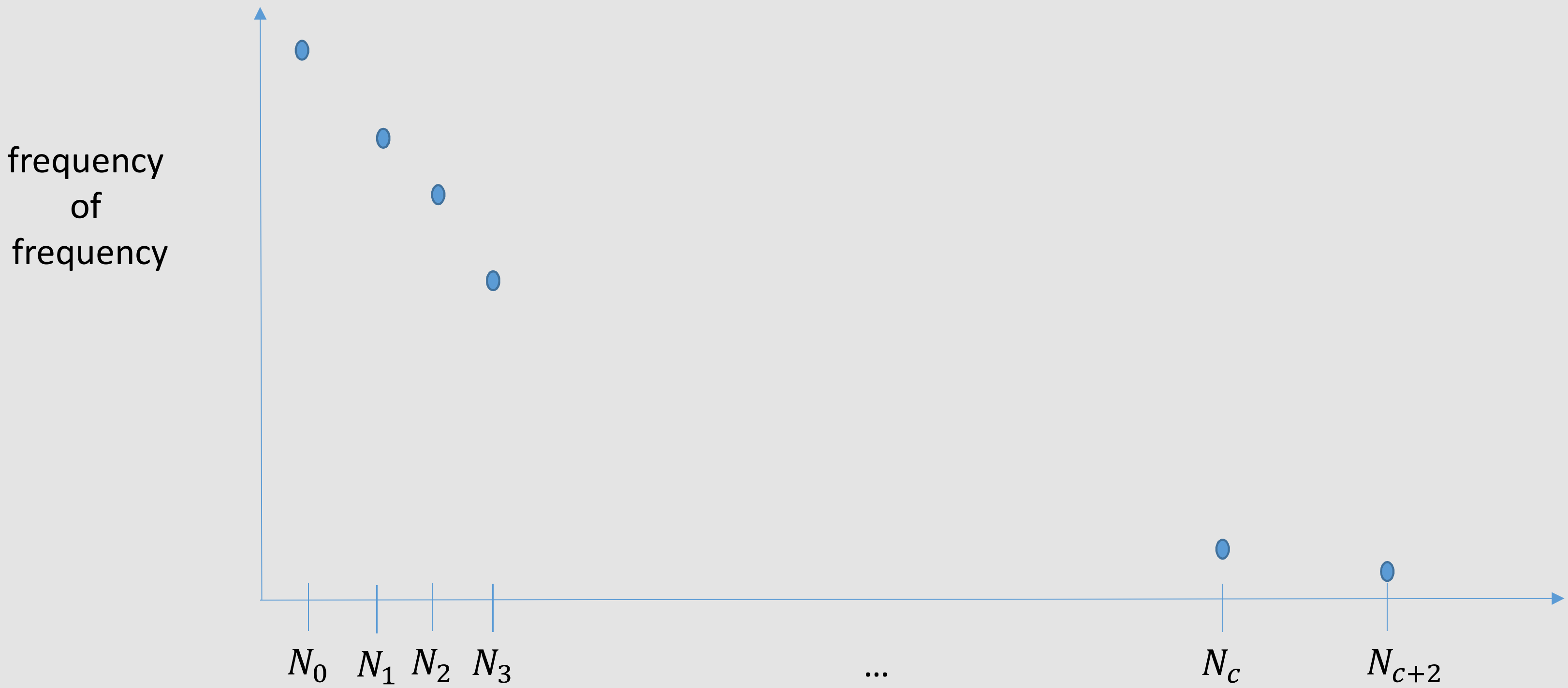
Estimating when $N_{c+1} = 0$

Frequency of frequency
Is the number of n-grams
That occurred N_{c+1} times



Frequency of frequency
Is the number of n-grams
That occurred N_{c+1} times

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

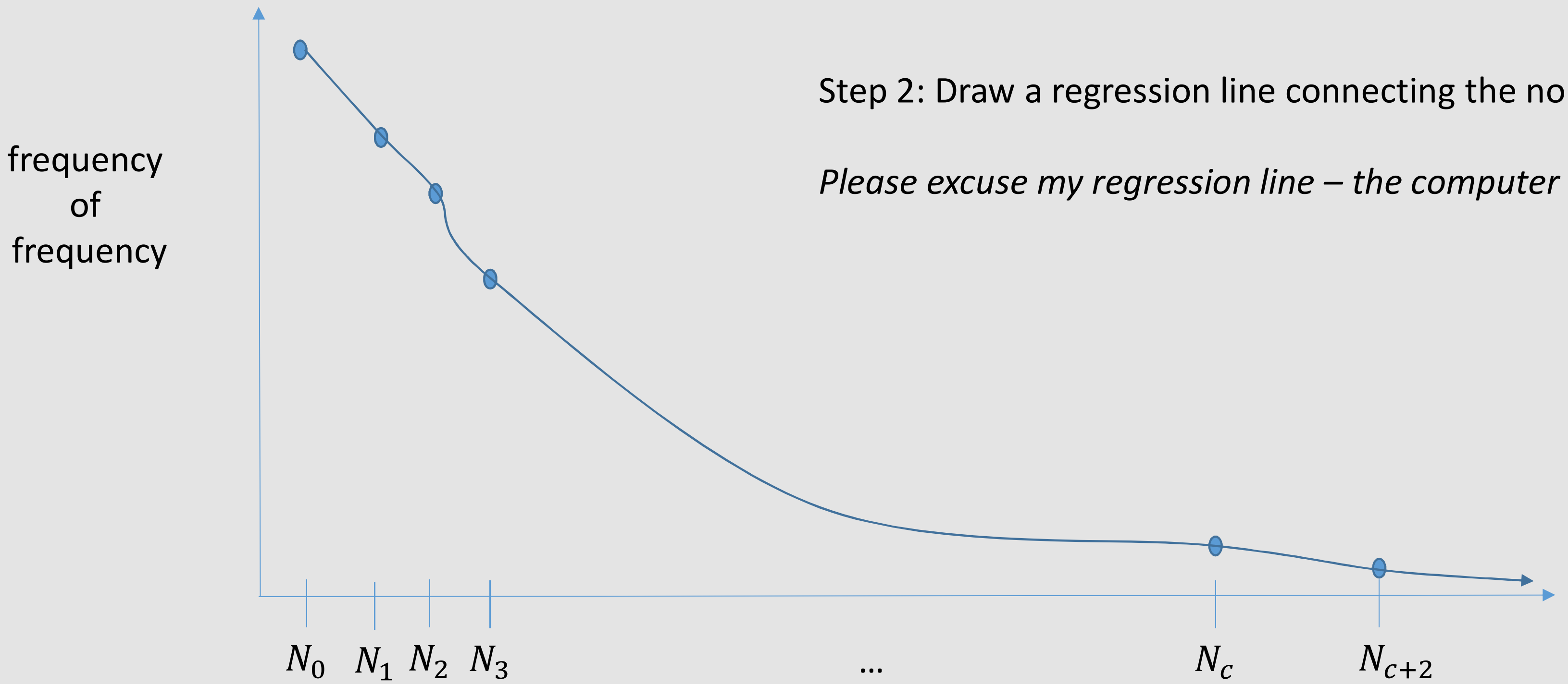


Estimating when $N_{c+1} = 0$

Frequency of frequency
Is the number of n-grams
That occurred N_{c+1} times

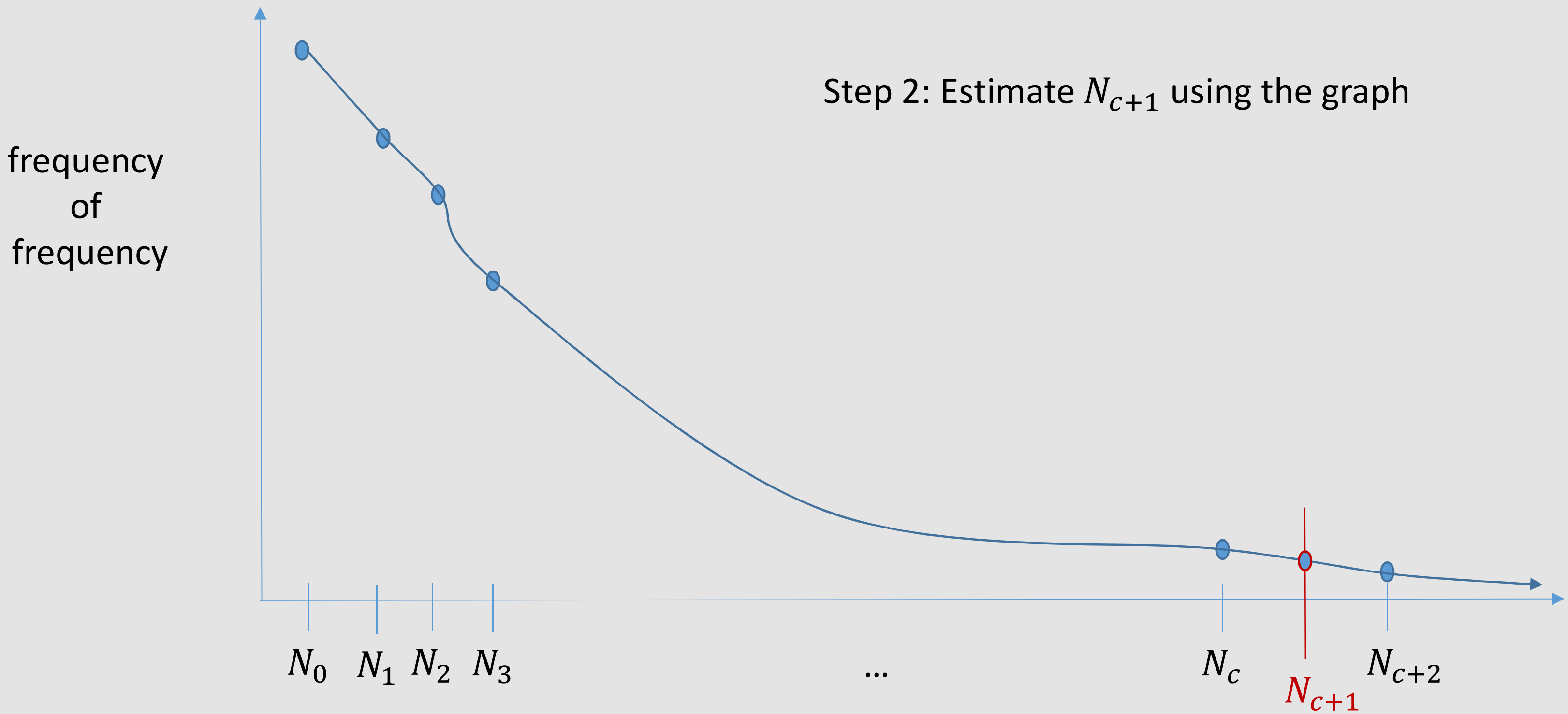
Step 2: Draw a regression line connecting the non-zero points

Please excuse my regression line – the computer does this better



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

Frequency of frequency
Is the number of n-grams
That occurred N_{c+1} times



Questions?