

CSCI 1470/2470  
Spring 2024

Ritambhara Singh

February 29, 2024  
Wednesday

Feed Forward LMs

# Deep Learning

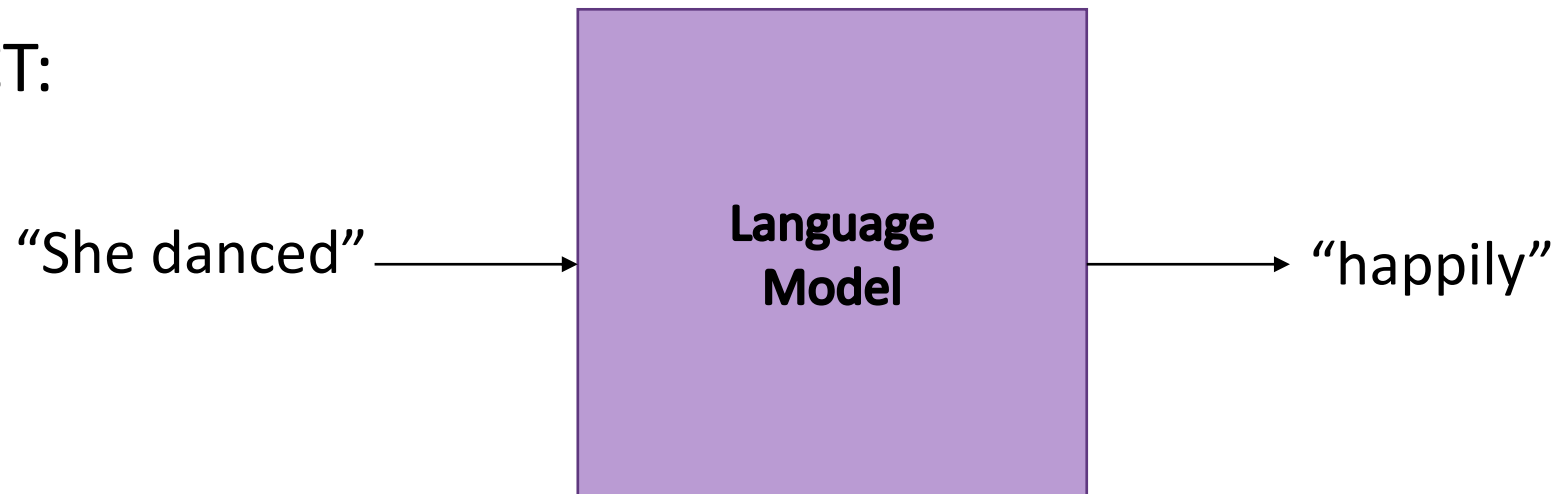
No new lab this week!  
Will hold lab hours for catch-up  
and late check-offs

# Review: Language Modeling

Goal: Predict future words in a sentence given previous words:

TRAIN: "She danced happily. They sang beautifully."

PREDICT:



# Review: N-gram counting

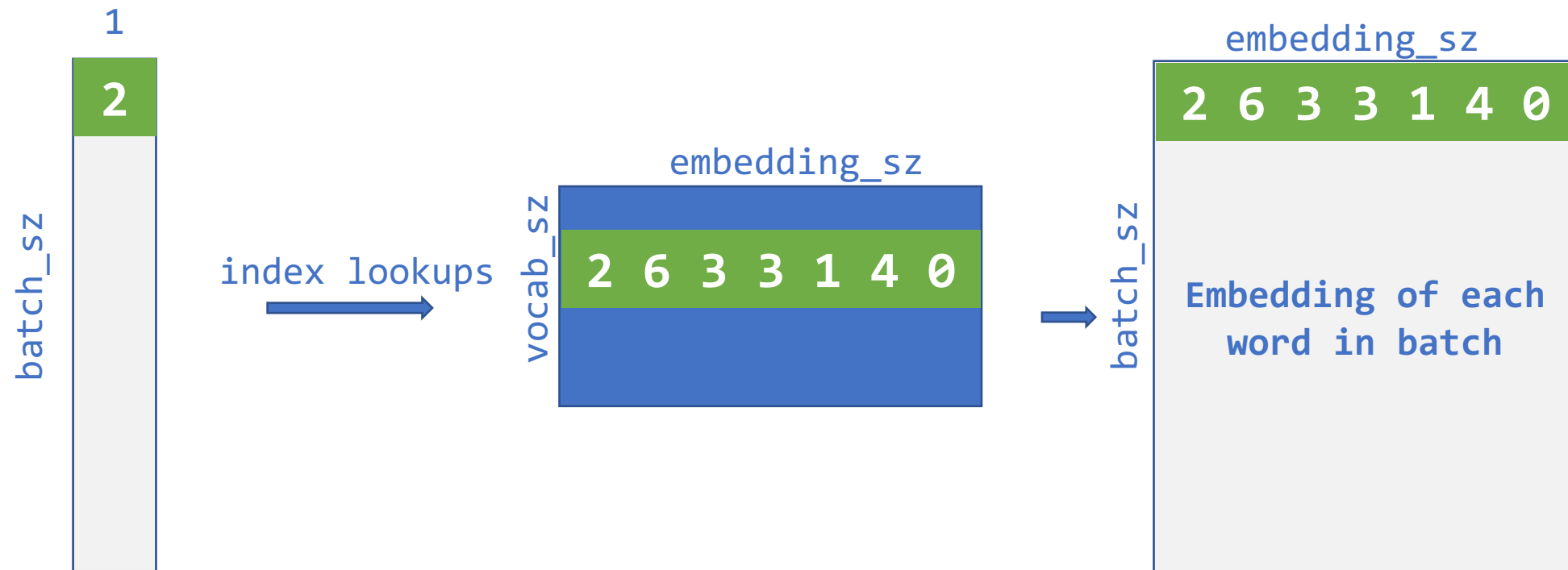
Improvement: **N-gram** model – only look at **N** words at a time  
(in this case, **bigrams** look at **2** words at a time)

- “*danced happily*”
- “*sang beautifully*”
- “*danced energetically*”
- “*sang happily*”
- “*danced gracefully*”

“*He danced happily*” now has 1/3 probability!

# Review: Embeddings

Represent words as embedding vectors using:  
**tf.nn.embedding\_lookup**





# Today's goal – Building a *Deep* Language Model

- (1) Learning a deep bigram model
- (2) Improve upon the deep bigram model
- (3) Evaluating language models

Once upon a time...



*“The dog barked loudly.”*



*“The cat meowed softly”*

# Data Preprocessing

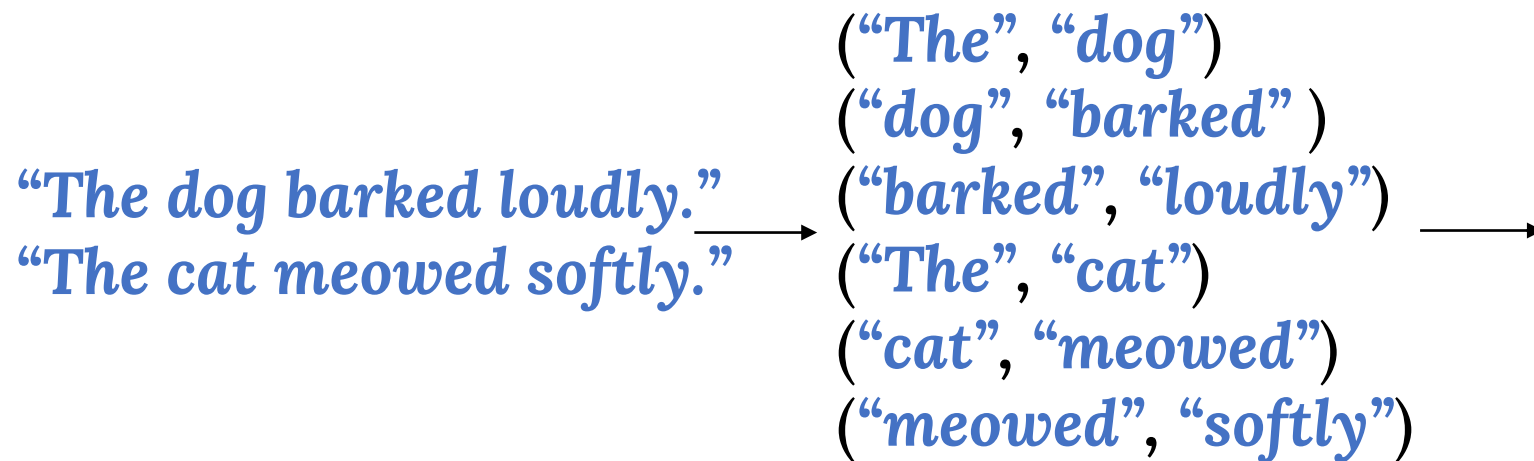
First, we extract all of the bigrams from the training corpus.

*“The dog barked loudly.”*  
*“The cat meowed softly.”* → *(“The”, “dog”)*  
*(“dog”, “barked” )*  
*(“barked”, “loudly”)*  
*(“The”, “cat”)*  
*(“cat”, “meowed”)*  
*(“meowed”, “softly”)*

# Data Preprocessing

First, we extract all of the bigrams from the training corpus.

Create training batches by pairing up first and second words:



inputs	labels
<i>“The”</i>	<i>“dog”</i>
<i>“dog”</i>	<i>“barked”</i>
<i>“barked”</i>	<i>“loudly”</i>
<i>“The”</i>	<i>“cat”</i>
<i>“cat”</i>	<i>“meowed”</i>
<i>“meowed”</i>	<i>“softly”</i>



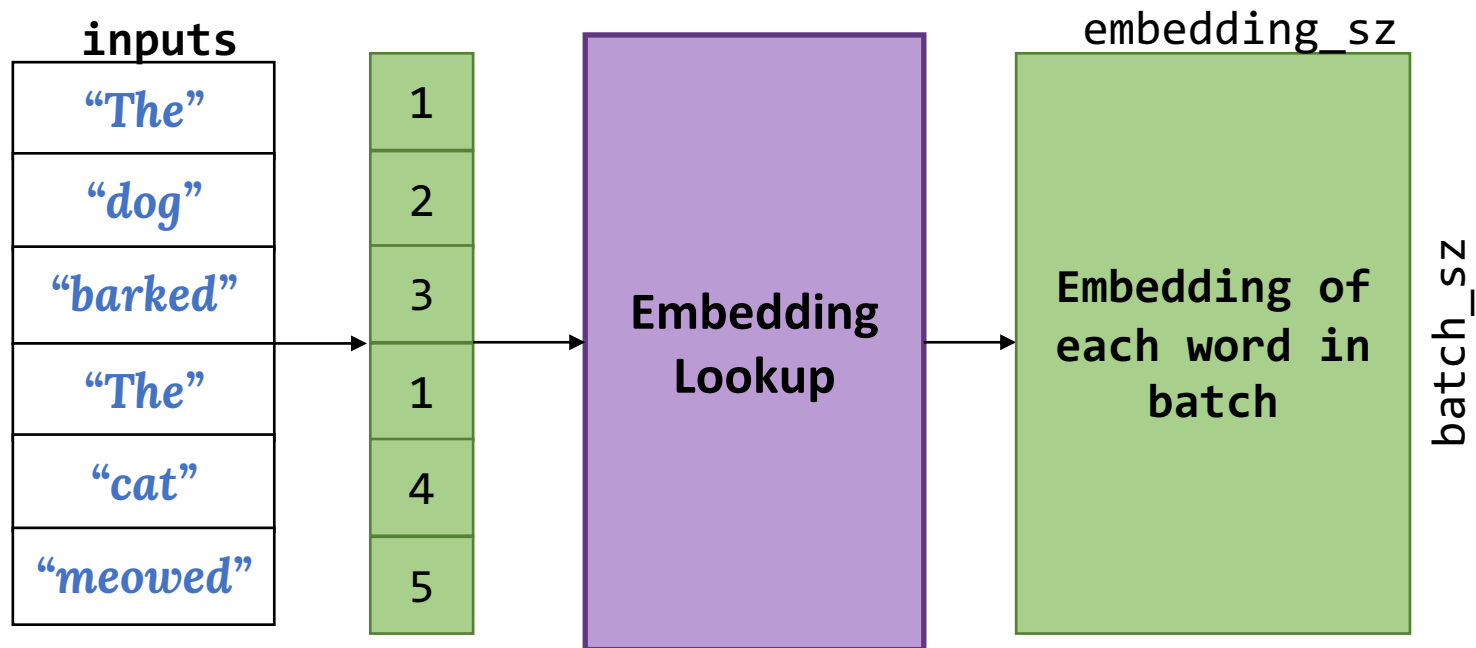
# Bigram Language Model Architecture

We convert all words to their corresponding vocab indices.



# Bigram Language Model Architecture

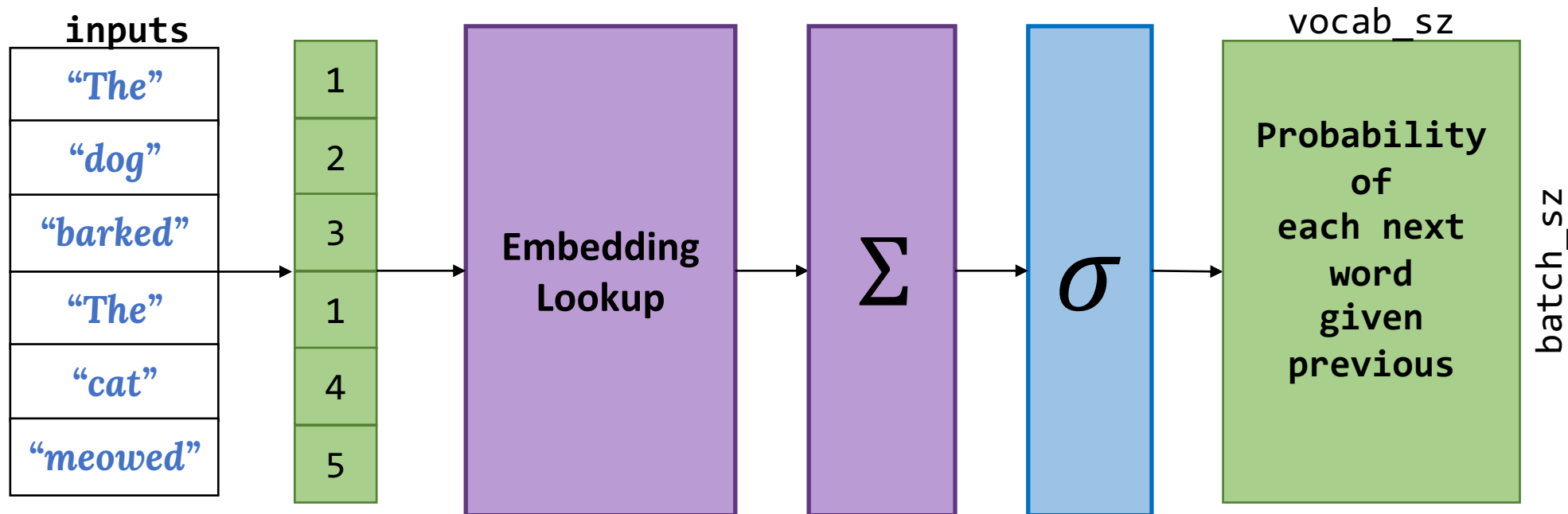
Applying `tf.nn.embedding_lookup` to our entire batch gets the embedding for each word in the batch.



What do we do next?

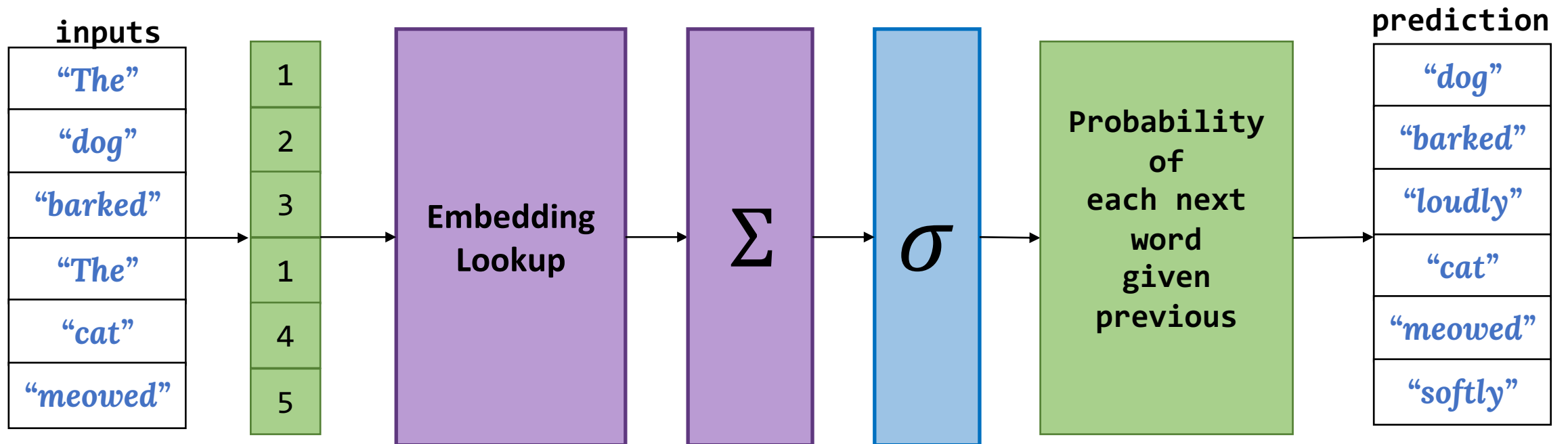
# Bigram Language Model Architecture

We feed our batch of embeddings to a fully connected layer with softmax activation to get probability of each word in vocab.



# Bigram Language Model Architecture

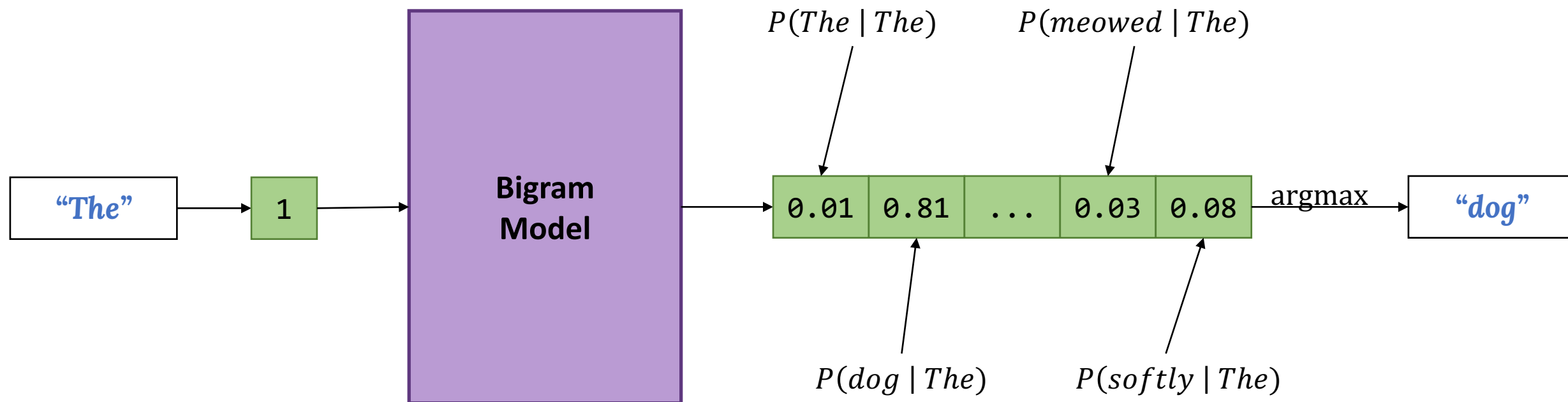
Finally, we choose the word with the max probability as our prediction.



Should we be concerned about a large vocabulary making the probabilities small?

# Bigram Language Model Output

The output of our model gives us the probability of each word in our vocabulary appearing next, given the previous word.





Why might the bigram model not be sufficient?

# Improving on the Bigram model

Why might the bigram model not be sufficient?

Consider slightly more *distant* sentence relationships:

“The dog was barking.”

“The cat was meowing.”



# Improving on the Bigram model

Why might the bigram model not be sufficient?

Consider slightly more *distant* sentence relationships:

“The dog was barking.”  


“The cat was meowing.”  


We want to capture *context* farther than the immediately preceding word.

Using the bigram model, we would need to predict “*barking*” and “*meowing*” based only on the word “*was*”.

Can we do better?

# Trigram Language Model

*“The dog was barking.”*  
*“The cat was meowing.”*

For the trigram model, we treat the first two words of each trigram as the input, and the third word as the target.

*(“The”, “dog”, “was”)*  
*(“dog”, “was”, “barking”)*  
*(“The”, “cat”, “was”)*  
*(“cat”, “was”, “meowing”)*

inputs		labels
<i>“The”</i>	<i>“dog”</i>	<i>“was”</i>
<i>“dog”</i>	<i>“was”</i>	<i>“barking”</i>
<i>“The”</i>	<i>“cat”</i>	<i>“was”</i>
<i>“cat”</i>	<i>“was”</i>	<i>“meowing”</i>



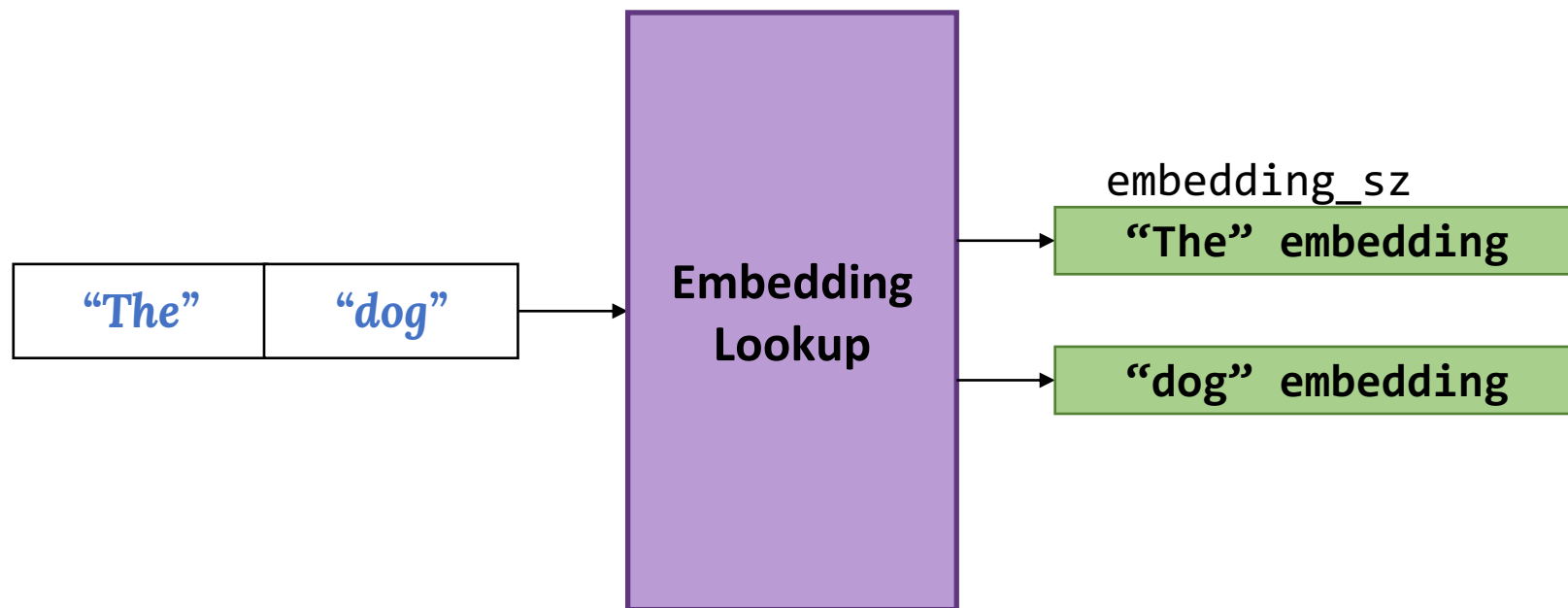
# Trigram Language Model Input

Now our network input is two words...

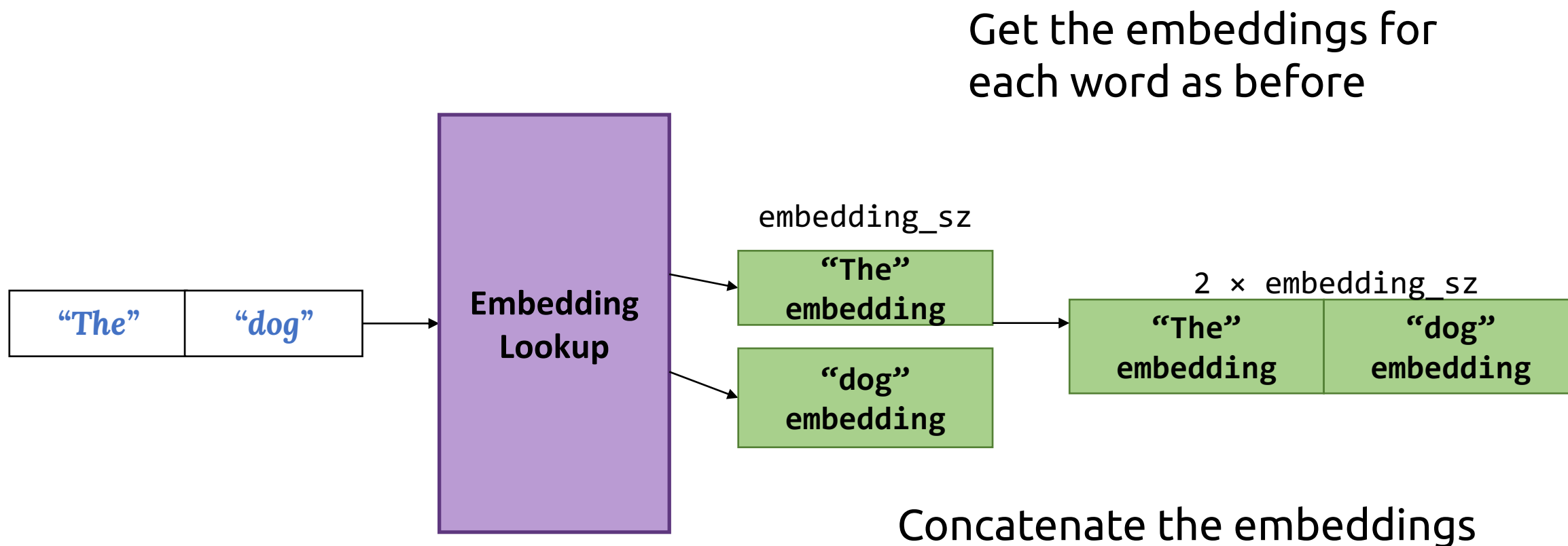
...how do we turn these into a tensor to feed into our network?

# Handling Multi-Word Input

Get the embeddings for each word as before



# Handling Multi-Word Input

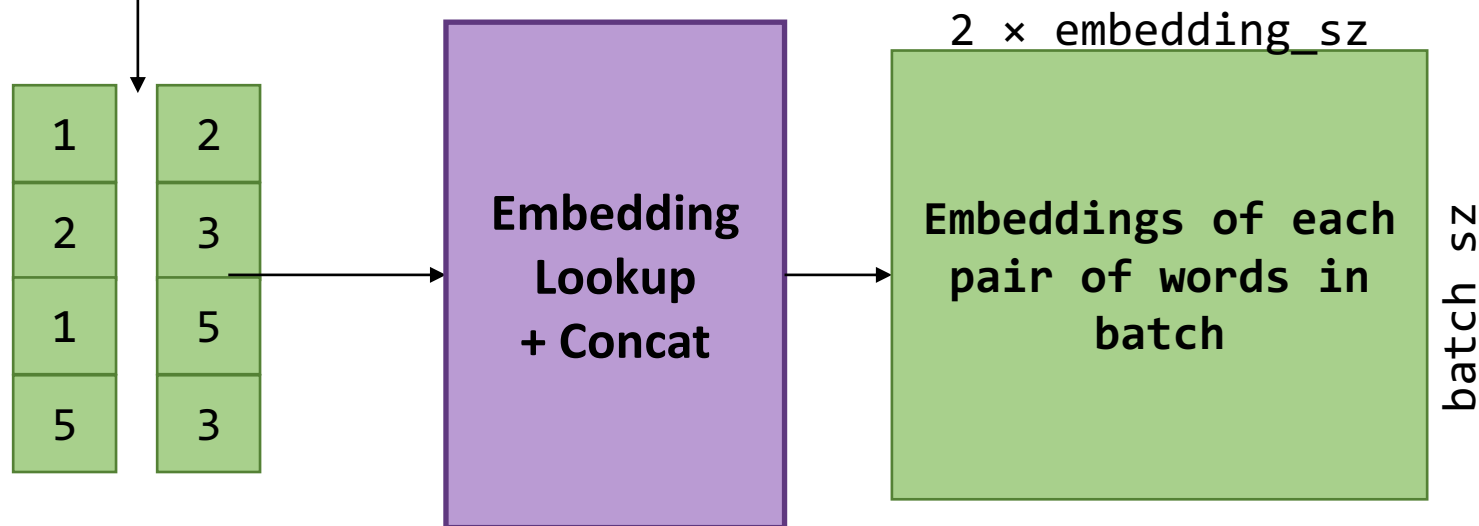


# Complete Trigram Language Model

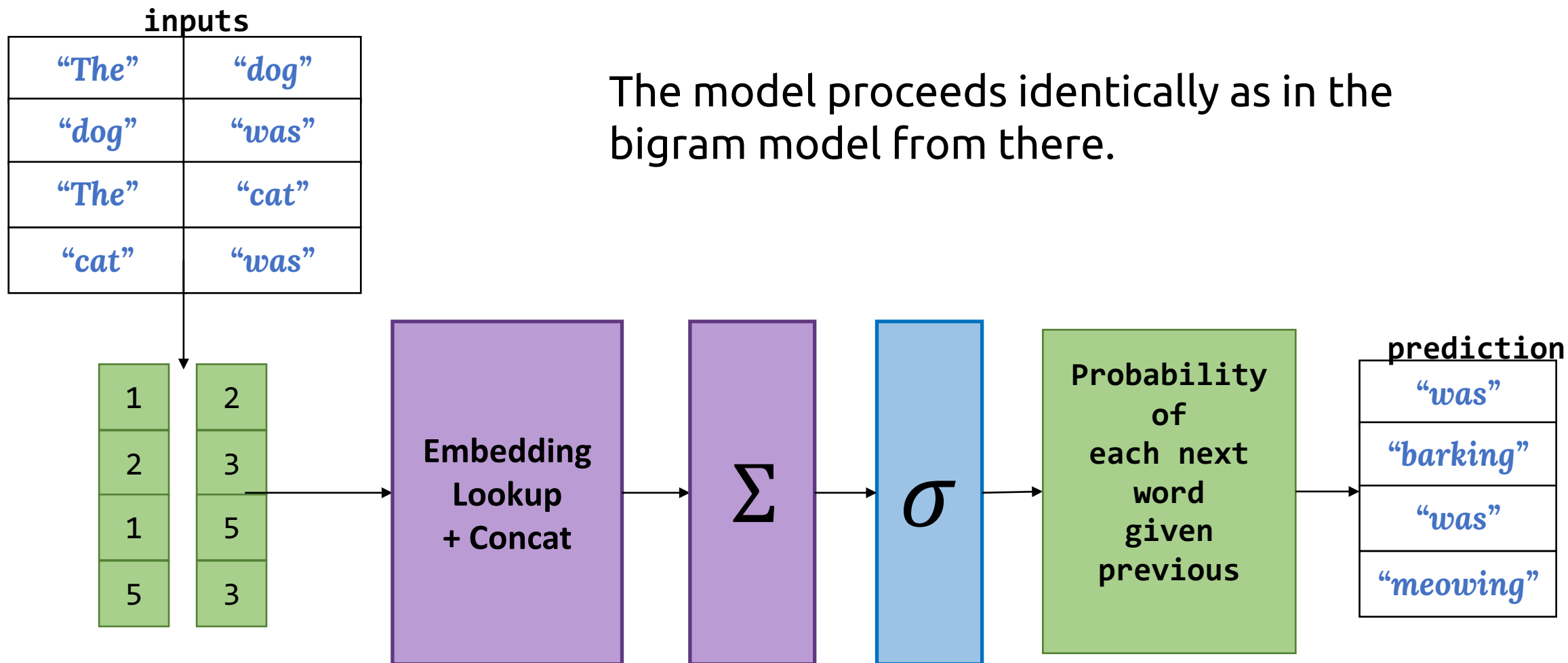
**inputs**

<i>"The"</i>	<i>"dog"</i>
<i>"dog"</i>	<i>"was"</i>
<i>"The"</i>	<i>"cat"</i>
<i>"cat"</i>	<i>"was"</i>

First we get the concatenated embeddings of the word pairs.



# Complete Trigram Language Model

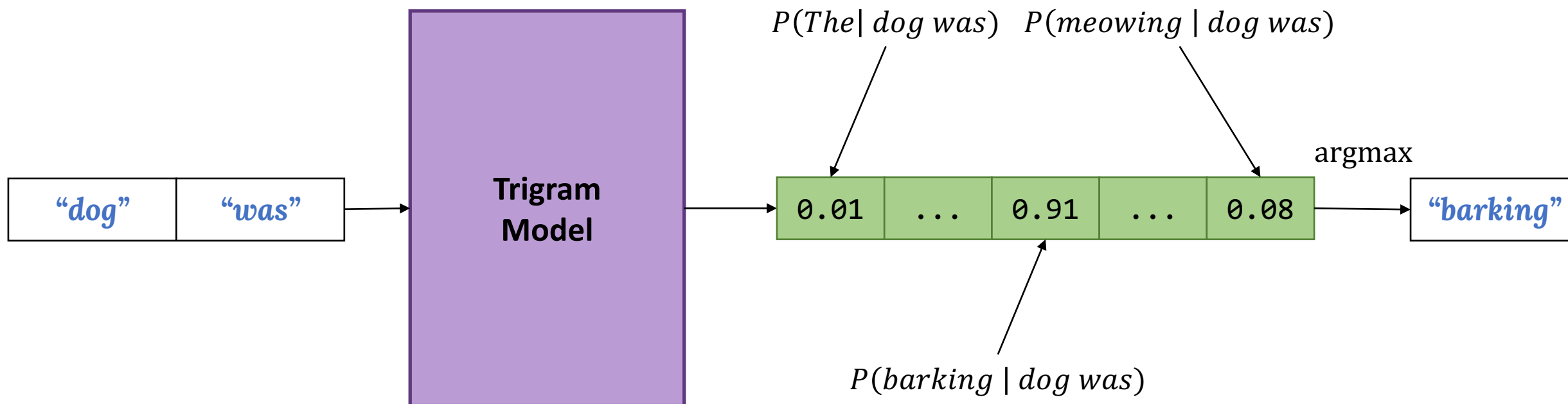






# Trigram Language Model Output

In the trigram version, the probabilities are now conditioned on two previous words rather than just one.



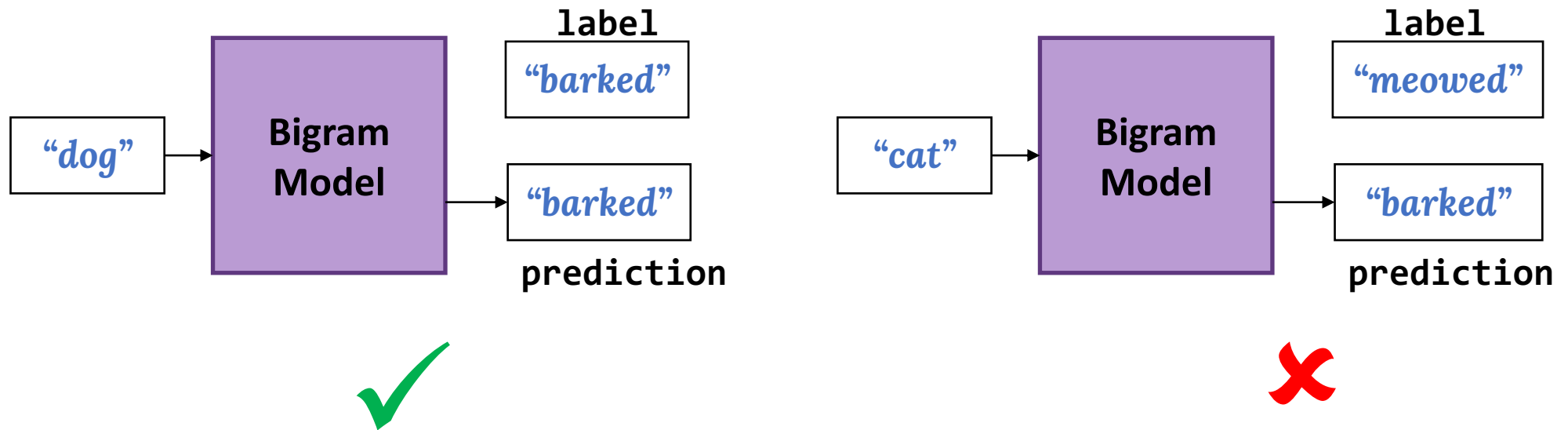
# Language Model Assessment

How do we know when our model is performing well?

# Language Model Assessment

How do we know when our model is performing well?

For starters, we can print some predictions and judge for ourselves:



# Language Model Assessment

How do we know when our model is performing well?

Or, we can examine similarities between embedding vectors

Enter a word and see words with similar vectors.

dog	<input type="button" value="List words"/>
dog	1
dogs	0.8680486950130833
puppy	0.8106427882830397
cat	0.7609456296774421
pet	0.7164786254811731
kitten	0.665988048830015
cats	0.6653174955688891
puppies	0.6637063702726447
pets	0.6538857831173411
doggie	0.6515337842020129

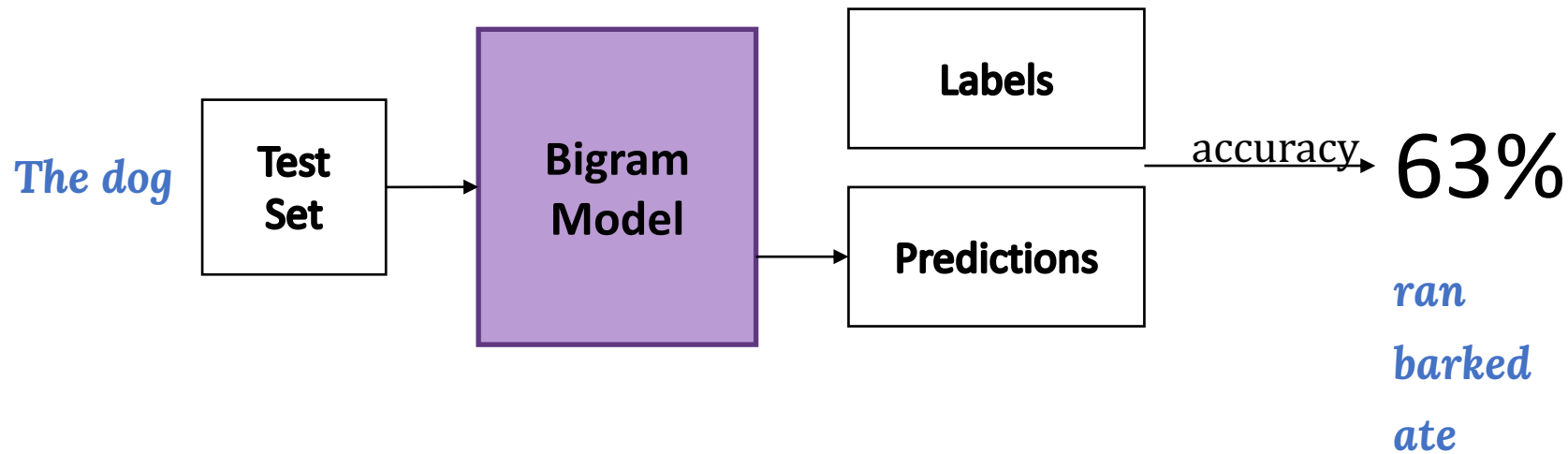
These are forms of *qualitative* evaluation

What about *quantitative* evaluation?

# Language Model Assessment: Quantitative

How do we know when our model is performing well?

We can evaluate the per-word accuracy on a test set:

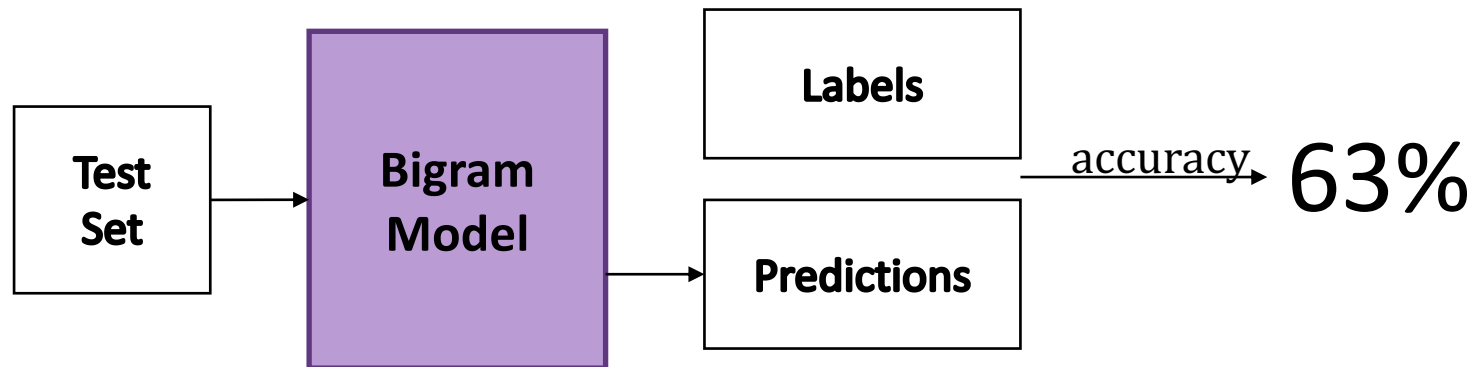


Is there an issue with this metric?

# Language Model Assessment: Quantitative

How do we know when our model is performing well?

We can evaluate the per-word accuracy on a test set:



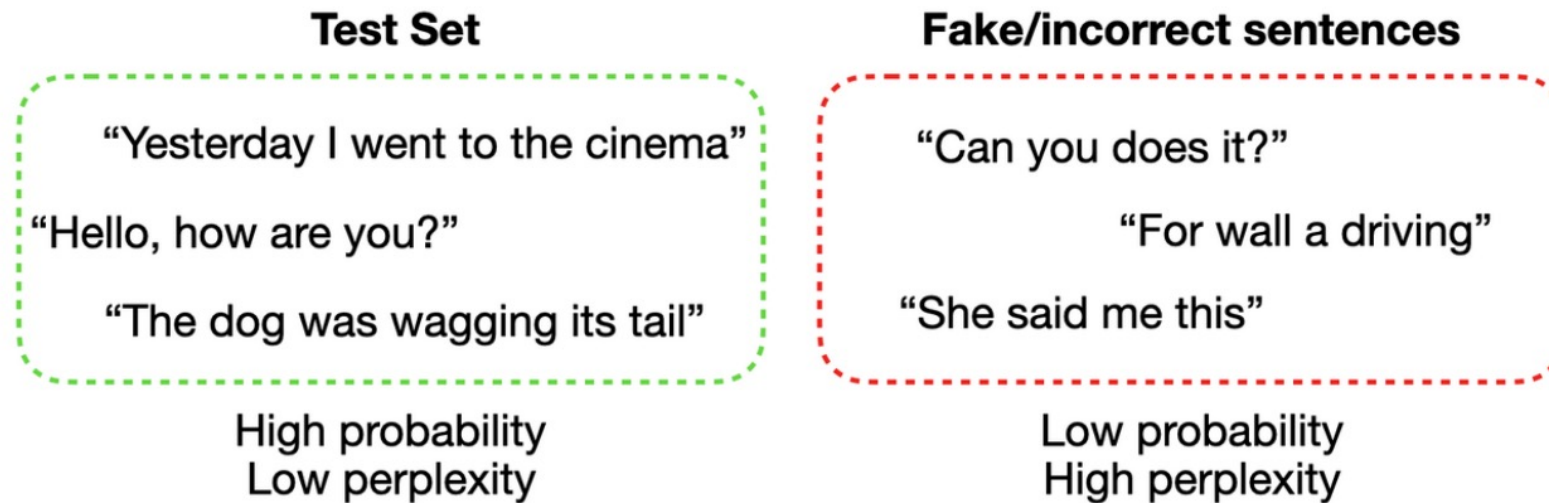
Is there an issue with this metric?

When you've got thousands of possible words, this is not a great measure (i.e. the top 1 prediction is going to differ from the ground truth label ***a lot*** of the time)

# Perplexity

What is a good language model?

Assigns high probabilities to sentences that are real and syntactically correct, and low probabilities to fake, incorrect, or highly infrequent sentences



Intuitively – A model assigning high probability to a sentence means it not “not perplexed” by this new sentence (low perplexity)!

# Perplexity

The standard quantitative metric in NLP for assessing language models

$$\text{Perplexity}(D) = e^{\frac{\sum_{s \in D} \sum_{w_i \in s} \frac{-\log p(w_i^s | w_1^s \dots w_{i-1}^s)}{|s|}}{|D|}}$$

where

- $D$  = an unseen test dataset of sentences
- $s$  = a sentence in the test set
- $w_i^s$  = the  $i^{\text{th}}$  word of sentence  $s$
- $p(\cdot)$  = the probability of the next word under our learned model



# Making Perplexity less perplexing...

$$\text{Perplexity}(D) = e^{\frac{\sum_{s \in D} \sum_{w_i \in s} \frac{-\log p(w_i^s | w_1^s \dots w_{i-1}^s)}{|s|}}{|D|}}$$

# Making Perplexity less perplexing...

$$\text{Perplexity}(D) = e^{\frac{\sum_{s \in D} \sum_{w_i \in s} \frac{\text{cross entropy loss for } w_i^s}{|s|}}{|D|}}$$

Why normalize  
by the sentence  
length?

# Making Perplexity less perplexing...

$$\text{Perplexity}(D) = e^{\frac{\sum_{s \in D} \text{avg cross entropy loss for } s}{|D|}}$$

Any questions?



# Making Perplexity less perplexing...

Perplexity( $D$ ) =  $e^{\text{avg cross entropy loss for all words in } D}$

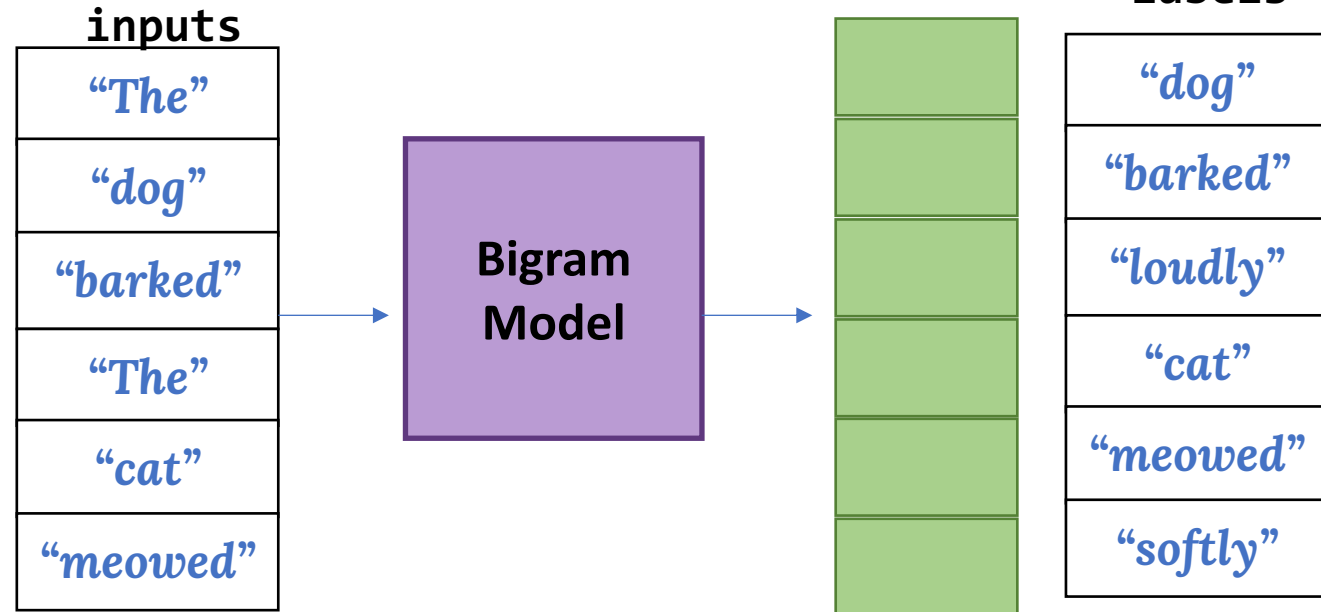
# Let's tie it together

$$\text{Perplexity}(D) = e^{\text{avg cross entropy loss for all words in } D}$$

$V = \{\text{"the", "dog", "cat", "barked", "meowed", "was", "barking", "meowing", "loudly", "softly"}\}$



*"The dog barked loudly."*



*"The cat meowed softly"*

- (1) What is the perplexity for a randomly initialized language model?  
[What output probabilities would you assign if you had no idea about the data]

# Perplexity: Intuitive Meaning

- “If a model has a perplexity of  $X$ , then it has the same odds of predicting the correct next word as a fair die with  $X$  sides”
- For a randomly-initialized model:
  - All words in the vocab  $V$  have equal probability  $\frac{1}{|V|}$

# Perplexity: Intuitive Meaning

- “If a model has a perplexity of  $X$ , then it has the same odds of predicting the correct next word as a fair die with  $X$  sides”

- For a randomly-initialized model:

- All words in the vocab  $V$  have equal probability  $\frac{1}{|V|}$

- Perplexity( $D$ ) =  $e^{\frac{\sum_{s \in D} \sum_{w_i \in S} \frac{-\log p(w_i^s | w_1^s \dots w_{i-1}^s)}{|S|}}{|D|}}$

# Perplexity: Intuitive Meaning

- “If a model has a perplexity of  $X$ , then it has the same odds of predicting the correct next word as a fair die with  $X$  sides”

- For a randomly-initialized model:

- All words in the vocab  $V$  have equal probability  $\frac{1}{|V|}$

- Perplexity( $D$ ) =  $e^{\frac{\sum_{s \in D} \sum_{w_i \in S} \frac{-\log p(w_i^s | w_1^s \dots w_{i-1}^s)}{|s|}}{|D|}}$  =  $e^{\frac{\sum_{s \in D} \sum_{w_i \in S} \frac{-\log(\frac{1}{|V|})}{|s|}}{|D|}}$



# Perplexity: Intuitive Meaning

- “If a model has a perplexity of  $X$ , then it has the same odds of predicting the correct next word as a fair die with  $X$  sides”

- For a randomly-initialized model:

- All words in the vocab  $V$  have equal probability  $\frac{1}{|V|}$

- Perplexity( $D$ ) =  $e^{\frac{\sum_{s \in D} \sum_{w_i \in S} \frac{-\log p(w_i^s | w_1^s \dots w_{i-1}^s)}{|S|}}{|D|}}$  =  $e^{\frac{\sum_{s \in D} \sum_{w_i \in S} \frac{-\log(\frac{1}{|V|})}{|S|}}{|D|}}$  =  $e^{-\log(\frac{1}{|V|})} = |V|$

# Perplexity: Intuitive Meaning

- “If a model has a perplexity of  $X$ , then it has the same odds of predicting the correct next word as a fair die with  $X$  sides”

- For a randomly-initialized model:

- All words in the vocab  $V$  have equal probability  $\frac{1}{|V|}$

- $$\text{Perplexity}(D) = e^{\frac{\sum_{s \in D} \sum_{w_i \in s} \frac{-\log p(w_i^s | w_1^s \dots w_{i-1}^s)}{|s|}}{|D|}} = e^{\frac{\sum_{s \in D} \sum_{w_i \in s} \frac{-\log(\frac{1}{|V|})}{|s|}}{|D|}} = e^{-\log(\frac{1}{|V|})} = |V|$$

- i.e. predicting from a randomly-initialized model is equivalent to rolling a  $|V|$ -sided die (which is consistent with our intuition)

# Let's tie it together

$$\text{Perplexity}(D) = e^{\text{avg cross entropy loss for all words in } D}$$

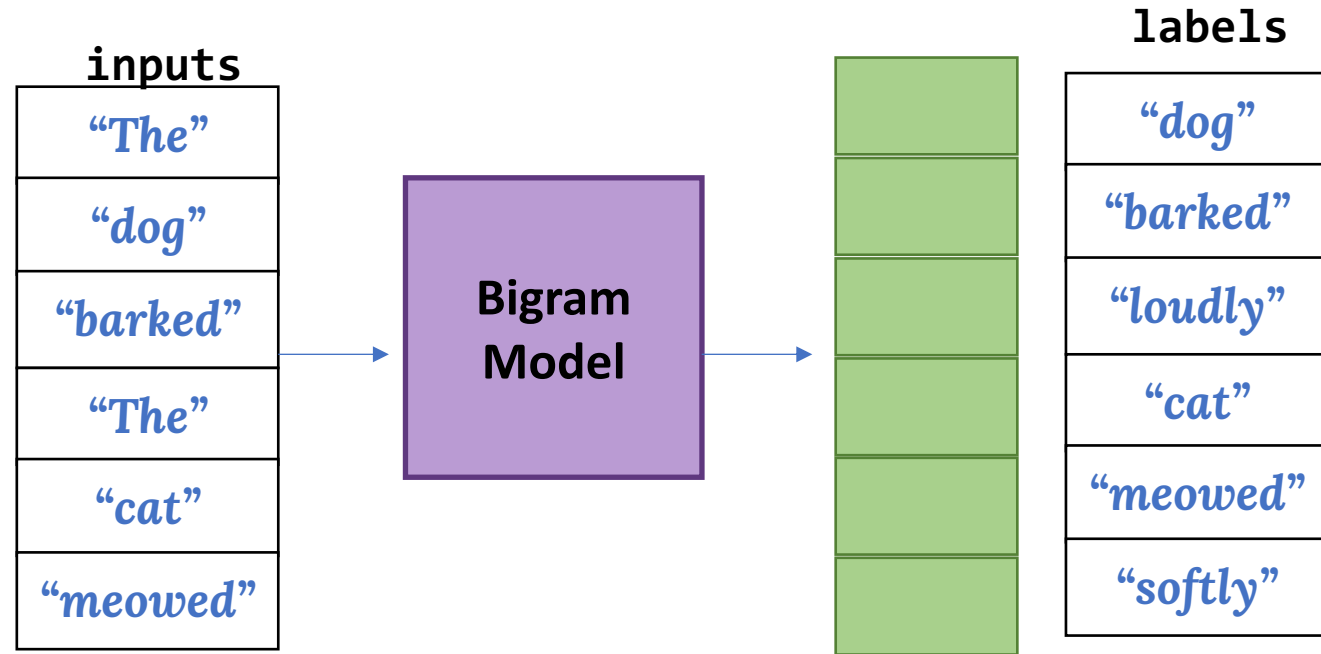
$V = \{\text{"the", "dog", "cat", "barked", "meowed", "was", "barking", "meowing", "loudly", "softly"}\}$



*"The dog barked loudly."*



*"The cat meowed softly"*



(1) What is the perplexity for a randomly initialized language model?

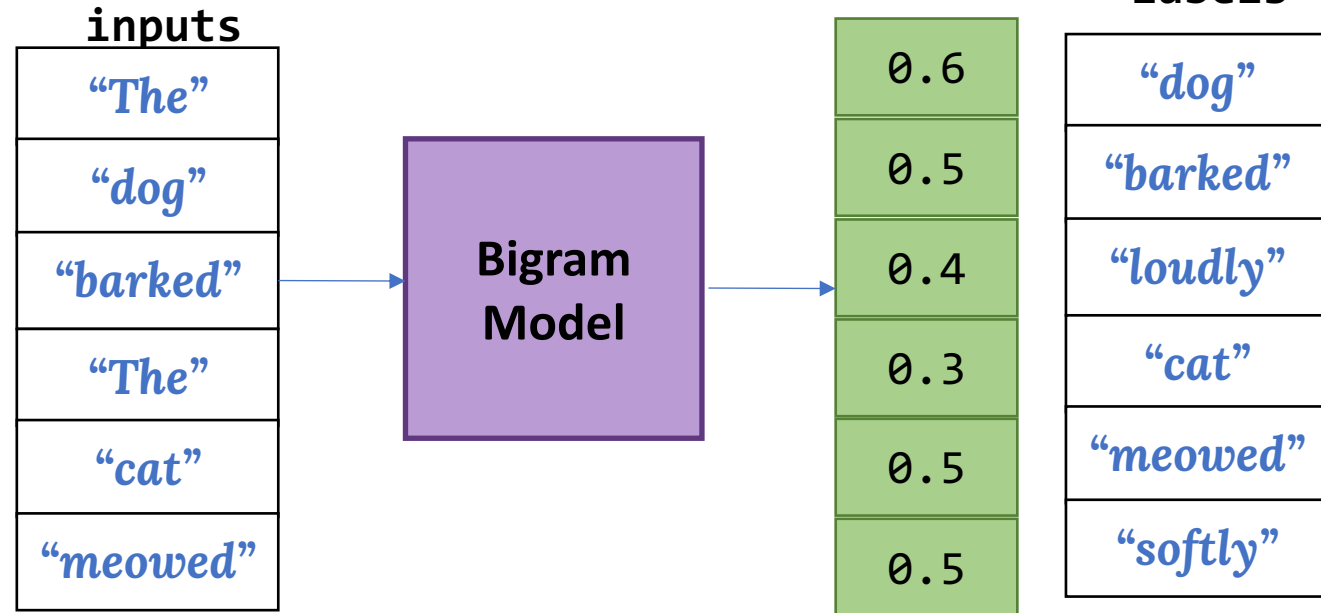
# Let's tie it together

$$\text{Perplexity}(D) = e^{\text{avg cross entropy loss for all words in } D}$$

$V = \{ \text{"the", "dog", "cat", "barked", "meowed", "was", "barking", "meowing", "loudly", "softly"} \}$



*"The dog barked loudly."*



*"The cat meowed softly"*

(2) What is the perplexity for a trained language model with the shown output probabilities?



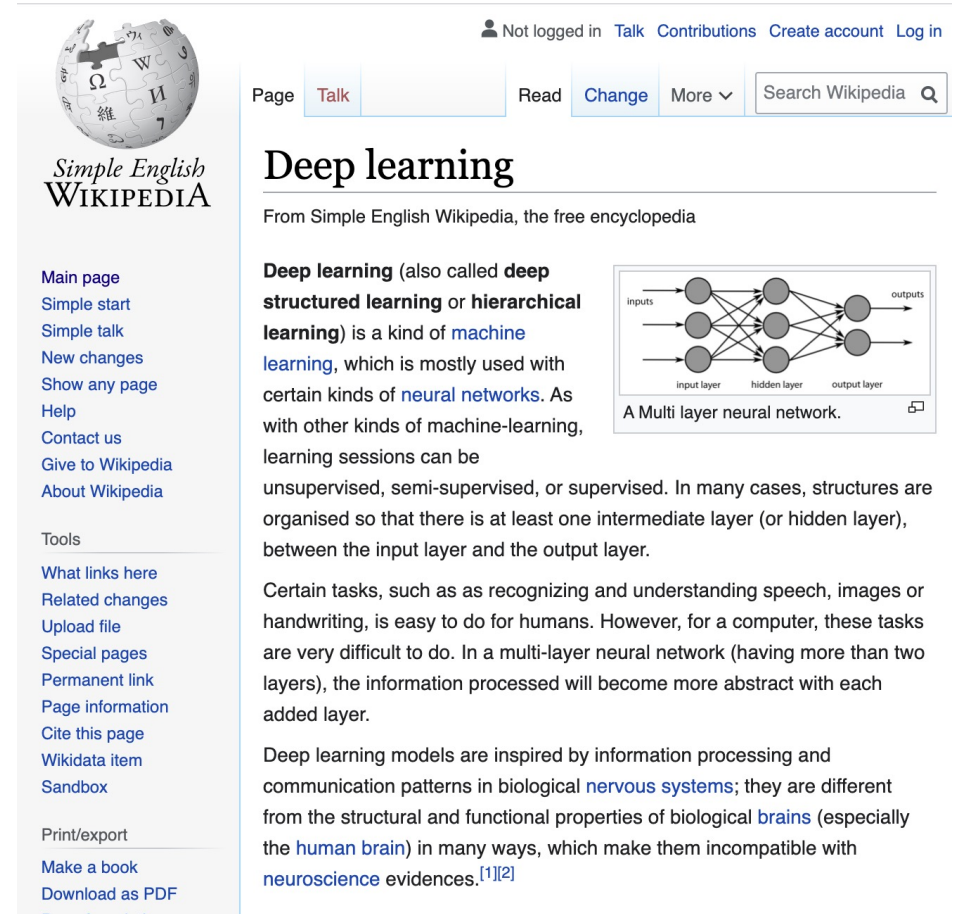
# Perplexity: Intuitive Meaning

- “If a model has a perplexity of  $X$ , then it has the same odds of predicting the correct next word as a fair die with  $X$  sides”
- Example: for a well-trained trigram model on a known NLP dataset (Penn Tree Bank with  $|V| \sim 10,000$ ):
  - Can expect perplexity  $< 240$ 
    - Much better to ‘guess’ words via a  $\sim 200$  sided die than a  $\sim 10,000$  sided die!
  - A perplexity threshold is what the hw4 autograder checks for, in fact ;)

Speaking of hw4...

# Hw4: Language Modeling

- Build and train a trigram language model
  - Perplexity  $< 165$
- Build and train a recurrent language model (next lecture!)
  - Perplexity  $< 95$
- Dataset
  - Articles scraped from [Simple English Wikipedia](#)
  - Focused on technology-related topics (for a smaller, more consistent vocabulary)

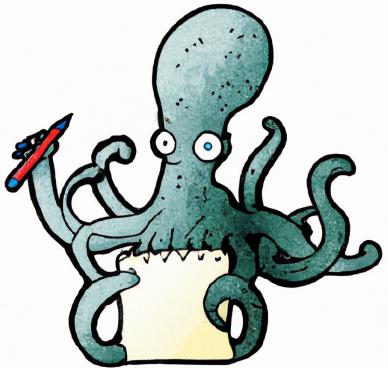


The screenshot shows the Simple English Wikipedia page for "Deep learning". At the top, it says "Not logged in" with links for "Talk", "Contributions", "Create account", and "Log in". Below that is a navigation bar with "Page", "Talk", "Read", "Change", "More", and a search box labeled "Search Wikipedia". The main heading is "Deep learning". Below the heading, it says "From Simple English Wikipedia, the free encyclopedia". The main text starts with "Deep learning (also called **deep structured learning** or **hierarchical learning**) is a kind of **machine learning**, which is mostly used with certain kinds of **neural networks**. As with other kinds of machine-learning, learning sessions can be unsupervised, semi-supervised, or supervised. In many cases, structures are organised so that there is at least one intermediate layer (or hidden layer), between the input layer and the output layer." To the right of this text is a diagram of a multi-layer neural network with three layers: an input layer with three nodes, a hidden layer with three nodes, and an output layer with three nodes. Arrows show connections between nodes in adjacent layers. Below the diagram is the caption "A Multi layer neural network." Below the main text, it says "Certain tasks, such as as recognizing and understanding speech, images or handwriting, is easy to do for humans. However, for a computer, these tasks are very difficult to do. In a multi-layer neural network (having more than two layers), the information processed will become more abstract with each added layer." At the bottom, it says "Deep learning models are inspired by information processing and communication patterns in biological **nervous systems**; they are different from the structural and functional properties of biological **brains** (especially the **human brain**) in many ways, which make them incompatible with **neuroscience** evidences.<sup>[1][2]</sup>" On the left side of the page, there is a sidebar with a globe icon and the text "Simple English WIKIPEDIA". Below that are links for "Main page", "Simple start", "Simple talk", "New changes", "Show any page", "Help", "Contact us", "Give to Wikipedia", and "About Wikipedia". At the bottom of the sidebar, there are links for "Tools", "What links here", "Related changes", "Upload file", "Special pages", "Permanent link", "Page information", "Cite this page", "Wikidata item", "Sandbox", "Print/export", "Make a book", and "Download as PDF".



# Recap

Feedforward  
Language Models



Evaluating  
Language  
Models

Bigram model

Limitations of bigram model

Trigram model

Qualitative assessment

Quantitative assessment

Perplexity and its intuition



*“The dog barked loudly.”*



*“The cat meowed softly”*