

# CMSC 473/673

# Natural Language Processing

---

Instructor: Lara J. Martin (she/they)

TA: Duong Ta (he)

*Slides modified from Dr. Frank Ferraro*

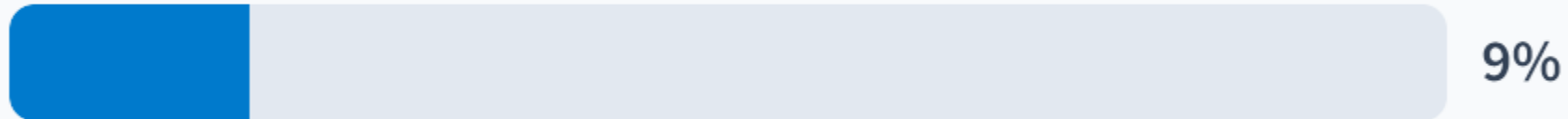
## What is smoothing (in language modeling)?

setting some words as UNK so that the model can deal with out-of-vocabulary words

setting some words as UNK so that the model can deal wi...



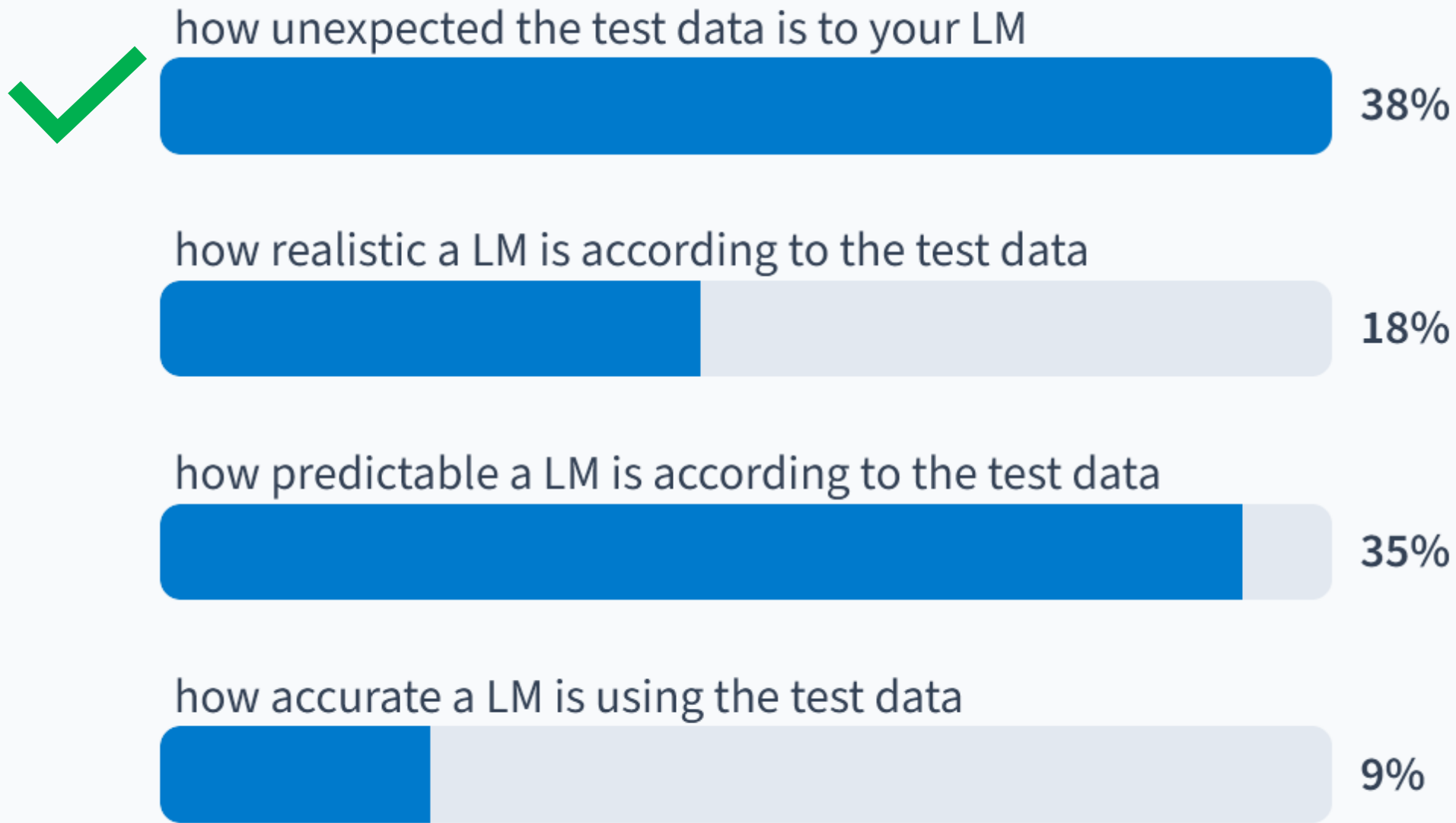
removing common words to even out the distribution



getting rid of zeroes in counts



## Perplexity is a measure of...



# Review: Add- $\lambda$ estimation

---

Other names: Laplace  
smoothing, Lidstone  
smoothing

Pretend we saw each word  $\lambda$   
more times than we did

Add  $\lambda$  to all the counts

$$p(\mathbf{z}) \cong \frac{\text{count}(\mathbf{z}) + \lambda}{\sum_v (\text{count}(v) + \lambda)}$$

# Review: An Extended Trigram Example

The film got a great opening and the film went on to become a hit .

Context: x y	Word (Type): z	Raw Count	Add-1 count	Norm.	Probability $p(z   x y)$	
The film	The	0	1	17 (=1+16*1)	1/17	
The film	film	0	1		1/17	
The film	got	1	2		2/17	
The film	went	0	1		1/17	
...					...	
The film	OOV	0	1		1/17	
The film	EOS	0	1		1/17	
...					...	
a great	great	0	1	17	1/17	
a great	opening	1	2		2/17	
a great	and	0	1		1/17	
a great	the	0	1		1/17	
...					...	

# Review: Language Model with Maxent n-grams

$$p_n(\text{📄} | y) = \prod_{i=1}^M \text{maxent}(y, \underbrace{x_{i-n+1:i-1}, x_i}_{\text{n-gram}})$$

Diagram annotations: An orange arrow labeled "label" points to the  $y$  argument of the maxent function. An orange bracket labeled "n-gram" spans the  $x_{i-n+1:i-1}, x_i$  arguments.

Iterate through all possible output vocab types  $x'$ ---just like in count-based LMs

$$= \prod_{i=1}^M \frac{\exp(\theta_{x_i}^T f(y, x_{i-n+1:i-1}))}{\sum_{x'} \exp(\theta_{x'}^T f(y, x_{i-n+1:i-1}))}$$

Diagram annotation: A red arrow points from the text "Iterate through all possible output vocab types  $x'$ ---just like in count-based LMs" to the summation symbol  $\sum_{x'}$  in the denominator of the equation.

# Review: Maxent Language Models

*given some context...*



*compute beliefs about what is likely...*

$$p(w_i | w_{i-3}, w_{i-2}, w_{i-1}) = \text{softmax}(\theta_{w_i} \cdot f(w_{i-3}, w_{i-2}, w_{i-1}))$$

*predict the next word*



Review:

A Closer Look at Maxent  $p(\text{Won't you please donate?} \mid \text{Primary})$

This is a *class-based* language model, but incorporate the label into the features

To learn  $p(\text{Won't you please donate?} \mid \text{Class})$ :

Define features  $f$  that make use of the specific label **Class**

Unlike count-based models, you don't *need* "separate" models here



# Maxent Language Models

given some context...



compute beliefs about what is likely...

$$p(w_i | w_{i-3}, w_{i-2}, w_{i-1}) = \text{softmax}(\theta_{w_i} \cdot f(w_{i-3}, w_{i-2}, w_{i-1}))$$

predict the next word

can we learn word-specific weights (by type)?



# Neural Language Models

given some context...



can we learn the feature function(s) for just the context?

compute beliefs about what is likely...



$$p(w_i | w_{i-3}, w_{i-2}, w_{i-1}) = \text{softmax}(\theta_{w_i} \cdot f(w_{i-3}, w_{i-2}, w_{i-1}))$$

predict the next word

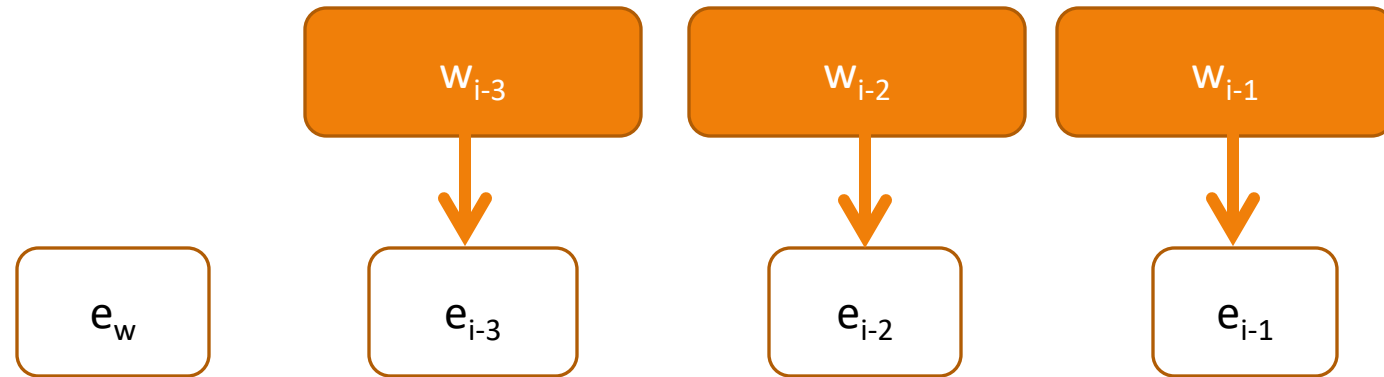
can we learn word-specific weights (by type)?



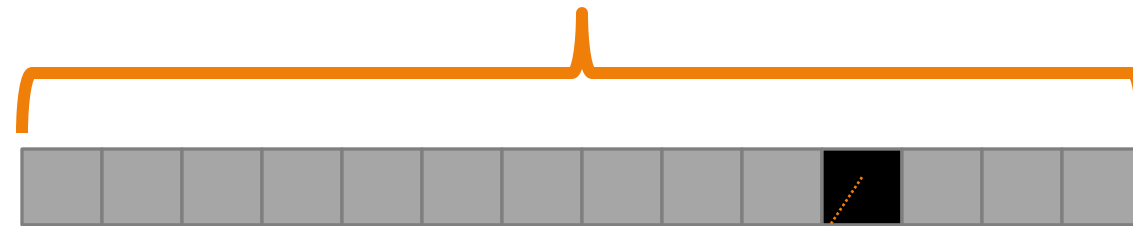
# Neural Language Models

given some context...

create/use  
“distributed  
representations” ...



compute beliefs about  
what is likely...



$$p(w_i | w_{i-3}, w_{i-2}, w_{i-1}) = \text{softmax}(\theta_{w_i} \cdot f(w_{i-3}, w_{i-2}, w_{i-1}))$$

predict the next word



# Neural Language Models

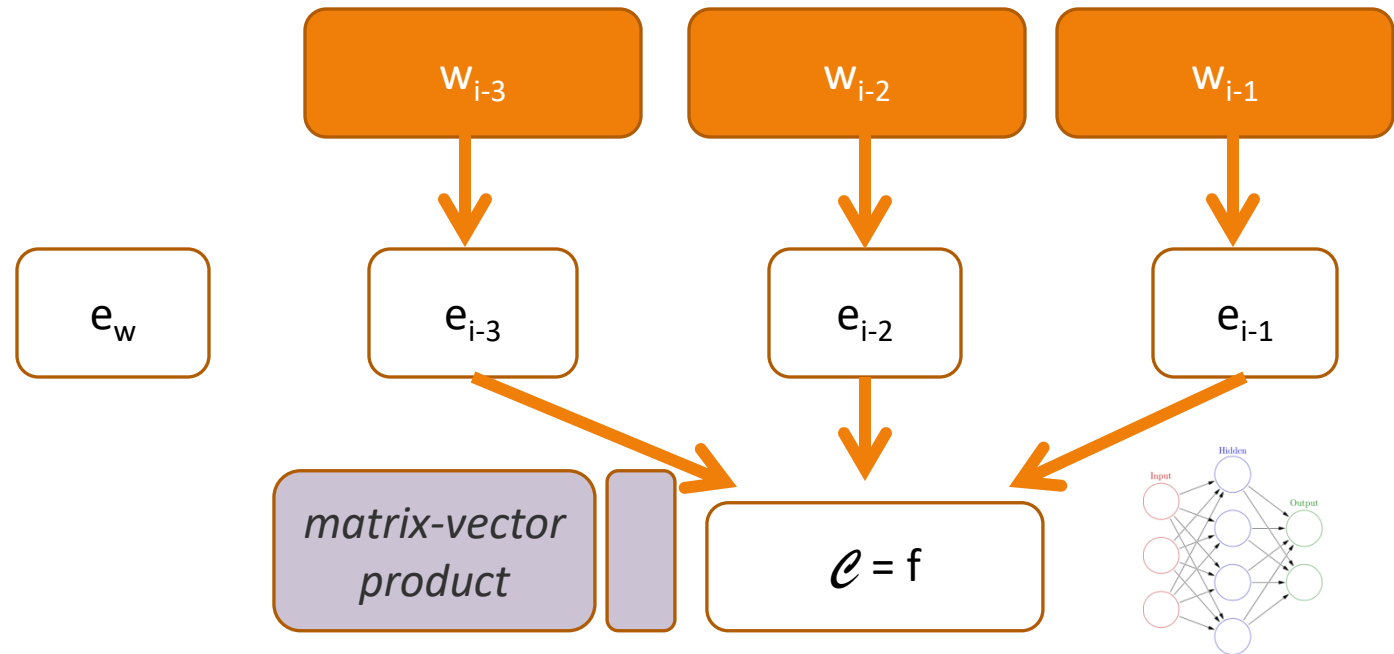
given some context...

create/use  
“distributed  
representations” ...

combine these  
representations...

compute beliefs about  
what is likely...

predict the next word



A horizontal bar represents a sequence of words. The bar is divided into segments. The segment corresponding to the word  $w_i$  is highlighted in black. A bracket above the bar spans from the first word to the last word. A dashed arrow points from the highlighted word to the  $w_i$  box in the bottom diagram.

$$p(w_i | w_{i-3}, w_{i-2}, w_{i-1}) = \text{softmax}(\theta_{w_i} \cdot f(w_{i-3}, w_{i-2}, w_{i-1}))$$



# Neural Language Models

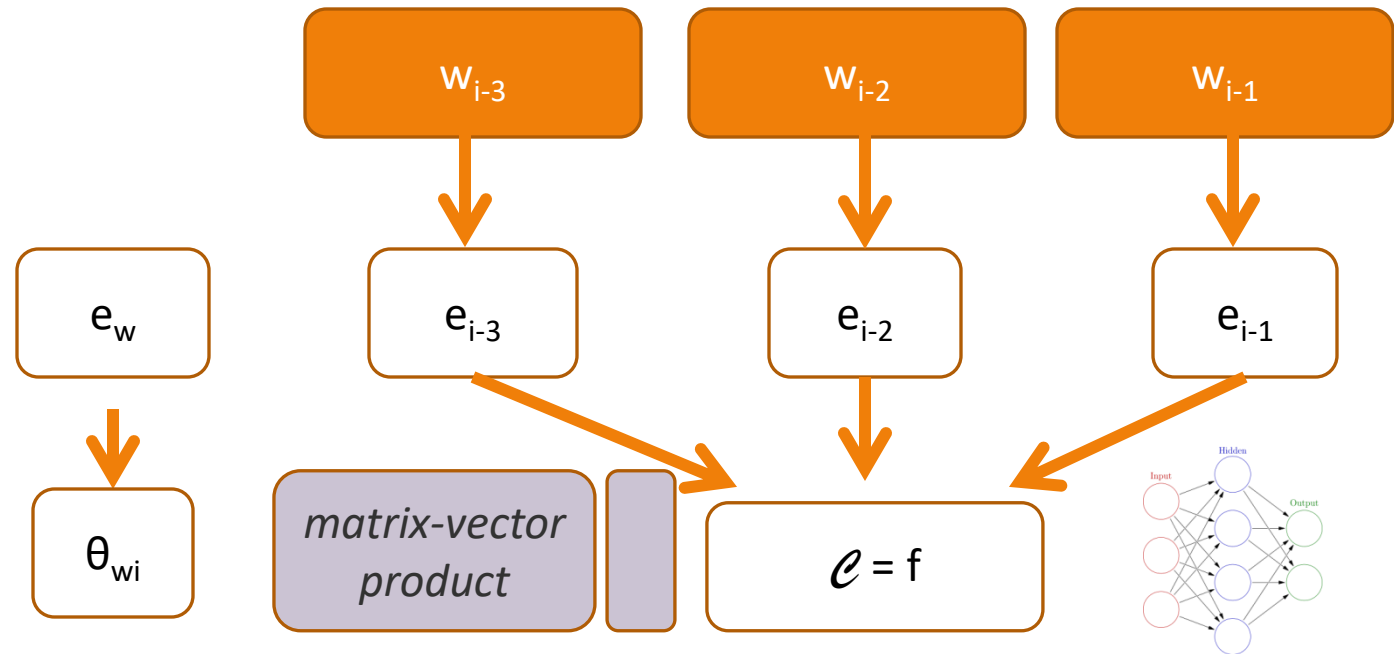
given some context...

create/use  
“distributed  
representations” ...

combine these  
representations...

compute beliefs about  
what is likely...

predict the next word



The diagram shows a horizontal bar representing a probability distribution over a sequence of words. A bracket above the bar indicates the context used for prediction. The equation below the diagram is:

$$p(w_i | w_{i-3}, w_{i-2}, w_{i-1}) = \text{softmax}(\theta_{w_i} \cdot f(w_{i-3}, w_{i-2}, w_{i-1}))$$



# Neural Language Models

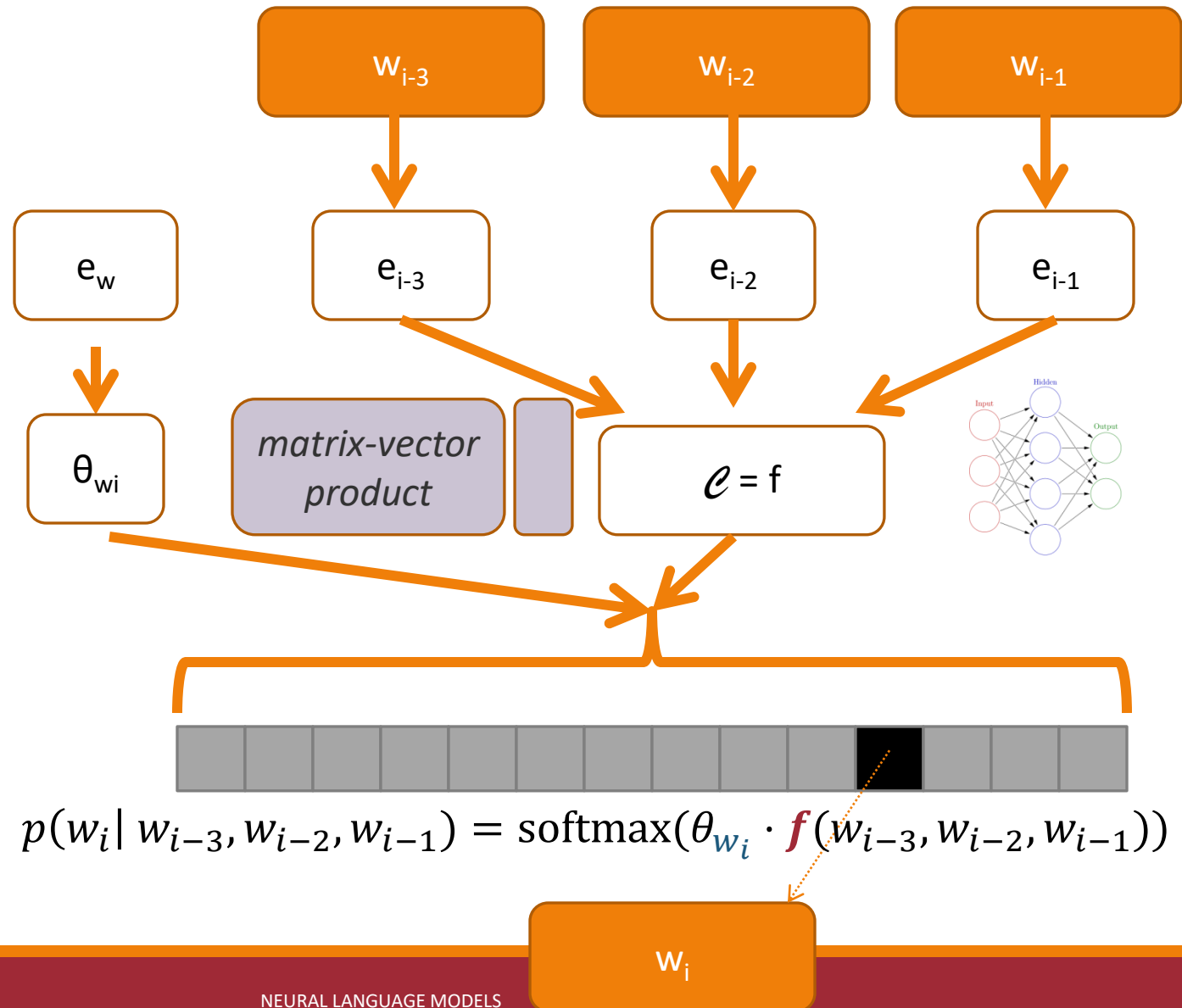
given some context...

create/use  
“distributed  
representations” ...

combine these  
representations...

compute beliefs about  
what is likely...

predict the next word



# A Neural N-Gram Model

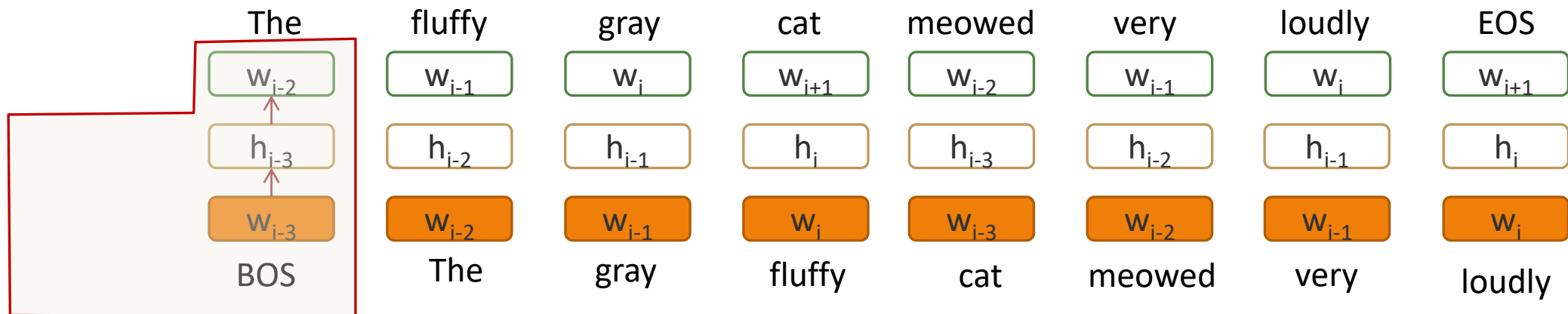
---

The fluffy gray cat meowed very loudly



# A Neural N-Gram Model (N=3)

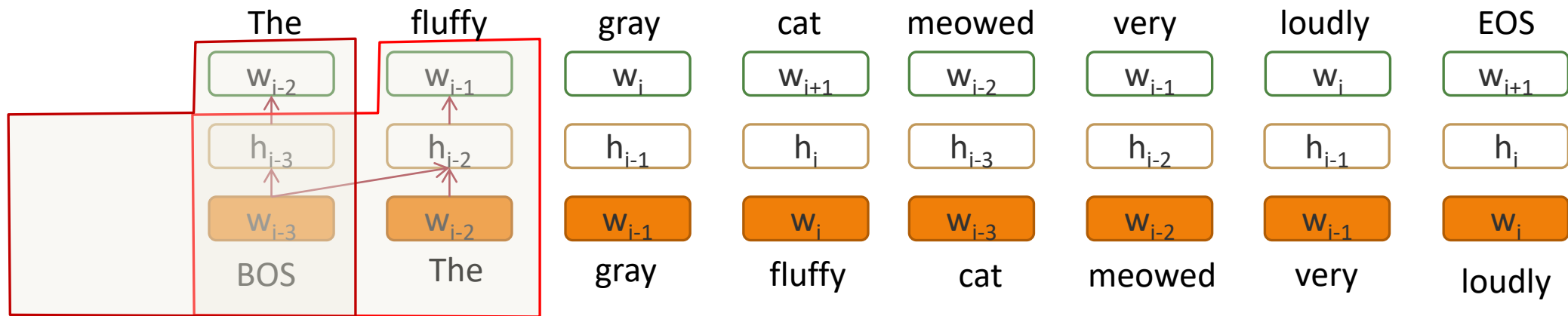
The fluffy gray cat meowed very loudly





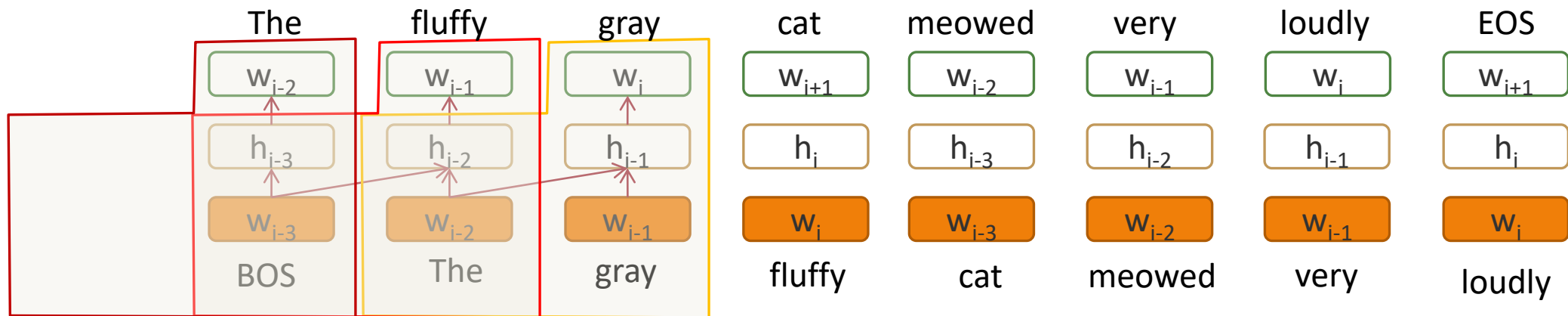
# A Neural N-Gram Model (N=3)

The fluffy gray cat meowed very loudly



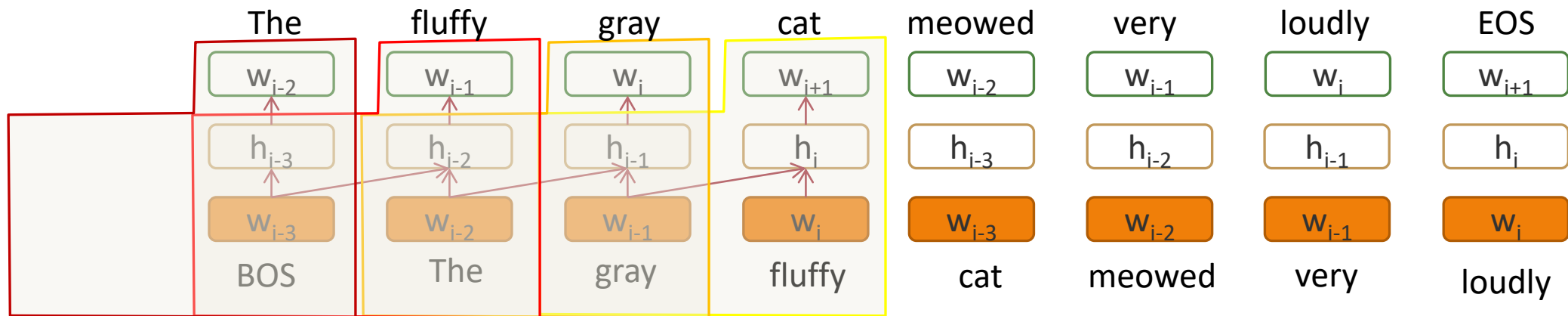
# A Neural N-Gram Model (N=3)

The fluffy gray cat meowed very loudly



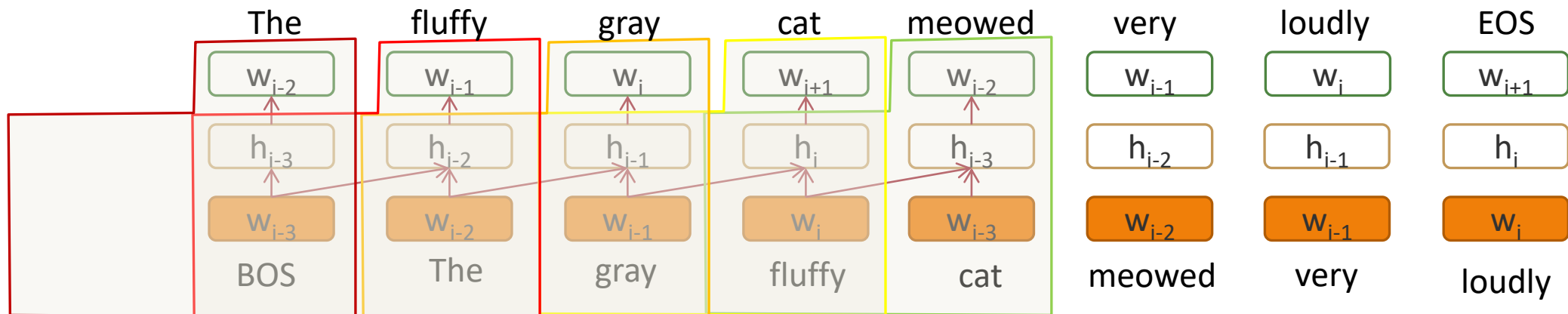
# A Neural N-Gram Model (N=3)

The fluffy gray cat meowed very loudly



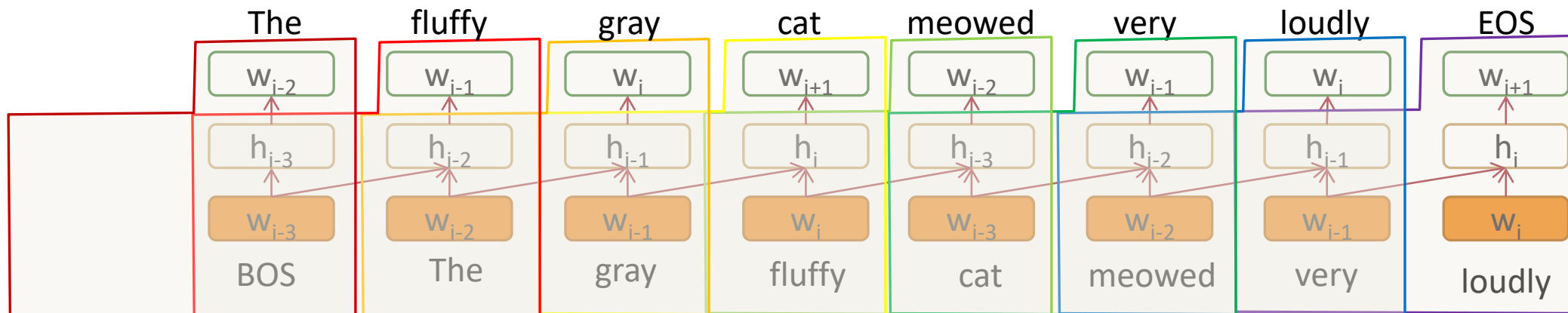
# A Neural N-Gram Model (N=3)

The fluffy gray cat meowed very loudly

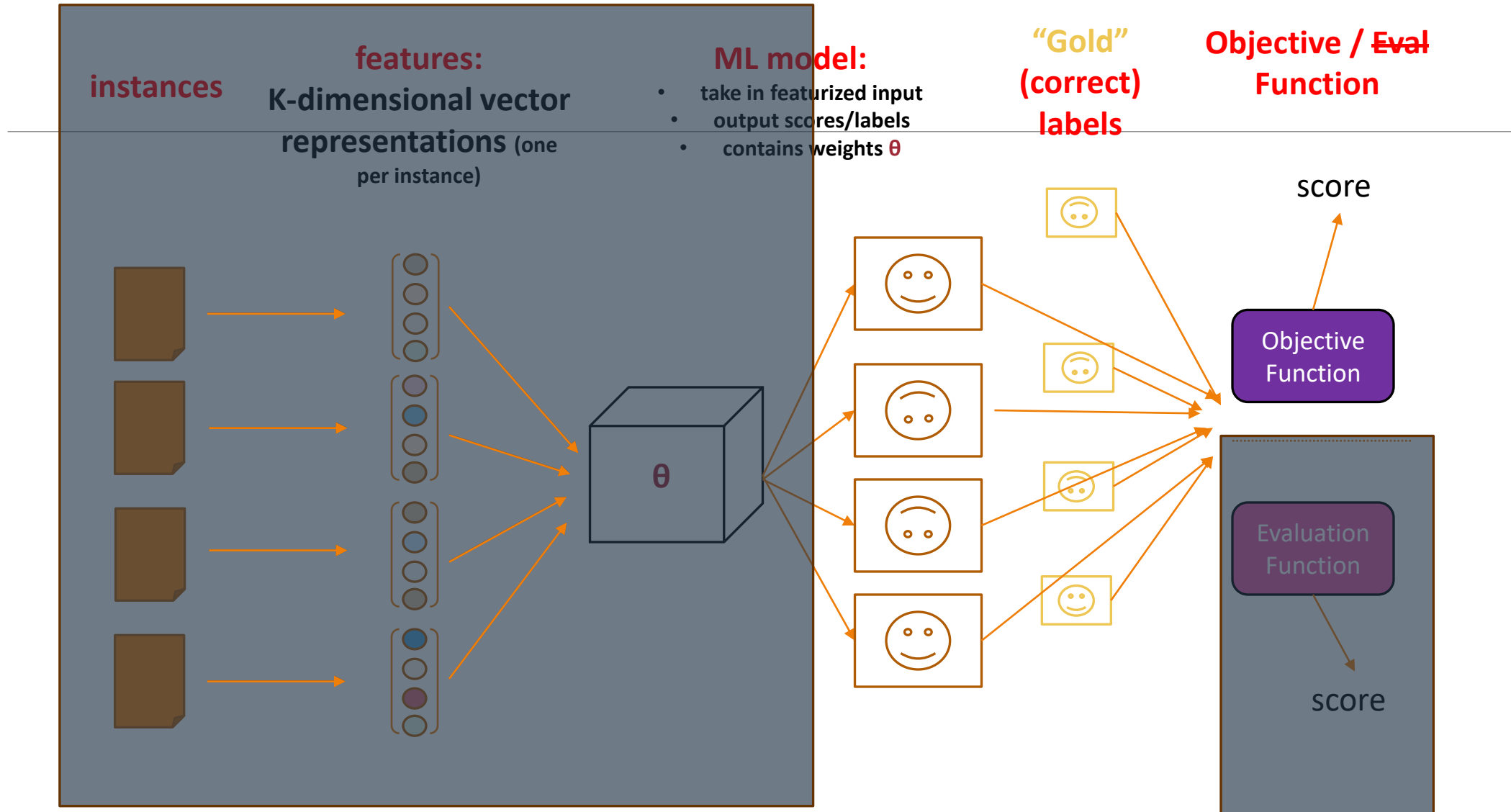


# A Neural N-Gram Model (N=3)

The fluffy gray cat meowed very loudly



# Defining the Objective



# Review:

## Maximize Log-Likelihood (Classification)

$$\log \prod_i p_{\theta}(y_i | x_i) = \sum_i \log p_{\theta}(y_i | x_i)$$

*Inverse of exp*  
 $\log(\exp(x)) = x$

$$= \sum_i \theta_{y_i}^T f(x_i) - \log Z(x_i)$$

$$= F(\theta)$$

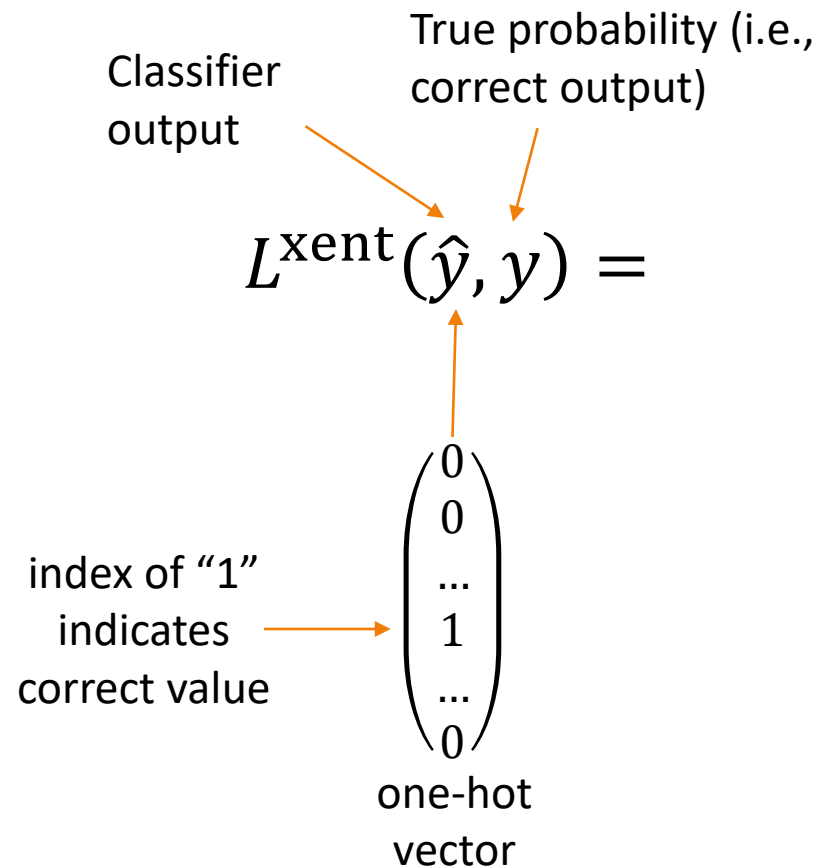
Original maxent equation

$$\frac{\exp(\theta_y^T f(x))}{\sum_{y'} \exp(\theta_{y'}^T f(x))}$$

Differentiating this becomes nicer (even though Z depends on  $\theta$ )

# Review:

## *Minimize Cross Entropy Loss*



**Cross entropy:**  
How much  $\hat{y}$  differs from the true  $y$

objective is convex  
(when  $f(x)$  is not learned)





# “A Neural Probabilistic Language Model,” Bengio et al. (2003)

---

## BASELINES

LM Name	N-gram	Params.	Test PPL
Interpolation	3	---	336
Kneser-Ney backoff	3	---	323
Kneser-Ney backoff	5	---	321
Class-based backoff	3	500 classes	312
Class-based backoff	5	500 classes	312

# “A Neural Probabilistic Language Model,” Bengio et al. (2003)

BASELINES

LM Name	N-gram	Params.	Test PPL
Interpolation	3	---	336
Kneser-Ney backoff	3	---	323
Kneser-Ney backoff	5	---	321
Class-based backoff	3	500 classes	312
Class-based backoff	5	500 classes	312

NPLM

N-gram	Word Vector Dim.	Hidden Dim.	Mix with non-neural LM	PPL
5	60	50	No	268
5	60	50	Yes	257
5	30	100	No	276
5	30	100	Yes	252

*“we were not able to see signs of over-fitting (on the validation set), possibly because we ran only 5 epochs (over 3 weeks using 40 CPUs)” (Sect. 4.2)*

# A Closer Look at Neural $p(\text{Won't you please donate?} \mid \text{Primary})$

This is a *class-based* language model, but incorporate the label into the *embedding representation*

To learn  $p(\text{Won't you please donate?} \mid \text{Class})$ :

Define an embedding method that makes use of the specific label **Class**

Unlike count-based models, you don't *need* “separate” models here

# LM Comparison for $p(\text{Won't you please donate?} \mid \text{Primary})$

N-GRAM/COUNT-BASED

MAXENT/LR

NEURAL

Class-specific

Class-based

Class-based

Uses features

Uses *embedded* features