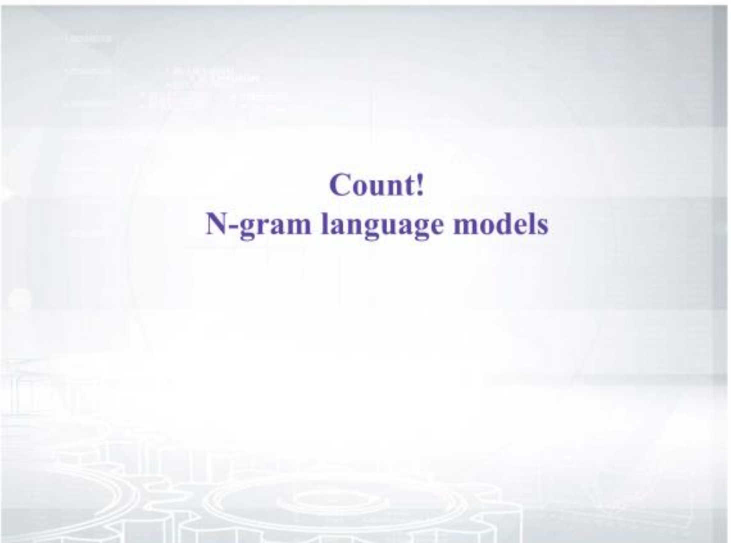
Language modeling: it's all about counting

Friday, June 15, 2018 10:42 AM

count n-gram



Language modeling

house

rat

This is the …

did

malt

What's the probability of the next word?

$$p(\textit{house} \mid \textit{this is the}) \;=\; ?$$

Toy corpus

This is the house that Jack built.
This is the malt
That lay in the house that Jack built.
This is the rat,
That ate the malt
That lay in the house that Jack built.
This is the cat,
That killed the rat,
That ate the malt
That lay in the house that Jack built.

$p(house \mid this\ is\ the) =$

Toy corpus

This is the house *that Jack built.*
This is the malt
That lay in the house that Jack built.
This is the rat*,*
That ate the malt
That lay in the house that Jack built.
This is the cat*,*
That killed the rat,
That ate the malt
That lay in the house that Jack built.

$p(house \mid this\ is\ the) =$

Toy corpus

This is the house *that Jack built.*
This is the malt
That lay in the house that Jack built.
This is the rat*,*
That ate the malt
That lay in the house that Jack built.
This is the cat*,*
That killed the rat,
That ate the malt
That lay in the house that Jack built.

$p(house \mid this\ is\ the) =$

Toy corpus

This is the house *that Jack built.*
This is the malt
That lay in the house that Jack built.
This is the rat*,*
That ate the malt
That lay in the house that Jack built.
This is the cat*,*
That killed the rat,
That ate the malt
That lay in the house that Jack built.

$$p(house \mid this\ is\ the) = \frac{c(this\ is\ the\ house)}{c(this\ is\ the\ ...)} = \frac{1}{4}$$

What is the probability of "Jack" given the previous word "that": p(Jack|that) = ?

Toy corpus:

This is the house that Jack built. This is the malt that lay in the house that Jack built. This is the rat, that ate the malt that lay in the house that Jack built. This is the cat, that killed the rat, that ate the malt that lay in the house that Jack built.

0.4

Correct Response
c(that) = 10, c(that Jack) = 4

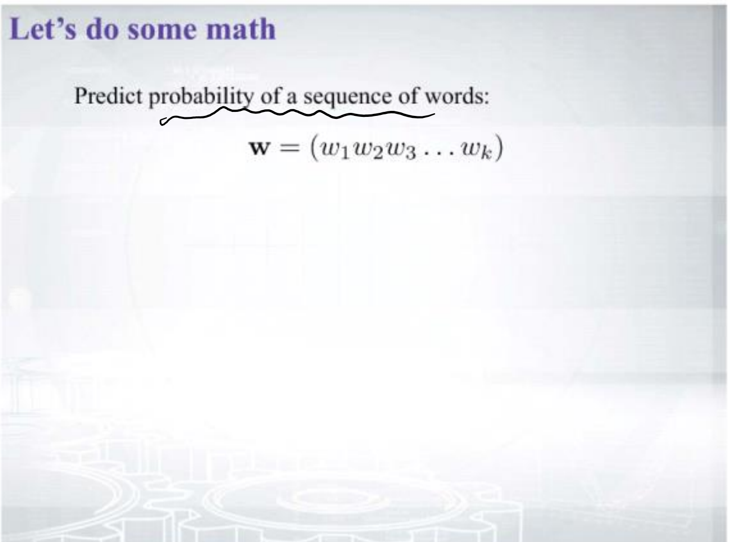
Toy corpus

This is the house *that Jack built.*
This is the malt
That lay in the house that Jack built.
This is the rat*,*
That ate the malt
That lay in the house that Jack built.
This is the cat*,*
That killed the rat,
That ate the malt
That lay in the house that Jack built.

4-grams

$$p(\textit{house} \mid \textit{this is the}) = \frac{c(\textit{this is the house})}{c(\textit{this is the ...})} = \frac{1}{4}$$

language model


Let's do some math
Predict probability of a sequence of words:
$\mathbf{w} = (w_1 w_2 w_3 \dots w_k)$

Let’s do some math

Predict probability of a sequence of words:

$$\mathbf{w} = (w_1 w_2 w_3 \dots w_k)$$

- **Chain rule:**

$$p(\mathbf{w}) = p(w_1)p(w_2|w_1)\dots p(w_k|w_1 \dots w_{k-1})$$

Let's do some math

Predict probability of a sequence of words:

$$\mathbf{w} = (w_1 w_2 w_3 \ldots w_k)$$

- **Chain rule:**

$$p(\mathbf{w}) = p(w_1)p(w_2|w_1)\ldots p(w_k|\cancel{w_1 \ldots} w_{k-1})$$

- **Markov assumption:**

$$p(w_i|w_1 \ldots w_{i-1}) = p(w_i|w_{i-n+1} \ldots w_{i-1})$$

Bigram language model

So that's what we get for n = 2:

$$p(\mathbf{w}) = p(w_1)p(w_2|w_1)\dots p(w_k|w_{k-1})$$

the prob. of all seq of the same length sums to 1

Bigram language model

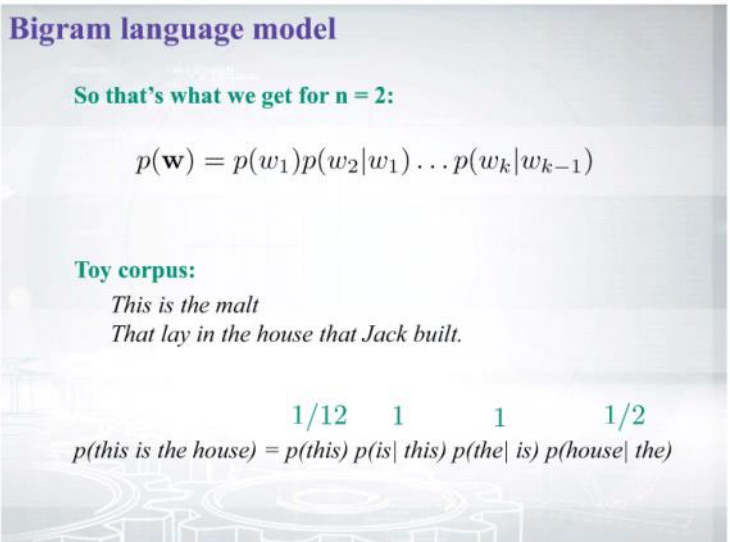
So that's what we get for n = 2:

$$p(\mathbf{w}) = p(w_1)p(w_2|w_1)\dots p(w_k|w_{k-1})$$

Toy corpus:

This is the malt
That lay in the house that Jack built.

p(this is the house) = p(this) p(is| this) p(the| is) p(house| the)


Bigram language model
So that's what we get for n = 2:
$$p(\mathbf{w}) = p(w_1)p(w_2|w_1)\dots p(w_k|w_{k-1})$$
Toy corpus:
This is the malt
That lay in the house that Jack built.
$1/12$ 1 1 $1/2$
p(this is the house) = p(this) p(is| this) p(the| is) p(house| the)

Bigram language model

So that's what we get for n = 2:

$$p(\mathbf{w}) = \cancel{p(w_1)}p(w_2|w_1)\dots p(w_k|w_{k-1})$$
$$p(w_1|start)$$

Toy corpus:

This is the malt
That lay in the house that Jack built.

$$\begin{array}{ccccc} & 1/2 & 1 & 1 & 1/2 \\ p(\textit{this is the house}) = & p(\textit{this}) & p(\textit{is}|\ \textit{this}) & p(\textit{the}|\ \textit{is}) & p(\textit{house}|\ \textit{the}) \end{array}$$

Bigram language model

So that's what we get for n = 2:

$$p(\mathbf{w}) = \cancel{p(w_1)}\, p(w_2|w_1) \ldots p(w_k|w_{k-1})$$

$$p(w_1|start)$$

Bigram language model

So that's what we get for n = 2:

$$p(\mathbf{w}) = \cancel{p(w_1)} p(w_2|w_1) \ldots p(w_k|w_{k-1})$$
$$p(w_1|start)$$

It's normalized separately for each sequence length!

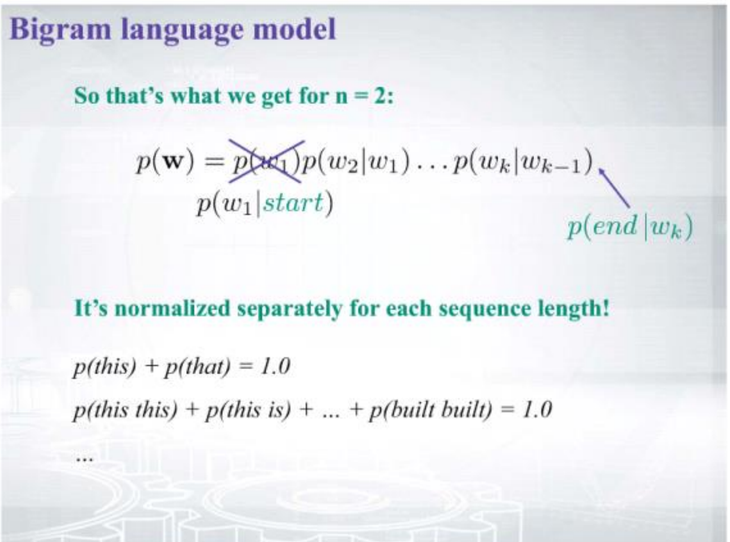
each sequence length

prob. sum 1 for each length

p(this) + p(that) = 1.0

p(this this) + p(this is) + … + p(built built) = 1.0

…


Bigram language model
So that's what we get for n = 2:
$p(\mathbf{w}) = \cancel{p(w_1)}p(w_2|w_1)\dots p(w_k|w_{k-1})$
$p(w_1|start)$
$p(end|w_k)$
It's normalized separately for each sequence length!
p(this) + p(that) = 1.0
p(this this) + p(this is) + … + p(built built) = 1.0
…

Let’s check the model

_ dog _

_ dog cat tiger _

_ cat dog cat _

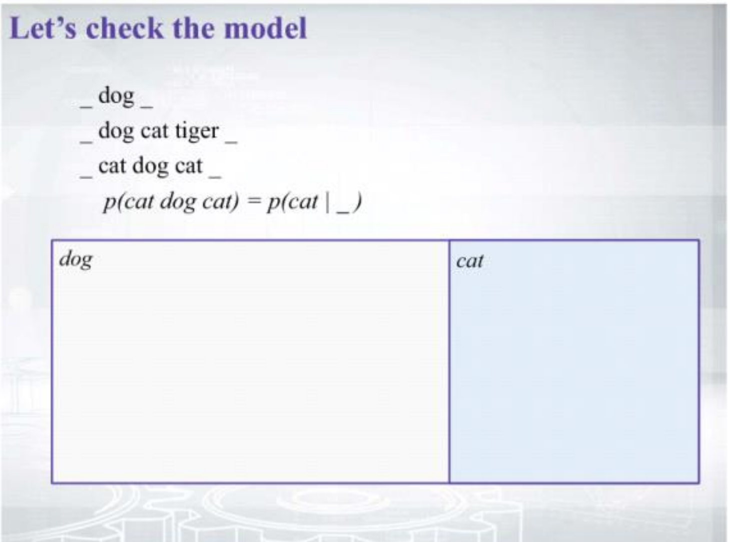
p(cat dog cat) =

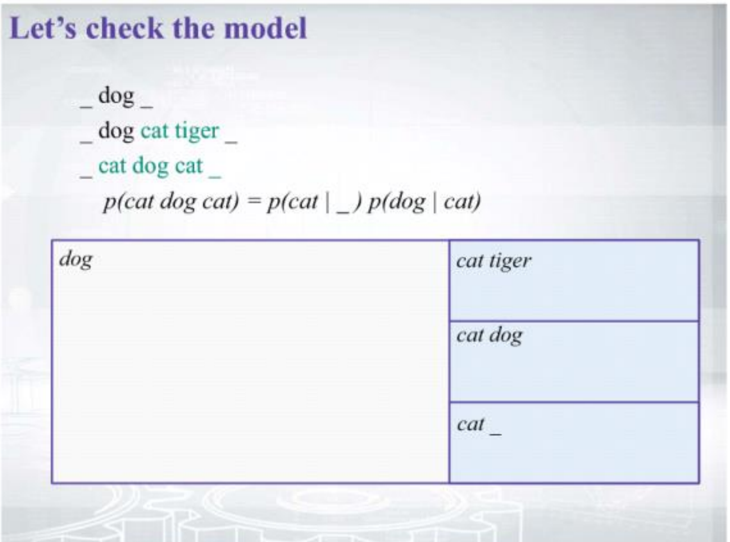
Let's check the model

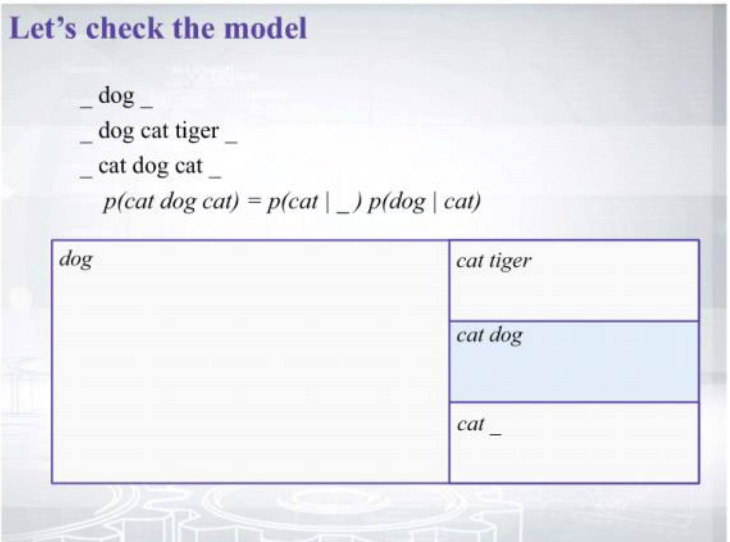
_ dog _
_ dog cat tiger _
_ cat dog cat _

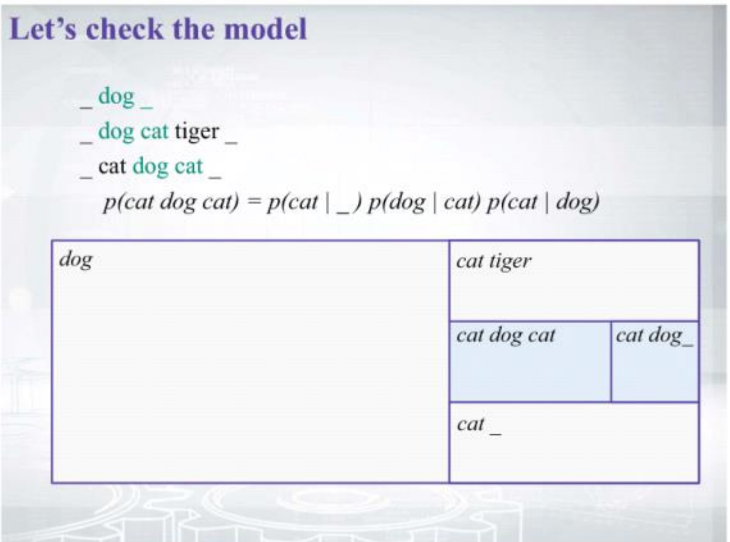
$p(cat\ dog\ cat) = p(cat \mid \_)$

dog	*cat*


Let's check the model
_ dog _
_ dog cat tiger _
_ cat dog cat _
p(cat dog cat) = p(cat | _)
dog
cat


Let’s check the model
_ dog _
_ dog cat tiger _
_ cat dog cat _
p(cat dog cat) = p(cat | _) p(dog | cat)
dog
cat tiger
cat dog
cat _


Let's check the model
_ dog _
_ dog cat tiger _
_ cat dog cat _
p(cat dog cat) = p(cat | _) p(dog | cat)
dog
cat tiger
cat dog
cat _


Let's check the model
_ dog _
_ dog cat tiger _
_ cat dog cat _
p(cat dog cat) = p(cat | _) p(dog | cat) p(cat | dog)
dog
cat tiger
cat dog cat
cat dog_
cat _

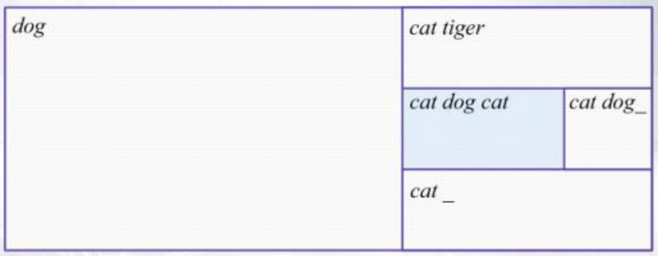
Let's check the model

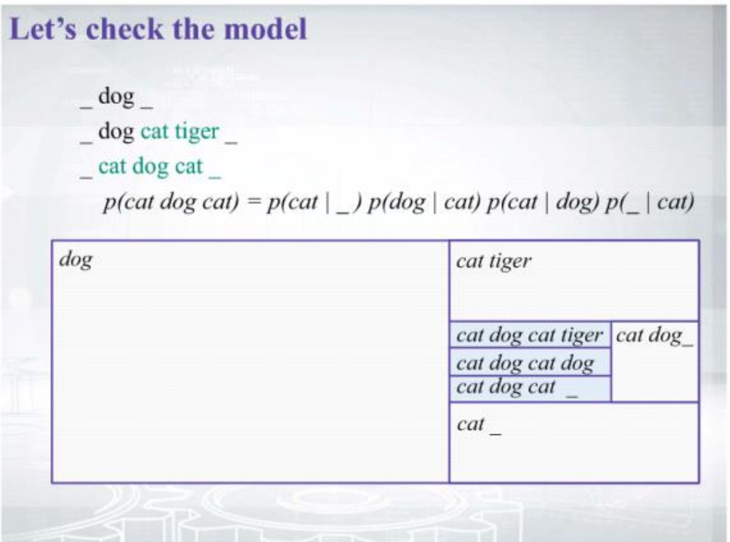
_ dog _

_ dog cat tiger _

_ cat dog cat _

$p(cat\ dog\ cat) = p(cat \mid \_)\ p(dog \mid cat)\ p(cat \mid dog)$




Let's check the model
_ dog _
_ dog cat tiger _
_ cat dog cat _
p(cat dog cat) = p(cat | _) p(dog | cat) p(cat | dog) p(_ | cat)
dog
cat tiger
cat dog cat tiger
cat dog_
cat dog cat dog
cat dog cat _
cat _

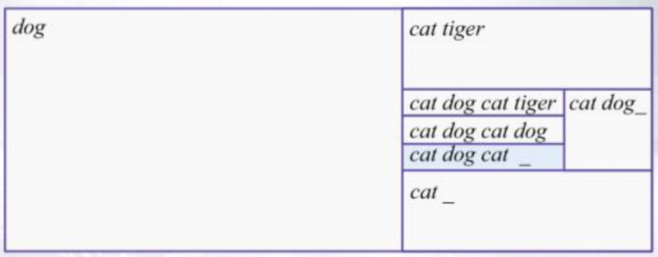
Let's check the model

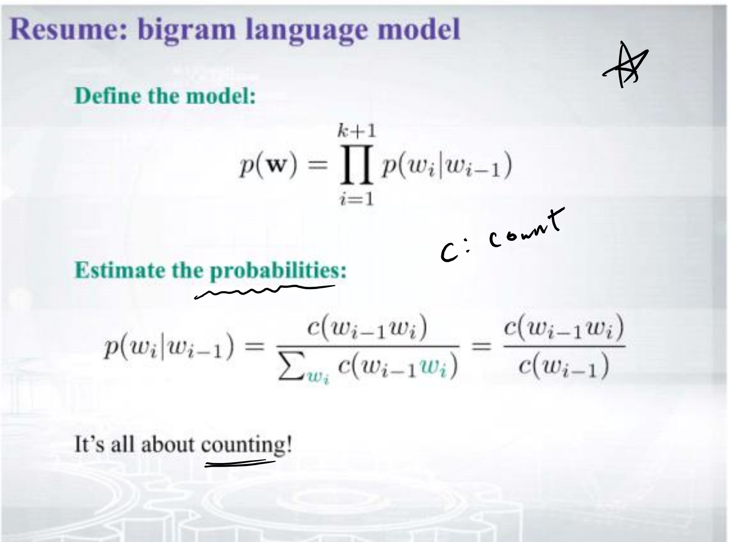
_ dog _

_ dog cat tiger _

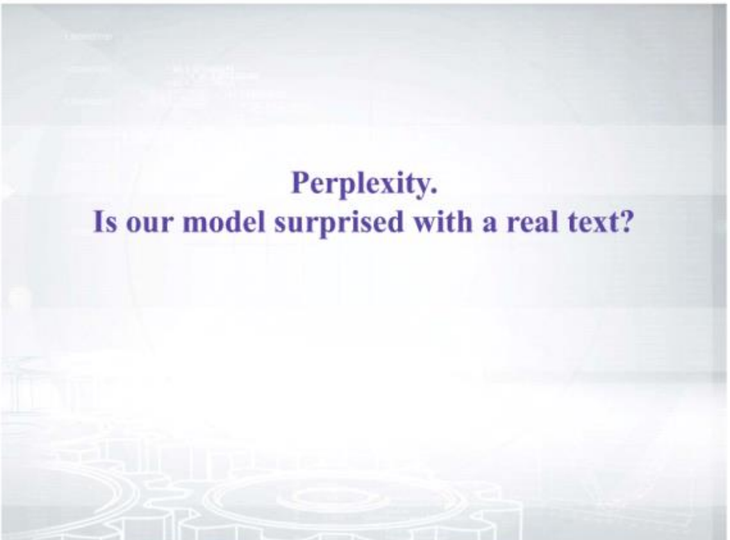
_ cat dog cat _

$$p(cat\ dog\ cat) = p(cat \mid \_)\ p(dog \mid cat)\ p(cat \mid dog)\ p(\_ \mid cat)$$




Resume: bigram language model
Define the model:
$$p(\mathbf{w}) = \prod_{i=1}^{k+1} p(w_i|w_{i-1})$$
c: count
Estimate the probabilities:
$$p(w_i|w_{i-1}) = \frac{c(w_{i-1}w_i)}{\sum_{w_i} c(w_{i-1}w_i)} = \frac{c(w_{i-1}w_i)}{c(w_{i-1})}$$
It's all about counting!

perplexity


Perplexity.
Is our model surprised with a real text?

How to train n-gram models

$$\mathbf{w} = (w_1 w_2 w_3 \dots w_k)$$

Bigram language model:

$$p(\mathbf{w}) = \prod_{i=1}^{k+1} p(w_i | w_{i-1})$$

How to train n-gram models

$$\mathbf{w} = (w_1 w_2 w_3 \dots w_k)$$

Bigram language model:

$$p(\mathbf{w}) = \prod_{i=1}^{k+1} p(w_i | w_{i-1})$$

N-gram language model:

extend the history

$(w_{i-n+1}, \dots w_{i-1})$

$$p(\mathbf{w}) = \prod_{i=1}^{k+1} p(w_i | w_{i-n+1}^{i-1})$$

How to train n-gram models

Log-likelihood maximization:

$$\log p(\mathbf{w}_{\text{train}}) = \sum_{i=1}^{N+1} \log p(w_i | w_{i-n+1}^{i-1}) \to \max$$

where are the parameters?

好像无 parameter 呀

Estimates for parameters:

$$p(w_i | w_{i-n+1}^{i-1}) = \frac{c(w_{i-n+1}^{i})}{\sum_{w_i} c(w_{i-n+1}^{i})} = \frac{c(w_{i-n+1}^{i})}{c(w_{i-n+1}^{i-1})}$$

N is the length of the **train corpus** (all words concatenated).

Generated Shakespeare

Unigrams:

To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have. Every enter now severally so, let. Hill he late speaks; or! a more to leg less first you enter.

Bigrams:

What means, sir. I confess she? then all sorts, he is trim, captain. Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow. What we, hath got so she that I rest and sent to scold and nature bankrupt, nor the first gentleman?

Jurafsky & Martin, https://lagunita.stanford.edu/c4x/Engineering/CS-224N/asset/slp4.pdf

Generated Shakespeare

3-grams:

Sweet prince, Falstaff shall die. Harry of Monmouth's grave. This shall forbid it should be branded, if renown made it empty. What is't that cried? Indeed the duke; and had a very good friend. Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done.

4-grams:

King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in; Will you not tell me who I am? It cannot be but so. Indeed the short and the long. Marry, 'tis a noble Lepidus. They say all lovers swear more performance than they are wont to keep obliged faith.

Jurafsky & Martin, https://lagunita.stanford.edu/c4x/Engineering/CS-224N/asset/slp4.pdf

Which model is better?

The best n might depend on how much data you have:

- bigrams might be not enough
- 7-grams might never occur

3 gram fine ...

Extrinsic evaluation:

- Quality of a downstream task: machine translation, speech recognition, spelling correction…

Intrinsic evaluation:

- Hold-out (text) perplexity!

we have data. hold out some data to complete perplexity

Evaluate the model on the test set

Likelihood:

$$\mathcal{L} = p(\mathbf{w}_{\text{test}}) = \prod_{i=1}^{N+1} p(w_i | w_{i-n+1}^{i-1})$$

Perplexity:

related to entropy.

$$\mathcal{P} = p(\mathbf{w}_{\text{test}})^{-\frac{1}{N}} = \frac{1}{\sqrt[N]{p(\mathbf{w}_{\text{test}})}}$$

→ lower perplexity ⇒ better.

N is the length of the **test corpus** (all words concatenated).

Lower perplexity is better!

Out-of-vocabulary words

Toy train corpus:
This is the house that Jack built.

Toy test corpus:
This is the *malt*.

Out-of-vocabulary words

Toy train corpus:
This is the house that Jack built.

Toy test corpus:
This is the *malt*.

What's the perplexity of the Bigram LM?

Out-of-vocabulary words

Toy train corpus:
This is the house that Jack built.

Toy test corpus:
This is the *malt*.

What's the perplexity of the Bigram LM?

$$p(malt|the) = \frac{c(the\,malt)}{c(the)} = 0$$

Out-of-vocabulary words

Toy train corpus:
This is the house that Jack built.

Toy test corpus:
This is the *malt*.

What's the perplexity of the Bigram LM?

$$p(malt|the) = \frac{c(the\, malt)}{c(the)} = 0$$

$$p(\mathbf{w}_{\text{test}}) = 0$$

Out-of-vocabulary words

Toy train corpus:
This is the house that Jack built.

Toy test corpus:
This is the *malt*.

What's the perplexity of the Bigram LM?

$$p(malt|the) = \frac{c(the\,malt)}{c(the)} = 0$$

$$p(\mathbf{w}_{\text{test}}) = 0$$

$$\mathcal{P} = \inf$$

Out-of-vocabulary words

Toy train corpus:
This is the house that Jack built.

Toy test corpus:
This is the *malt*. → unknown word. problem

What's the perplexity of the Bigram LM?

$$p(malt|the) = \frac{c(the\,malt)}{c(the)} = 0$$

$$p(\mathbf{w}_{\text{test}}) = 0$$

$$\mathcal{P} = \text{inf}$$

How can we fix that?

Simple idea:

- Build a vocabulary (e.g. by word frequencies)
- Substitute OOV words by <UNK> (both in train and test!)
- Compute counts as usual for all tokens
- Profit!

OK, no OOV words

Toy train corpus:
This is the house that Jack built.

Toy test corpus:
This is *Jack*.

What's the perplexity of the Bigram LM?

OK, no OOV words

Toy train corpus:
This is the house that Jack built.

Toy test corpus:
This is *Jack*.

What's the perplexity of the Bigram LM?

$$p(Jack \mid is) = \frac{c(is\ Jack)}{c(is)} = 0$$

OK, no OOV words

Toy train corpus:
This is the house that Jack built.

Toy test corpus:
This is *Jack*.

What's the perplexity of the Bigram LM?

$$p(Jack \mid is) = \frac{c(is\ Jack)}{c(is)} = 0$$

$$p(\mathbf{w}_{\text{test}}) = 0$$

OK, no OOV words

Toy train corpus:
This is the house that Jack built.

Toy test corpus:
This is *Jack*.

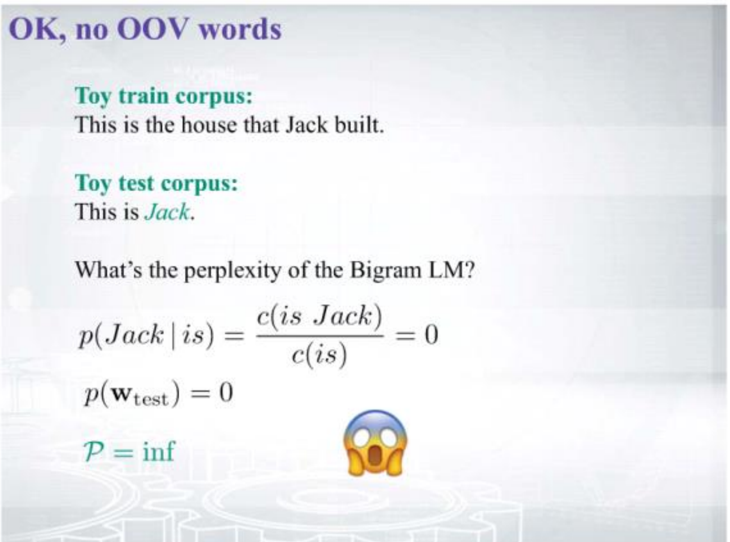
What's the perplexity of the Bigram LM?

$$p(Jack \mid is) = \frac{c(is\ Jack)}{c(is)} = 0$$

$$p(\mathbf{w}_{\text{test}}) = 0$$

$$\mathcal{P} = \inf$$

need smoothing techniques.


OK, no OOV words
Toy train corpus:
This is the house that Jack built.
Toy test corpus:
This is *Jack*.
What's the perplexity of the Bigram LM?
$p(Jack \mid is) = \frac{c(is\ Jack)}{c(is)} = 0$
$p(\mathbf{w}_{\text{test}}) = 0$
$\mathcal{P} = \inf$

smoothing

Smoothing.
What if we see new n-grams?

Zero probabilities for test data

Toy train corpus:
This is the house that Jack built.

Toy test corpus:
This is *Jack.*

What's the perplexity of the Bigram LM?

$$p(Jack \mid is) = \frac{c(is\ Jack)}{c(is)} = 0$$

$$p(\mathbf{w}_{\text{test}}) = 0$$

$$\mathcal{P} = \inf$$

Laplacian smoothing

Idea:

- Pull some probability from frequent bigrams to infrequent ones
- Just add 1 to the counts (add-one smoothing):

$$\hat{p}(w_i|w_{i-n+1}^{i-1}) = \frac{c(w_{i-n+1}^{i}) + 1}{c(w_{i-n+1}^{i-1}) + V}$$

- Or tune a parameter (add-k smoothing):

$$\hat{p}(w_i|w_{i-n+1}^{i-1}) = \frac{c(w_{i-n+1}^{i}) + k}{c(w_{i-n+1}^{i-1}) + Vk}$$

k. can tune using test data.

Laplacian Smoothing

Katz backoff

Problem:

- Longer n-grams are better, but data is not always enough

Idea:

- Try a longer n-gram and back off to shorter if needed

$$\hat{p}(w_i|w_{i-n+1}^{i-1}) = \begin{cases} \tilde{p}(w_i|w_{i-n+1}^{i-1}), & \text{if } c(w_{i-n+1}^{i}) > 0 \\ \alpha(w_{i-n+1}^{i-1})\,\hat{p}(w_i|w_{i-n+2}^{i-1}), & \text{otherwise} \end{cases}$$

Katz Backoff.

Katz backoff

Problem:

- Longer n-grams are better, but data is not always enough

Idea:

- Try a longer n-gram and back off to shorter if needed

$$\hat{p}(w_i|w_{i-n+1}^{i-1}) = \begin{cases} \tilde{p}(w_i|w_{i-n+1}^{i-1}), & \text{if } c(w_{i-n+1}^{i}) > 0 \\ \alpha(w_{i-n+1}^{i-1})\,\hat{p}(w_i|w_{i-n+2}^{i-1}), & \text{otherwise} \end{cases}$$

where $\tilde{p}$ and α are chosen to ensure normalization.

We have some α weights in the else brunch. What's their purpose? Do you think they should be less or greater than 1?

- (●) They should be less than 1

Correct

- ( ) They should be greater than 1

the prob should some into ONE

Interpolation smoothing

Idea:

- Let us have a mixture of several n-gram models
- Example for a trigram model:

$$\hat{p}(w_i|w_{i-2}w_{i-1}) = \lambda_1 p(w_i|w_{i-2}w_{i-1}) + \lambda_2 p(w_i|w_{i-1}) + \lambda_3 p(w_i)$$

$$\lambda_1 + \lambda_2 + \lambda_3 = 1$$

- The weights are optimized on a test (dev) set
- Optionally they can also depend on the context

Absolute discounting

Idea:

- Let's compare the counts for bigrams in train and test sets

Experiment (Church and Gale, 1991):

- Subtract 0.75 and get a good estimate for the test count!

Train bigram count	Test bigram count
2	1.25
3	2.24
4	3.23
5	4.21
6	5.23
7	6.21
8	7.21

https://web.stanford.edu/~jurafsky/slp3/4.pdf

Average # of bigram

.75 ? some magical property ...

Absolute discounting

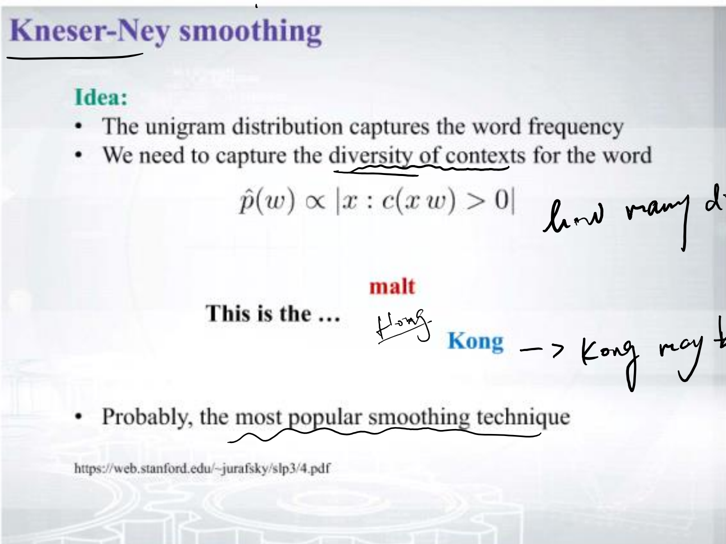
Idea:

- Let's compare the counts for bigrams in train and test sets

Experiment (Church and Gale, 1991):

- Subtract 0.75 and get a good estimate for the test count!

$$\hat{p}(w_i|w_{i-1}) = \frac{c(w_{i-1}w_i) - d}{\sum_x c(w_{i-1}x)} + \lambda(w_{i-1})p(w_i)$$

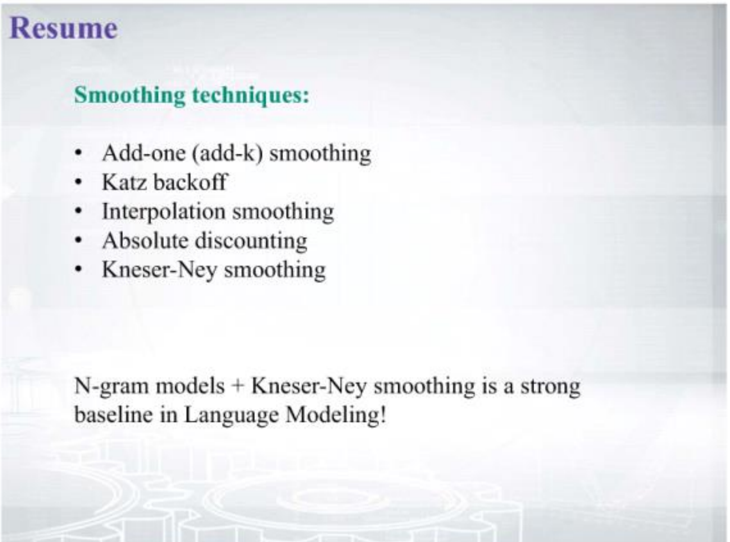


?

Hong

how many different context can go before the word.

-> Kong may ~~be more~~ have higher P. But it only happens after "Hong".
So. choose malt over Kong.



This is the house that Jack built.

This is the malt that lay in the house that Jack built.

This is the rat that ate the malt that lay in the house that Jack built.

From <https://www.coursera.org/learn/language-processing/exam/Lratd/language-modeling>

$P(\text{lay} \mid \text{that})$

$= \dfrac{c(\text{"that lay"})}{c(\text{"that"})} = \dfrac{2}{6} = \dfrac{1}{3}$

Consider the `bigram language model` trained on the sentence:

This is the cow with the crumpled horn that tossed the dog that worried the cat that killed the rat that ate the malt that lay in the house that Jack built.

Find the `probability of the sentence:`

<s> <s> *This is the rat that worried the dog that Jack built.* <e>.

From <https://www.coursera.org/learn/language-processing/exam/Lratd/language-modeling>

1×1

$P(\text{This} \mid \text{<s> <s>})$

$\dfrac{c(\text{3-gram})}{c(\text{context})}$ $V=$

$\dfrac{1}{\sqrt[n]{P(W_{test})}}$

$P(\text{this} \mid \text{<s> <s>}) \times P(\text{is} \mid \text{<s> this}) \times P(\text{the} \mid \text{this is}) \times P(\text{house} \mid \text{is the}) \times P(\text{that} \mid \text{the house})$

$\times P(\text{Jack} \mid \text{house that}) \times P(\text{built} \mid \text{that Jack})$

$=$

This is the rat that ate the malt that lay in the house that Jack built.

2. Consider the **bigram language model** trained on the sentence:

This is the cow with the crumpled horn that tossed the dog that worried the cat that killed the rat that ate the malt that lay in the house that Jack built.

Find the **probability of the sentence:**

This is the rat that worried the dog that Jack built.

$P(this|<s>) \times P(is|this)$

<s> This → the rat that worried the

dog that Jack built </s>.

- $\frac{1}{2}\cdot\frac{1}{3}\cdot\frac{1}{6}\cdot\frac{1}{2}\cdot\frac{1}{7}\cdot\frac{1}{2}\cdot\frac{1}{6}\cdot\frac{1}{2}\cdot\frac{1}{7}\cdot\frac{1}{3}\cdot\frac{1}{5}\cdot\frac{1}{4}$

  This should not be selected
  Actually not, these were just random fractions :)

  Let's start with the first probability: p(This | <start>) = 1 since there is only one <start> in the training corpus and it's followed by "This".

- ∞
- $\frac{1}{6}\cdot\frac{1}{7}\cdot\frac{1}{6}\cdot\frac{1}{7}$
- 1/8
- 0

Exactly! Most of the conditional probabilities are equal to 1, e.g. p(is|This) = 1 since "This" occurs only once in the training data and it's followed by "is". Only the probabilities for "the" and "that" are non-trivial.

From <https://www.coursera.org/learn/language-processing/exam/Lratd/language-modeling>

this | <s>

4. Apply **add-one smoothing** to the trigram language model trained on the sentence:

This is the rat that ate the malt that lay in the house that Jack built.

Find the **perplexity** of this smoothed model on the test sentence:

This is the house that Jack built.

Write the answer with precision of 3 digits after the decimal point.

Enter answer here

Incorrect Response
The answer you gave is not a number.

<s>
this
is
the
rate
that

---

5. Find one incorrect statement below:

- Trigram language models can have a larger perplexity than bigram language models. ?
- End-of-sentence tokens are necessary for modelling probabilities of sentences of different lengths. ok ✓

  This should not be selected
  This fact was discussed in the lecture a lot.

- The smaller holdout perplexity is - the better the model. ✓
- N-gram language models cannot capture distant contexts. ✓
- If a test corpus does not have out-of-vocabulary words, smoothing is not needed.

	Count
built	1
Jack	1
house	1
this	1
is	1
the	3
rat	1
that	3
ate	1
malt	1
lay	1
in	1
</s>	1

4. Apply **add-one smoothing** to the trigram language model trained on the sentence:

This is the rat that ate the malt that lay in the house that Jack built.

Find the **perplexity** of this smoothed model on the test sentence:

This is the house that Jack built.

Write the answer with precision of 3 digits after the decimal point.

Enter answer here

$\frac{1+1}{2+16}$

Some hints for you: there are 12 unique words in train, so V=13 (because of the fake end token).
The length of the test sentence is N=7 words. Use these values for the vocabulary size in add-one

Enter answer here

$\frac{1+1}{1+16}$

$\frac{2}{7}$

in — 1

</s> — 1

<s> — 1

Some hints for you: there are 12 unique words in train, so V=13 (because of the fake end token). The length of the test sentence is N=7 words. Use these values for the vocabulary size in add-one smoothing and for the root index in the perplexity respectively. And do not forget about start and end tokens!

You might need a piece of paper to calculate this. Get back to our reading material in this module to review a similar task.

From <https://www.coursera.org/learn/language-processing/exam/Lratd/language-modeling>