N-grams and smoothing; or, how language is (a bit) like the weather

Language Technology I

Asad Sayeed

Saarland University

Objectives for today

1. Explore the idea of sequences in language (n-grams).
2. Consider sequences as models of probability.
3. Handle the prediction of unseen items (smoothing).

Q: Does language have anything to do with the weather?

A: Yes. But first. . .

. . . a tongue-twister in English.

How much wood could a woodchuck chuck if a woodchuck could chuck wood?

. . . a tongue-twister in English.

How much wood could a woodchuck chuck if a woodchuck could chuck wood?

One possible answer:

As much wood as a woodchuck could chuck.

Can we calculate how likely that answer is?

Can we calculate how likely that answer is?

That depends . . . on what you mean by “likely”.

Can we calculate how likely that answer is?

That depends ... on what you mean by "likely".

To estimate the likelihood of an answer (in the form of a sentence), you need:

- An evidentiary basis.
  $\Rightarrow$ in modern statistical natural language processing, we use large corpora.

Can we calculate how likely that answer is?

That depends . . . on what you mean by "likely".

To estimate the likelihood of an answer (in the form of a sentence), you need:

- An evidentiary basis.
  $\Rightarrow$ in modern statistical natural language processing, we use large corpora.
- A theory that connects the evidence to the likelihood you're trying to estimate.

Can we calculate how likely that answer is?

That depends ...on what you mean by "likely".

To estimate the likelihood of an answer (in the form of a sentence), you need:

- An evidentiary basis.
  $\Rightarrow$ in modern statistical natural language processing, we use large corpora.
- A theory that connects the evidence to the likelihood you're trying to estimate.
  - Assume sentences are made of words.
  - So the probability of a sentence might have something to do with the probability of the words in the sentence.

Can we calculate how likely that answer is?

That depends . . . on what you mean by "likely".

To estimate the likelihood of an answer (in the form of a sentence), you need:

- An evidentiary basis.
  ⇒ in modern statistical natural language processing, we use large corpora.
- A theory that connects the evidence to the likelihood you're trying to estimate.
  - Assume sentences are made of words.
  - So the probability of a sentence might have something to do with the probability of the words in the sentence.
- A means to combine the pieces of evidence.
  ⇒ if words matter, then we need a theory of sentence structure from words.

Why do we want a likelihood?

Consider natural language processing systems in real life. E.g., machine translation:

- Translate "How much wood could a woodchuck chuck?" to French.
  - The word "could": possibility in French expressible with two different grammatical forms (*"peut"*/*"pourrait"*).
  - Choose better one *in context*.
  - Hard to do over all words deterministically ← years of effort to create the "rules", but never succeed.
- Countless other applications: such as answering a question. . . .

So how do we get the evidence?

Count words.

how much wood could a woodchuck chuck if a woodchuck could chuck wood ?

Assume that this is our corpus. Total number of words: 14 (incl. the “?”).

So how do we get the evidence?

Count words.

how much wood could a woodchuck chuck if a woodchuck could chuck wood ?

Assume that this is our corpus. Total number of words: 14 (incl. the "?").

word type	token count
a	2
chuck	2
could	2
how	1
if	1

word type	token count
much	1
wood	2
woodchuck	2
?	1

So how do we get the evidence?

Count words.

how much wood could a woodchuck chuck if a woodchuck could chuck wood ?

Assume that this is our corpus. Total number of words: 14 (incl. the "?").

word type	token count	p(word)
a	2	0.14
chuck	2	0.14
could	2	0.14
how	1	0.07
if	1	0.07

word type	token count	p(word)
much	1	0.07
wood	2	0.14
woodchuck	2	0.14
?	1	0.07

Then calculate probability per type of word as count/14.

Calculate the probability of expressions.

word type	token count	p(word)
a	2	0.14
chuck	2	0.14
could	2	0.14
how	1	0.07
if	1	0.07

word type	token count	p(word)
much	1	0.07
wood	2	0.14
woodchuck	2	0.14
?	1	0.07

The joint probability of multiple words: how likely they are to occur in the same text.

$$p(w_1, w_2, \ldots) = p(w_1)p(w_2)\ldots$$

Calculate some joint probabilities:

- $p(\text{if,woodchuck}) =$
- $p(\text{wood,woodchuck}) =$
- $p(\text{how,could,a}) =$

Calculate the probability of expressions.

word type	token count	p(word)
a	2	0.14
chuck	2	0.14
could	2	0.14
how	1	0.07
if	1	0.07

word type	token count	p(word)
much	1	0.07
wood	2	0.14
woodchuck	2	0.14
?	1	0.07

The joint probability of multiple words: how likely they are to occur in the same text.

$$p(w_1, w_2, \ldots) = p(w_1)p(w_2)\ldots$$

Calculate some joint probabilities:

- $p(\text{if,woodchuck}) = 0.07 \times 0.14 = 0.01$
- $p(\text{wood,woodchuck}) = 0.14 \times 0.14 = 0.02$
- $p(\text{how,could,a}) = 0.07 \times 0.14 \times 0.14 = 0.001$

Calculating the probability of expressions.

word type	token count	p(word)
a	2	0.14
chuck	2	0.14
could	2	0.14
how	1	0.07
if	1	0.07

word type	token count	p(word)
much	1	0.07
wood	2	0.14
woodchuck	2	0.14
?	1	0.07

Now we can calculate the joint probability of our answer.

As much wood as a woodchuck could chuck.

- $p(\text{as,much,wood,as,a,woodchuck,could,chuck}) =$

Calculating the probability of expressions.

word type	token count	p(word)
a	2	0.14
chuck	2	0.14
could	2	0.14
how	1	0.07
if	1	0.07

word type	token count	p(word)
much	1	0.07
wood	2	0.14
woodchuck	2	0.14
?	1	0.07

Now we can calculate the joint probability of our answer.

As much wood as a woodchuck could chuck.

- p(as,much,wood,as,a,woodchuck,could,chuck) $=$ 0 x . . .

Calculating the probability of expressions.

word type	token count	p(word)
a	2	0.14
chuck	2	0.14
could	2	0.14
how	1	0.07
if	1	0.07

word type	token count	p(word)
much	1	0.07
wood	2	0.14
woodchuck	2	0.14
?	1	0.07

Now we can calculate the joint probability of our answer.

As much wood as a woodchuck could chuck.

- p(as,much,wood,as,a,woodchuck,could,chuck) $=$ 0 x ...
- Uh oh: there's no "as" in our probability table.
  $\Rightarrow$ we will get to missing items soon.

Calculating the probability of expressions.

word type	token count	p(word)
a	2	0.14
chuck	2	0.14
could	2	0.14
how	1	0.07
if	1	0.07

word type	token count	p(word)
much	1	0.07
wood	2	0.14
woodchuck	2	0.14
?	1	0.07

Now we can calculate the joint probability of our answer.

As much wood as a woodchuck could chuck.

- p(as,much,wood,as,a,woodchuck,could,chuck) $=$ 0 x . . .
- Uh oh: there's no "as" in our probability table.
  $\Rightarrow$ we will get to missing items soon.
- So, try p(much,wood,a,woodchuck,could,chuck) $=$

Calculating the probability of expressions.

word type	token count	p(word)
a	2	0.14
chuck	2	0.14
could	2	0.14
how	1	0.07
if	1	0.07

word type	token count	p(word)
much	1	0.07
wood	2	0.14
woodchuck	2	0.14
?	1	0.07

Now we can calculate the joint probability of our answer.

As much wood as a woodchuck could chuck.

- p(as,much,wood,as,a,woodchuck,could,chuck) $= 0 \times \ldots$
- Uh oh: there's no "as" in our probability table.
  $\Rightarrow$ we will get to missing items soon.
- So, try p(much,wood,a,woodchuck,could,chuck) $=$
  $0.07 \times 0.14 \times 0.14 \times 0.14 \times 0.14 \times 0.14 = 3.76\text{e-}05$

Words come in an order.

Calculating the joint probability of unigrams (single words): is it a good model?

Words come in an order.

Calculating the joint probability of unigrams (single words): is it a good model?

Backwards...

chuck could woodchuck a as wood much as

...is not an English sentence.

Words come in an order.

Calculating the joint probability of unigrams (single words): is it a good model?

Backwards. . .

chuck could woodchuck a as wood much as

. . . is not an English sentence.

- Joint unigram probability: the same, no matter what, as "as much wood as a woodchuck could chuck".

Words come in an order.

Calculating the joint probability of unigrams (single words): is it a good model?

Backwards. . .

chuck could woodchuck a as wood much as

. . . is not an English sentence.

- Joint unigram probability: the same, no matter what, as "as much wood as a woodchuck could chuck".
- We definitely don't want that to be true. So our theory must include sequences.

And this is what language has to do with the weather.

What was the weather like two years ago in Holland?

Average temperature at Amsterdam Schiphol:

18.11.2014
8 C

And what was it the day before that?

Average temperature at Amsterdam Schiphol:

17.11.2014	18.11.2014
10 C	8 C

And before that?

Average temperature at Amsterdam Schiphol:

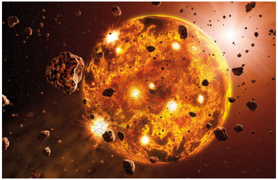
16.11.2014	17.11.2014	18.11.2014
9 C	10 C	8 C

It's as though we know something about the next day from the previous days!

But how many days do we need?

Surely not to the beginning of the Earth!

Average temperature at Amsterdam Schiphol:

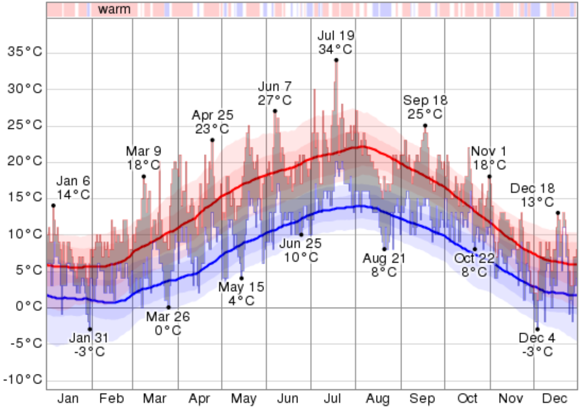


. . .

16.11.2014	17.11.2014	18.11.2014
9 C	10 C	8 C

We have expectations about changes.

We know that yesterday is a good clue about today.
Temperatures in Amsterdam in 2014:



The daily temperature is a Markov process.

Let T_d = temperature T on day d.
We can represent the probability conditionally.

Probability of today's temperature given universe

$p(T_d | T_{d-1}, T_{d-2}, \ldots, T_{d-\infty})$

The daily temperature is a Markov process.

Let T_d = temperature T on day d.
We can represent the probability conditionally.

Probability of today's temperature given 2 previous days

$p(T_d | T_{d-1}, T_{d-2}, \ldots, T_{d-\infty}) \approx p(T_d | T_{d-1}, T_{d-2})$

But we only need a few days to give us a trend. So we make a Markov assumption.

The daily temperature is a Markov process.

Let T_d = temperature T on day d.
We can represent the probability conditionally.

Probability of today's temperature given 2 previous days

$$p(T_d | T_{d-1}, T_{d-2}, \ldots, T_{d-\infty}) \approx p(T_d | T_{d-1}, T_{d-2})$$

But we only need a few days to give us a trend. So we make a Markov assumption.
Then we can calculate the joint probability of a sequence of days:

Markov chain

$$p(T_d, T_{d-1}, T_{d-2}) =$$
$$p(T_d | T_{d-1}, T_{d-2}) p(T_{d-1} | T_{d-2}, T_{d-3}) p(T_{d-2} | T_{d-3}, T_{d-4})$$

Getting Markovian with language.

Let's make a Markov assumption over sentences. So how many words previous to "chuck" do we need?

As much wood as a woodchuck could **chuck**.

Getting Markovian with language.

Let's make a Markov assumption over sentences. So how many words previous to "chuck" do we need?

As much wood as a woodchuck could **chuck**.

- "could" is an auxiliary that selects for a verb.

Getting Markovian with language.

Let's make a Markov assumption over sentences. So how many words previous to "chuck" do we need?

As much wood as a woodchuck could **chuck**.

- "could" is an auxiliary that selects for a verb.
- "woodchuck" – maybe. We're asking if woodchucks can chuck, it's in the corpus.

Getting Markovian with language.

Let's make a Markov assumption over sentences. So how many words previous to "chuck" do we need?

As much wood as a woodchuck could **chuck**.

- "could" is an auxiliary that selects for a verb.
- "woodchuck" – maybe. We're asking if woodchucks can chuck, it's in the corpus.
- "much"? No, probably not.

Getting Markovian with language.

Let's make a Markov assumption over sentences. So how many words previous to "chuck" do we need?

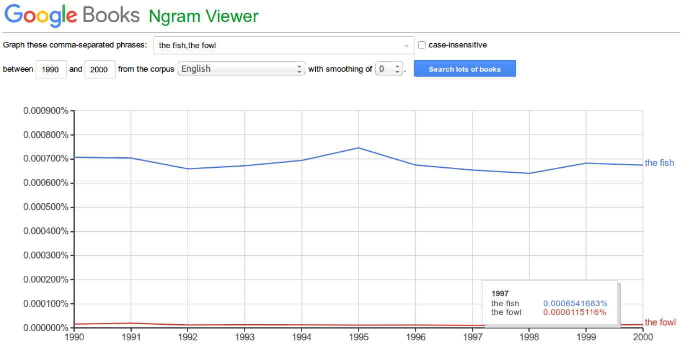
As much wood as a woodchuck could **chuck**.

- "could" is an auxiliary that selects for a verb.
- "woodchuck" – maybe. We're asking if woodchucks can chuck, it's in the corpus.
- "much"? No, probably not.

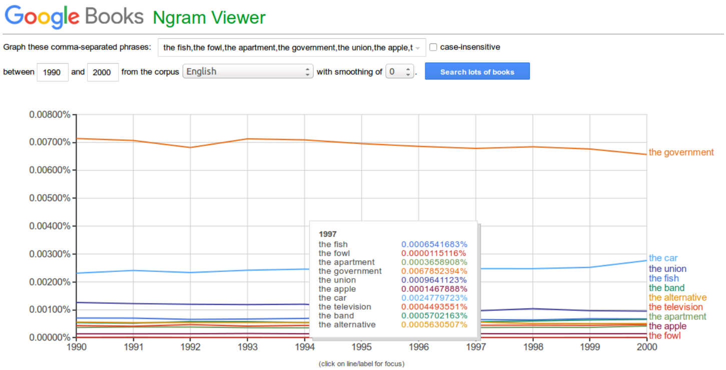
Two words back seems to be a common choice.

We can check a bigger corpus.

Leave aside the woodchucks for a moment. Let's try a couple of 2-word expressions. "The fish" vs "the fowl.".
The Google Books Ngram viewer:



But lots of things follow “the”.

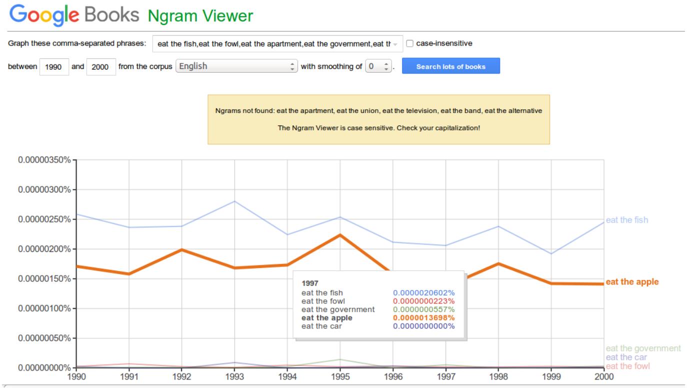


It's not hugely informative...

...because the whole category of nouns can follow "the".

It’s not hugely informative. . .

. . . because the whole category of nouns can follow “the”.
So what if we add another word, “eat”:



The additional word is hugely informative!

So this is a way language is **not** like the weather.

The additional word is hugely informative!

So this is a way language is **not** like the weather.

- Sure, tomorrow will resemble today, in terms of temperature.
  - But knowing what happened yesterday doesn't drastically change the estimate.

The additional word is hugely informative!

So this is a way language is **not** like the weather.

- Sure, tomorrow will resemble today, in terms of temperature.
  - But knowing what happened yesterday doesn't drastically change the estimate.
- But make your **bi**gram into a **tri**gram:
  - The distribution radically changes.
  - "eat" is very informative.

We can even look for 4-grams.

Thus we just call these *n*-grams, for any *n*.
So when we look for 4-grams starting with “quickly eat the fish/apple/car”?

We can even look for 4-grams.

Thus we just call these *n*-grams, for any *n*.
So when we look for 4-grams starting with "quickly eat the fish/apple/car"?

Google Ngrams doesn't find anything! (for 1990-2000).

We can even look for 4-grams.

Thus we just call these *n*-grams, for any *n*.
So when we look for 4-grams starting with "quickly eat the fish/apple/car"?

Google Ngrams doesn't find anything! (for 1990-2000).

Not even "quickly eat the apple"!

We can even look for 4-grams.

Thus we just call these *n*-grams, for any *n*.
So when we look for 4-grams starting with "quickly eat the fish/apple/car"?

Google Ngrams doesn't find anything! (for 1990-2000).

Not even "quickly eat the apple"!

It's not always the case that trigrams work, but they're often practical because of sparsity.

Getting back to our woodchucks

(start) How much wood could a woodchuck chuck if a woodchuck could chuck wood ?

Getting back to our woodchucks

(start) How much wood could a woodchuck chuck if a woodchuck could chuck wood ?

Since our "corpus" is short, let's stick to bigrams.

bigram	count
(start) how	
how much	
much wood	
wood could	
could a	
a woodchuck	
woodchuck chuck	

bigram	count
chuck if	
if a	
woodchuck could	
could chuck	
chuck wood	
wood ?	

Getting back to our woodchucks

(start) How much wood could a woodchuck chuck if a woodchuck could chuck wood ?

Since our "corpus" is short, let's stick to bigrams.

bigram	count
(start) how	1
how much	1
much wood	1
wood could	1
could a	1
a woodchuck	2
woodchuck chuck	1

bigram	count
chuck if	1
if a	1
woodchuck could	1
could chuck	1
chuck wood	1
wood ?	1

With a total of 14.

Then, bigram probability.

We write this as $p(w_2|w_1)$.

We want to calculate them so we can calculate our “answer”.

word	As	much	wood	as	a	woodchuck	could	chuck
$p(w_2\|w_1)$								

Then, bigram probability.

We write this as $p(w_2|w_1)$.
We want to calculate them so we can calculate our "answer".

word	As	much	wood	as	a	woodchuck	could	chuck
$p(w_2\|w_1)$								

So what's the probability of "chuck" given "could"?

Then, bigram probability.

We write this as $p(w_2|w_1)$.
We want to calculate them so we can calculate our "answer".

word	As	much	wood	as	a	woodchuck	could	chuck
$p(w_2\|w_1)$								

So what's the probability of "chuck" given "could"?

- Collect all the bigram occurrences of "could".

Then, bigram probability.

We write this as $p(w_2|w_1)$.
We want to calculate them so we can calculate our "answer".

word	As	much	wood	as	a	woodchuck	could	chuck
$p(w_2|w_1)$								

So what's the probability of "chuck" given "could"?

- Collect all the bigram occurrences of "could".
  w_1 = "could a" + "could chuck" = 2

Then, bigram probability.

We write this as $p(w_2|w_1)$.
We want to calculate them so we can calculate our "answer".

word	As	much	wood	as	a	woodchuck	could	chuck
$p(w_2\|w_1)$								

So what's the probability of "chuck" given "could"?

- Collect all the bigram occurrences of "could".
  w_1 = "could a" + "could chuck" = 2
- Only one of them is "chuck".

Then, bigram probability.

We write this as $p(w_2|w_1)$.

We want to calculate them so we can calculate our "answer".

word	As	much	wood	as	a	woodchuck	could	chuck
$p(w_2\|w_1)$								0.5

So what's the probability of "chuck" given "could"?

- Collect all the bigram occurrences of "could".
  w_1 = "could a" + "could chuck" = 2
- Only one of them is "chuck".
  $p(\text{chuck}|\text{could}) = 1/2 = 0.5$

Then, bigram probability.

We write this as $p(w_2|w_1)$.
We want to calculate them so we can calculate our “answer”.

word	As	much	wood	as	a	woodchuck	could	chuck
$p(w_2\|w_1)$								0.5

So what's the probability of “chuck” given “could”?

- Collect all the bigram occurrences of “could”.
  $w_1 =$ “could a” + “could chuck” $= 2$
- Only one of them is “chuck”.
  $p(\text{chuck}|\text{could}) = 1/2 = 0.5$

Then we just work our way backwards.

Data sparsity strikes again.

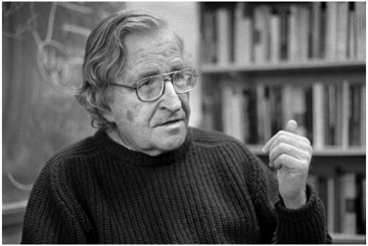
word	As	much	wood	as	a	woodchuck	could	chuck
$p(w_2\|w_1)$	0	undef	1	0	undef	1.0	0.5	0.5

"As" is nowhere in the model.

So we can't compute p("As much wood as a woodchuck could chuck").

The data just doesn't contain what we need.

So is this guy right?



'But it must be recognized that the notion of "probability of a sentence" is an entirely useless one, under any known interpretation of this term.'

Could be. . .

Could be. . .

. . . for linguistic *theory*.

Could be. . .

. . . for linguistic *theory*.

- Humans are creative: what we can say is not directly connected to what we've heard in the past.

Could be. . .

. . . for linguistic *theory*.

- Humans are creative: what we can say is not directly connected to what we've heard in the past.
- How we connect previous knowledge to language production/comprehension is still an open question.

Could be. . .

. . . for linguistic *theory*.

- Humans are creative: what we can say is not directly connected to what we've heard in the past.
- How we connect previous knowledge to language production/comprehension is still an open question.
- But for the time being, we can "cheat".

Could be. . .

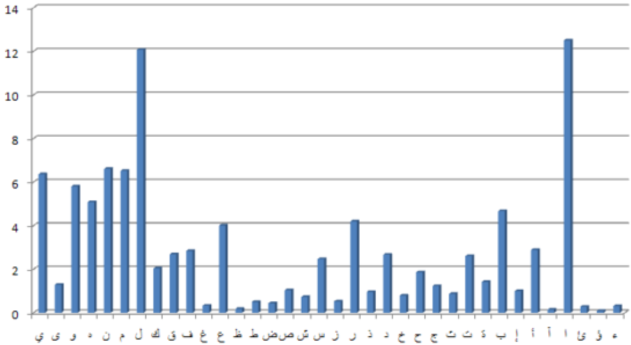
. . . for linguistic *theory*.

- Humans are creative: what we can say is not directly connected to what we've heard in the past.
- How we connect previous knowledge to language production/comprehension is still an open question.
- But for the time being, we can "cheat".

Which brings us to the topic of smoothing.

So what is smoothing?

Consider frequencies in language as a histogram.

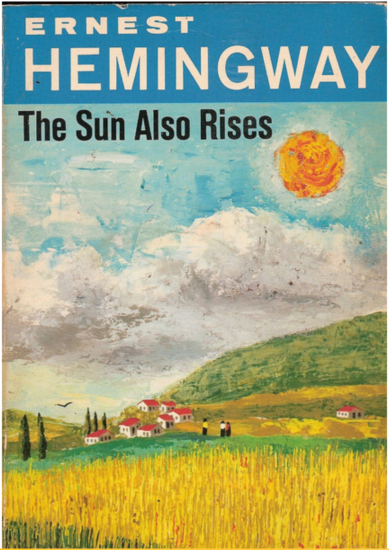


Counts that are zero make things “bumpy”.

...and it's just hard to do probability on bumpy distributions (as we've seen).

So what we want is to "smooth" the distribution.

Which gives me an opportunity to talk about the sun.



What does the sun have to do with anything?

What does the sun have to do with anything?

It has a lot!

What does the sun have to do with anything?

It has a lot!

What is the chance of it not rising tomorrow?

What does the sun have to do with anything?

It has a lot!

What is the chance of it not rising tomorrow?

- It's always risen before.

What does the sun have to do with anything?

It has a lot!

What is the chance of it not rising tomorrow?

- It's always risen before.
- But the chance of it not rising is not zero!
  - "Hard" science fiction space disasters – can happen!

What does the sun have to do with anything?

It has a lot!

What is the chance of it not rising tomorrow?

- It's always risen before.
- But the chance of it not rising is not zero!
  - "Hard" science fiction space disasters – can happen!
- **Laplace:** how to reason about this? Fudge the count of the never-seen eventuality.

And hence, Laplace/add-one smoothing.

That's it. Just add some constant. A simple smoothing.

And hence, Laplace/add-one smoothing.

That's it. Just add some constant. A simple smoothing.

So can we solve our little sparsity problem?

word	As	much	wood	as	a	woodchuck	could	chuck
$p(w_2 \mid w_1)$	0	undef	1	0	undef	1.0	0.5	0.5

And hence, Laplace/add-one smoothing.

That's it. Just add some constant. A simple smoothing.
So can we solve our little sparsity problem?

word	As	much	wood	as	a	woodchuck	could	chuck
$p(w_2\|w_1)$	0	undef	1	0	undef	1.0	0.5	0.5

Sure we can!

Laplace smoothing

$p'(w_2|w_1) = \frac{count(w_1 w_2)+d}{count(w_1)+dV}$

Often we pick $d = 1$, which is why it's "add-one".

Let's just add some bigram counts.

We'll pick a constant of 1 and add the bigrams we need. Everything else gets incremented by 1.

bigram	count
(start) how	2
how much	2
much wood	2
wood could	2
could a	2
a woodchuck	3
woodchuck chuck	2

bigram	count
chuck if	2
if a	2
woodchuck could	2
could chuck	2
chuck wood	2
wood ?	2
(start) as	1
as much	1
would as	1
as a	1

And $V = 17$, so we can calculate our denominator.

We can calculate our smoothed probabilities.

word	As	much	wood	as	a	woodchuck	could	chuck
$p(w_2\|w_1)$	0	undef	1	0	undef	1.0	0	0.5
$p'(w_2\|w_1)$	0.06							

Calculate "(start) as":

$p'(\text{as}|(\text{start})) = \text{count'}(\text{"start as"}) \; / \; (\text{count}(\text{"start"}) + 17)$

$= 1/(1+17) = 0.06$ (Must use original count of start.)

So now we have a sequence of bigram probabilities.

word	As	much	wood	as	a	woodchuck	could	chuck
$p'(w_2\|w_1)$	0.06	0.06	0.11	0.05	0.06	0.17	0.11	0.11

We can now compute the probability of the sentence!

So now we have a sequence of bigram probabilities.

word	As	much	wood	as	a	woodchuck	could	chuck
$p'(w_2\|w_1)$	0.06	0.06	0.11	0.05	0.06	0.17	0.11	0.11

We can now compute the probability of the sentence!
(Which is 2.44e-9, a lot lower than just multiplying the nonzero unigrams, which was 3.76e-05.)

So we've succeeded in making zero nonzero!

There are still some issues for you to think about:

So we've succeeded in making zero nonzero!

There are still some issues for you to think about:

- Is there a reason to believe that add-one smoothing is fair?
  - We did this independently of the likelihood of individual words.
- Add-one smoothing is not typically used.

So we've succeeded in making zero nonzero!

There are still some issues for you to think about:

- Is there a reason to believe that add-one smoothing is fair?
  - We did this independently of the likelihood of individual words.
- Add-one smoothing is not typically used.
  - It's very aggressive! It "steals" too much probability mass from things we've actually seen.

So we've succeeded in making zero nonzero!

There are still some issues for you to think about:

- Is there a reason to believe that add-one smoothing is fair?
  - We did this independently of the likelihood of individual words.
- Add-one smoothing is not typically used.
  - It's very aggressive! It "steals" too much probability mass from things we've actually seen.
  - Not all *hapax legomena* are equally likely.

So we've succeeded in making zero nonzero!

There are still some issues for you to think about:

- Is there a reason to believe that add-one smoothing is fair?
  - We did this independently of the likelihood of individual words.
- Add-one smoothing is not typically used.
  - It's very aggressive! It "steals" too much probability mass from things we've actually seen.
  - Not all *hapax legomena* are equally likely.
- **Are there better ways to do it?**

Discounting: an interlude

Before we move on…there's another way to look at add-one: in terms of a **discount**.

Add-one discount formula

$$d_c = \frac{c^*}{c}$$

This tells us how much we "stole" from a word with original count c in order to give to the unseen forms.

Good-Turing discounting

In add-one smoothing:

Good-Turing discounting

In add-one smoothing:

- We pretend we've seen unknown n-grams **once**.

Good-Turing discounting

In add-one smoothing:

- We pretend we've seen unknown n-grams **once**.
- We don't take into account what effect it will have on the other n-grams.

Good-Turing discounting

In add-one smoothing:

- We pretend we've seen unknown n-grams **once**.
- We don't take into account what effect it will have on the other n-grams.
- We compensate for it in a VERY crude manner.

Good-Turing discounting

In add-one smoothing:

- We pretend we've seen unknown n-grams **once**.
- We don't take into account what effect it will have on the other n-grams.
- We compensate for it in a VERY crude manner.

We can do better:

- We can START by estimating how likely it is we're going to see something new.

Good-Turing discounting

Insight

The number of things we've never seen can be estimated from the number of things we've seen only once.

Good-Turing discounting

Insight

The number of things we've never seen can be estimated from the number of things we've seen only once.

But THEN, that means that we have to steal probability from everyone else.

Good-Turing discounting

Insight

The number of things we've never seen can be estimated from the number of things we've seen only once.

But THEN, that means that we have to steal probability from everyone else.

- How to do that fairly?

Good-Turing discounting

Insight

The number of things we've never seen can be estimated from the number of things we've seen only once.

But THEN, that means that we have to steal probability from everyone else.

- How to do that fairly?
- We need to reestimate the probability of **everything** by the same principle.

Good-Turing discounting

Key concept: **frequency of frequency**.

- N_c — how often n-grams of frequency c appear in the corpus.

Good-Turing discounting

Key concept: **frequency of frequency**.

- N_c — how often n-grams of frequency c appear in the corpus.

$$N_c = \sum_{x:\mathrm{count}(x)=c} 1$$

Good-Turing discounting

Key concept: **frequency of frequency**.

- N_c — how often n-grams of frequency c appear in the corpus.

$$N_c = \sum_{x:\mathrm{count}(x)=c} 1$$

Then we can compute revised counts for everything.

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

Good-Turing discounting

So how do we get the probability of missing items?

Good-Turing discounting

So how do we get the probability of missing items?

$$P^*_{GT}(\mathrm{count}(w) = 0) = \frac{N_1}{N}$$

Where N is the total number of tokens.
(I'll leave proof as an exercise for the reader.)

Issues with Good-Turing

Some things to note:

- Assumes that the distribution of each bigram is binomial.

Issues with Good-Turing

Some things to note:

- Assumes that the distribution of each bigram is binomial.
  - If you have a vocab V, then the number of bigrams is V^2, so what happens to OOV words?

Issues with Good-Turing

Some things to note:

- Assumes that the distribution of each bigram is binomial.
  - If you have a vocab V, then the number of bigrams is V^2, so what happens to OOV words?
- What happens when we don't know N_{c+1}?

Issues with Good-Turing

Some things to note:

- Assumes that the distribution of each bigram is binomial.
  - If you have a vocab V, then the number of bigrams is V^2, so what happens to OOV words?
- What happens when we don't know N_{c+1}?
  - We have to smooth out the frequency of frequency counts!

Issues with Good-Turing

Some things to note:

- Assumes that the distribution of each bigram is binomial.
  - If you have a vocab V, then the number of bigrams is V^2, so what happens to OOV words?
- What happens when we don't know N_{c+1}?
  - We have to smooth out the frequency of frequency counts!
- We don't necessarily discount things where the count is big: probably reliable.

Issues with Good-Turing

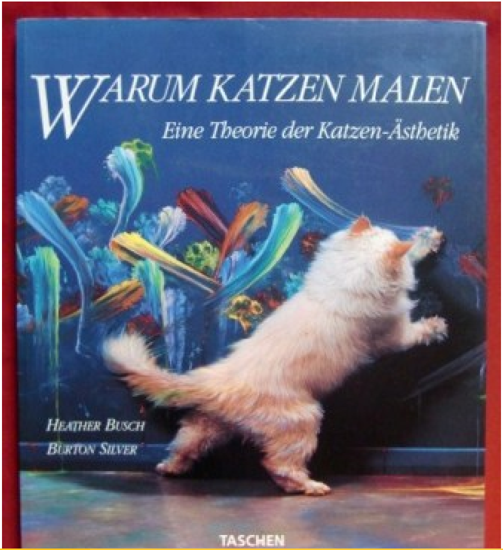
Some things to note:

- Assumes that the distribution of each bigram is binomial.
  - If you have a vocab V, then the number of bigrams is V^2, so what happens to OOV words?
- What happens when we don't know N_{c+1}?
  - We have to smooth out the frequency of frequency counts!
- We don't necessarily discount things where the count is big: probably reliable.
  - But everything must sum to 1!

So far we've focused mostly on bigrams.
But what about bigger "grams"?

This raises an important question.

Why cats smooth.



Higher-order n-grams

What about “why cats smooth”?

Higher-order n-grams

What about "why cats smooth"?

- Not frequent enough to appear in Google n-grams.

Higher-order n-grams

What about “why cats smooth”?

- Not frequent enough to appear in Google n-grams.
- But maybe the bigrams will help us: “why cats” and “cats smooth”.

Higher-order n-grams

What about “why cats smooth”?

- Not frequent enough to appear in Google n-grams.
- But maybe the bigrams will help us: “why cats” and “cats smooth”.

And even if bigrams don’t help us, maybe some other combination will get us a more realistic estimate.

Interpolation

So what we want to find is $P(w_n|w_{n-2}w_{n-1})$ — that's the definition of the probability of a trigram.

Interpolation

So what we want to find is $P(w_n|w_{n-2}w_{n-1})$ — that's the definition of the probability of a trigram.

Linear interpolation

$$\hat{P}(w_n|w_{n-2}w_{n-1}) = \lambda_1 P(w_n|w_{n-2}w_{n-1})$$
$$+\lambda_2 P(w_n|w_{n-1}) + \lambda_3 P(w_n)$$

Interpolation

So what we want to find is $P(w_n|w_{n-2}w_{n-1})$ — that's the definition of the probability of a trigram.

Linear interpolation

$$\hat{P}(w_n|w_{n-2}w_{n-1}) = \lambda_1 P(w_n|w_{n-2}w_{n-1})$$
$$+\lambda_2 P(w_n|w_{n-1}) + \lambda_3 P(w_n)$$

Then we just need to learn the λ weights (by EM or any other linear regression trick).

Interpolation

So what we want to find is $P(w_n|w_{n-2}w_{n-1})$ — that's the definition of the probability of a trigram.

Linear interpolation

$$\hat{P}(w_n|w_{n-2}w_{n-1}) = \lambda_1 P(w_n|w_{n-2}w_{n-1})$$
$$+\lambda_2 P(w_n|w_{n-1}) + \lambda_3 P(w_n)$$

Then we just need to learn the λ weights (by EM or any other linear regression trick).

We can also make the weights context-dependent by making them relative to bigrams.

Backoff

An even better way: **Backoff**
For example, Katz (haha!) backoff.

$$P_{bo}(w_i|w_{i-n+1}\cdots w_{i-1}) = \begin{cases} d_{w_{i-n+1}\cdots w_i}\frac{C(w_{i-n+1}\ldots w_{i-1}w_i)}{C(w_{i-n+1}\cdots w_{i-1})} \text{ if } C(w_{i-n+1}\cdots w_i) > k \\ \alpha_{w_{i-n+1}\cdots w_{i-1}} P_{bo}(w_i|w_{i-n+2}\cdots w_{i-1}) \text{ otherwise} \end{cases}$$

$$\beta_{w_{i-n+1}\cdots w_{i-1}} = 1 - \sum_{\{w_i : C(w_{i-n+1}\cdots w_i) > k\}} d_{w_{i-n+1}\cdots w_i} \frac{C(w_{i-n+1}\ldots w_{i-1}w_i)}{C(w_{i-n+1}\cdots w_{i-1})}$$

$$\alpha_{w_{i-n+1}\cdots w_{i-1}} = \frac{\beta_{w_{i-n+1}\cdots w_{i-1}}}{\sum_{\{w_i : C(w_{i-n+1}\cdots w_i) \leq k\}} P_{bo}(w_i|w_{i-n+2}\cdots w_{i-1})}$$

Katz backoff

...but that's kind of ugly-looking. What it's really saying is that:

Katz backoff

...but that's kind of ugly-looking. What it's really saying is that:

- Use the discounted weight if the count of the n-gram in question is acceptably large.

Katz backoff

…but that's kind of ugly-looking. What it's really saying is that:

- Use the discounted weight if the count of the n-gram in question is acceptably large.
- If not, use the n-minus-1-gram's count, adjusted by a special α factor that adjusts the count to include the mass you lost by excluding one word.

Katz backoff

…but that's kind of ugly-looking. What it's really saying is that:

- Use the discounted weight if the count of the n-gram in question is acceptably large.
- If not, use the n-minus-1-gram's count, adjusted by a special α factor that adjusts the count to include the mass you lost by excluding one word.
- You calculate THAT using all the n-minus-1-grams that involve the word you dropped.

So just a couple of final thoughts.

- Is there a better way to estimate n-gram probabilities in the first place?

So just a couple of final thoughts.

- Is there a better way to estimate n-gram probabilities in the first place?
- Are n-grams and smoothing a good model of sentences? Where are they deficient?

The End.