

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:

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

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

How to train n-gram models

Log-likelihood maximization:

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

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?*

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.*

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

Extrinsic evaluation:

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

Intrinsic evaluation:

- Hold-out (text) 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:

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

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

Lower perplexity is better!

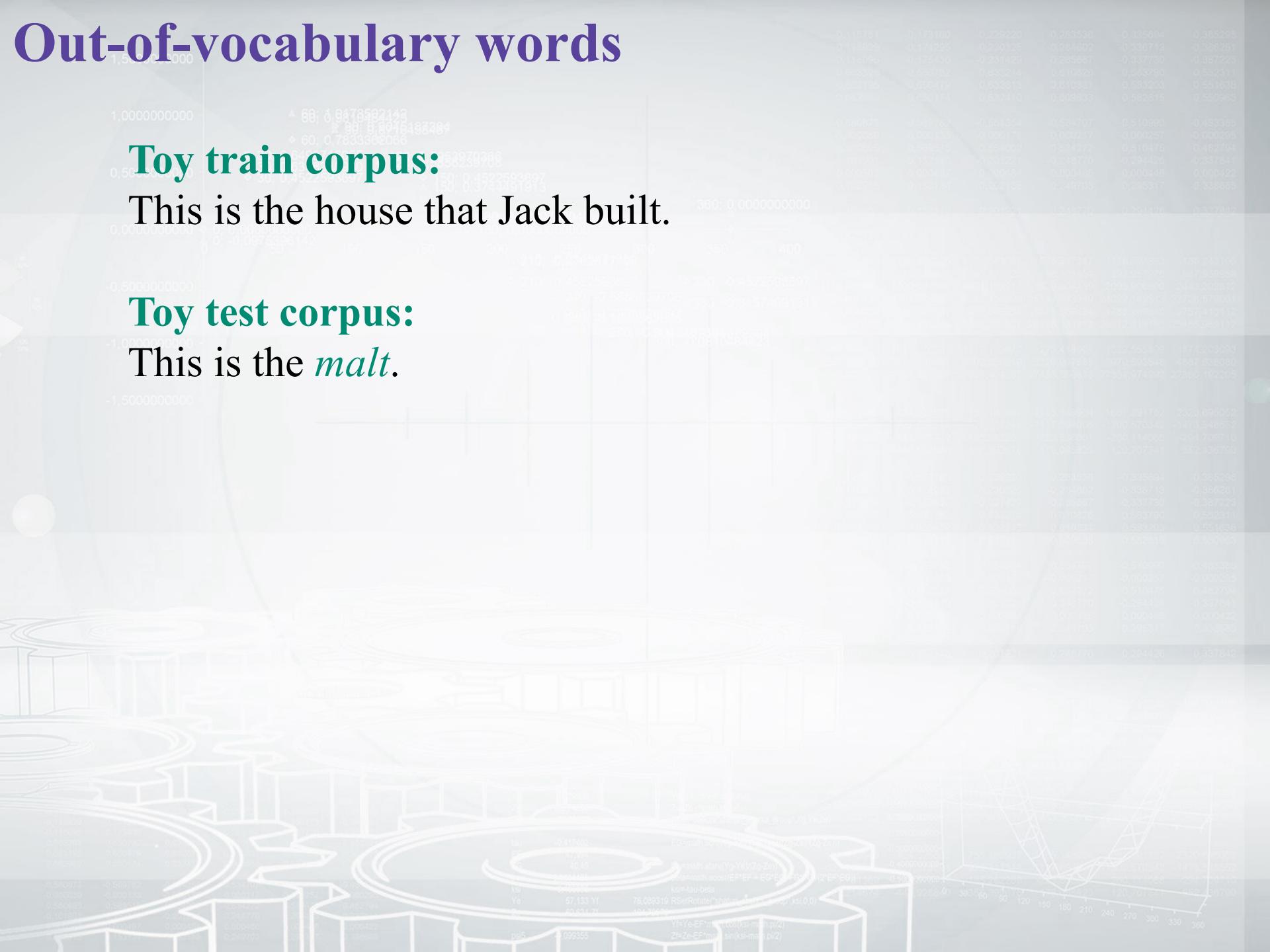
Out-of-vocabulary words

Toy train corpus:

This is the house that Jack built.

Toy test corpus:

This is the *malt*.



Out-of-vocabulary words

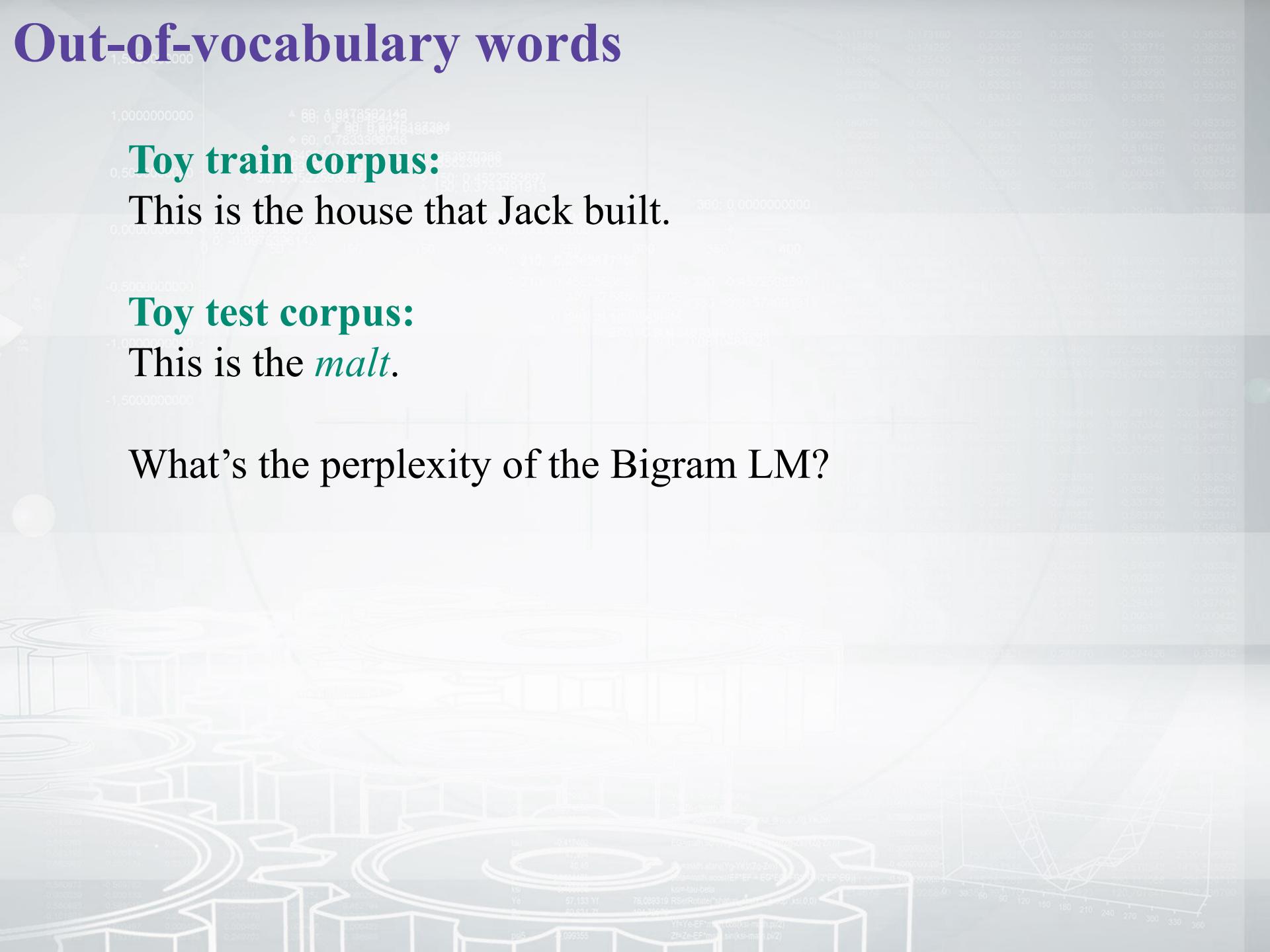
Toy train corpus:

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:

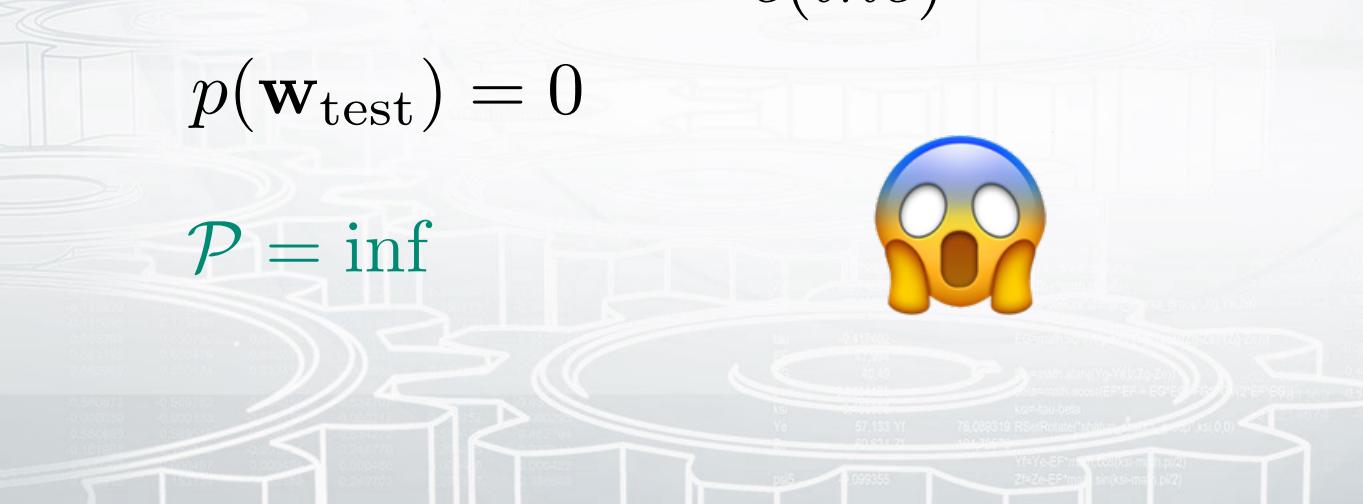
This is the *malt*.

What's the perplexity of the Bigram LM?

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

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

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



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?

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

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?

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

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

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?

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

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

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

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?

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

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

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

