(Generative) Language Modeling

Frank Ferraro – ferraro@umbc.edu

CMSC 473/673

Some slides adapted from 3SLP, Jason Eisner

Goal of Language Modeling

$$p_\theta(\ [...text..]\ )$$

Learn a probabilistic model of text

Accomplished through observing text and updating model parameters to make text more likely

Two Perspectives: Prediction vs. Generation

Prediction

Given observed word tokens $w_1 \ldots w_{N-1}$, create a classifier p to predict the next word w_N

$$p(w_N = v \mid w_1 \ldots w_{N-1})$$

Generation

Two Perspectives: Prediction vs. Generation

Prediction

Given observed word tokens $w_1 \dots w_{N-1}$, create a classifier p to predict the next word w_N

$$p(w_N = v \mid w_1 \dots w_{N-1}), \text{ e.g.,}$$

$$p(w_N = \text{meowed} \mid \text{The, fluffy, cat})$$

Generation

Two Perspectives: Prediction vs. Generation

Prediction

Given observed word tokens $w_1 \ldots w_{N-1}$, create a classifier p to predict the next word w_N

$$p(w_N = v \mid w_1 \ldots w_{N-1}), \text{ e.g.,}$$
$$p(w_N = \text{meowed} \mid \text{The, fluffy, cat})$$

Generation

Develop a probabilistic model p to *explain/score* the word sequence $w_1 \ldots w_N$

$$p(w_1 \ldots w_N), \text{ e.g.,}$$
$$p(\text{The, fluffy, cat, meowed})$$

Design Question 1: What Part of Language Do We Estimate?

$$p_\theta(\ [...text..]\ )$$

Is *[…text..]* a
- Full document?
- Sequence of sentences?
- Sequence of words?
- Sequence of characters?

A: It's task-dependent!

Design Question 2: How do we estimate robustly?

$$p_\theta(\textit{[...typo-text..]})$$

What if *[...text..]* has a typo?

Design Question 3: How do we generalize?

$$p_\theta(\textit{[...synonymous-text..]})$$

What if *[...text..]* has a word (or character or...) we've never seen before?

Key Idea: Probability Chain Rule

$$p(x_1, x_2, \dots, x_S) =$$

$$p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1, x_2) \cdots p(x_S \mid x_1, \dots, x_{S-1})$$

Key Idea: Probability Chain Rule

$$p(x_1, x_2, \ldots, x_S) =$$
$$p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1, x_2) \cdots p(x_S \mid x_1, \ldots, x_{S-1}) =$$
$$\prod_{i}^{S} \underbrace{p(x_i \mid x_1, \ldots, x_{i-1})}$$

Language modeling is about how to estimate each of these factors in {great, good, sufficient, …} ways

Problem: Develop a Probabilistic Email Classifier

Input: an email (all text)

Output (Google categories):

Primary, Social, Forums, Spam

$$\operatorname{argmax}_{\mathrm{y}}\, p(\text{label } Y = y \mid \text{email } X)$$

Approach #1: Discriminatively trained

Approach #2: Using Bayes rule

Classify Using Bayes Rule

$$p(\text{label } Y \mid \text{email } X) \propto p(X \mid Y) * p(Y)$$

Classify Using Bayes Rule

$$p(\text{label } Y \mid \text{email } X) \propto p(X \mid Y) * p(Y)$$

Q: Why is p(Y | X) what we want to model?

Classify Using Bayes Rule

$$p(\text{label } Y \mid \text{email } X) \propto p(X \mid Y) * p(Y)$$

$$p\left(\text{Primary} \,\middle|\, \text{Won't you please donate?}\right) \propto p\left(\text{Won't you please donate?} \,\middle|\, \text{Primary}\right) p\left(\text{Primary}\right)$$

A Closer Look at $p($Primary$)$

This is the **prior probability** of each *class*

Answers the question: without knowing anything specific about a document, how likely is each class?

A Closer Look at $p(\text{Primary})$

This is the **prior probability** of each *class*

Answers the question: without knowing anything specific about a document, how likely is each class?

Q: What's an easy way to estimate it?

A Closer Look at $p($ Won't you please donate? $|$ Primary $)$

This is a *class specific* language model

A Closer Look at $p($ Won’t you please donate? $|$ Primary $)$

This is a *class specific* language model

$p($ Won’t you please donate? $|$ Primary $)$ is different from

$p($ Won’t you please donate? $|$ Social $)$ is different from

$p($ Won’t you please donate? $|$ Forums $)$ …

A Closer Look at $p($ Won’t you please donate? $|$ Primary $)$

This is a *class specific* language model

To learn $p($ Won’t you please donate? $|$ Class $)$:

For each class Class:

- Get a bunch of Class documents D_{Class}
- Learn a new language model p_{Class} on just D_{Class}

Language Models & Smoothing

- Maximum likelihood (MLE): simple counting
- Other count-based models
  - Laplace smoothing, add- λ
  - Interpolation models
  - Discounted backoff
  - Interpolated (modified) Kneser-Ney
  - Good-Turing
  - Witten-Bell
- Maxent n-gram models
- Neural n-gram models
- Recurrent/autoregressive NNs

Easy to implement

Advanced/ out of scope

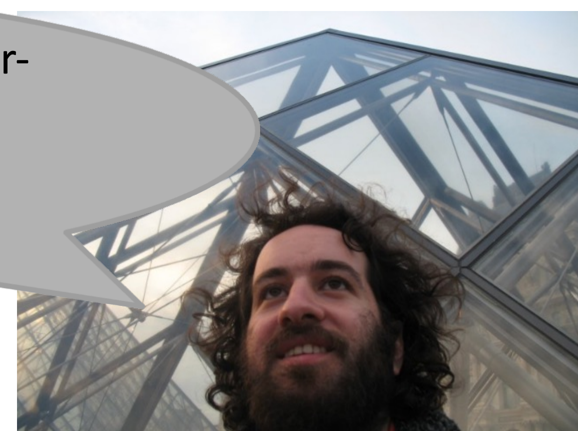
Featureful LMs

Feedforward LMs

Super modern

“The Unreasonable Effectiveness of
Recurrent Neural Networks”
http://karpathy.github.io/2015/05/21/rnn-effectiveness/

“The Unreasonable Effectiveness of Character-
level Language Models”
(and why RNNs are still cool)
http://nbviewer.jupyter.org/gist/yoavg/d76121dfde2618422139



Language Models & Smoothing

- **Maximum likelihood (MLE): simple counting**
- Other count-based models
  - Laplace smoothing, add- λ
  - Interpolation models
  - Discounted backoff
  - Interpolated (modified) Kneser-Ney
  - Good-Turing
  - Witten-Bell
- Maxent n-gram models
- Neural n-gram models
- Recurrent/autoregressive NNs

Easy to implement

Advanced/ out of scope

Featureful LMs

Feedforward LMs

Super modern

N-Grams

Maintaining an entire inventory over sentences could be too much to ask

Store “smaller” pieces?

p(Colorless green ideas sleep furiously)

N-Grams

Maintaining an entire *joint* inventory over sentences could be too much to ask

Store “smaller” pieces?

p(Colorless green ideas sleep furiously) =
p(Colorless) *

N-Grams

Maintaining an entire *joint* inventory over sentences could be too much to ask

Store "smaller" pieces?

p(Colorless green ideas sleep furiously) =
p(Colorless) *
p(green | Colorless) *

N-Grams

Maintaining an entire *joint* inventory over sentences could be too much to ask

Store “smaller” pieces?

p(Colorless green ideas sleep furiously) =
p(Colorless) *
p(green | Colorless) *
p(ideas | Colorless green) *
p(sleep | Colorless green ideas) *
p(furiously | Colorless green ideas sleep)

N-Grams

p(furiously | Colorless green ideas sleep)

How much does “Colorless” influence the choice of “furiously?”

N-Grams

p(furiously | Colorless green ideas sleep)

How much does “Colorless” influence the choice of “furiously?”

Remove history and contextual info

N-Grams

p(furiously | Colorless green ideas sleep)

How much does “Colorless” influence the choice of “furiously?”

Remove history and contextual info

p(furiously | Colorless green ideas sleep) ≈
p(furiously | ~~Colorless green~~ ideas sleep)

N-Grams

p(furiously | Colorless green ideas sleep)

How much does "Colorless" influence the choice of "furiously?"

Remove history and contextual info

p(furiously | Colorless green ideas sleep) ≈
p(furiously | ideas sleep)

N-Grams

p(Colorless green ideas sleep furiously) =
p(Colorless) *
p(green | Colorless) *
p(ideas | Colorless green) *
p(sleep | Colorless green ideas) *
p(furiously | Colorless green ideas sleep)

N-Grams

p(Colorless green ideas sleep furiously) =
p(Colorless) *
p(green | Colorless) *
p(ideas | Colorless green) *
p(sleep | ~~Colorless~~ green ideas) *
p(furiously | ~~Colorless green~~ ideas sleep)

Trigrams

p(Colorless green ideas sleep furiously) =
p(Colorless) *
p(green | Colorless) *
p(ideas | Colorless green) *
p(sleep | green ideas) *
p(furiously | ideas sleep)

Trigrams

p(Colorless green ideas sleep furiously) =
p(Colorless) *
p(green | Colorless) *
p(ideas | Colorless green) *
p(sleep | green ideas) *
p(furiously | ideas sleep)

Trigrams

p(Colorless green ideas sleep furiously) =
p(Colorless | *<BOS> <BOS>*) *
p(green | *<BOS>* Colorless) *
p(ideas | Colorless green) *
p(sleep | green ideas) *
p(furiously | ideas sleep)

Consistent notation: Pad the left with <BOS> (beginning of sentence) symbols

Trigrams

p(Colorless green ideas sleep furiously) =
p(Colorless | *<BOS> <BOS>*) *
p(green | *<BOS>* Colorless) *
p(ideas | Colorless green) *
p(sleep | green ideas) *
p(furiously | ideas sleep) *
p(*<EOS>* | sleep furiously)

Consistent notation: Pad the left with <BOS> (beginning of sentence) symbols
Fully proper distribution: Pad the right with a single <EOS> symbol

N-Gram Terminology

n	Commonly called	History Size (Markov order)	Example
1	unigram	0	p(furiously)

N-Gram Terminology

n	Commonly called	History Size (Markov order)	Example
1	unigram	0	p(furiously)
2	bigram	1	p(furiously | sleep)

N-Gram Terminology

n	Commonly called	History Size (Markov order)	Example
1	unigram	0	p(furiously)
2	bigram	1	p(furiously | sleep)
3	trigram (3-gram)	2	p(furiously | ideas sleep)

N-Gram Terminology

n	Commonly called	History Size (Markov order)	Example
1	unigram	0	p(furiously)
2	bigram	1	p(furiously | sleep)
3	trigram (3-gram)	2	p(furiously | ideas sleep)
4	4-gram	3	p(furiously | green ideas sleep)
n	n-gram	n-1	$p(w_i \mid w_{i-n+1} \ldots w_{i-1})$

N-Gram Probability

$$p(w_1, w_2, w_3, \cdots, w_S) =$$

$$\prod_{i=1}^{S} p(w_i | w_{i-N+1}, \cdots, w_{i-1})$$

Count-Based N-Grams (Unigrams)

$$p(\text{item}) \propto count(\text{item})$$

Count-Based N-Grams (Unigrams)

$$p(\mathrm{z}) \propto count(\mathrm{z})$$

Count-Based N-Grams (Unigrams)

word type word type

$$p(\mathrm{z}) \propto count(\mathrm{z})$$

$$= \frac{count(\mathrm{z})}{\sum_v count(\mathrm{v})}$$

word type

Count-Based N-Grams (Unigrams)

word type

word type

$$p(z) \propto count(z)$$

$$= \frac{count(z)}{W}$$

number of tokens observed

Count-Based N-Grams (Unigrams)

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

Word (Type) z	Raw Count count(z)	Normalization	Probability p(z)
The	1		
film	2		
got	1		
a	2		
great	1		
opening	1		
and	1		
the	1		
went	1		
on	1		
to	1		
become	1		
hit	1		
.	1		

Count-Based N-Grams (Unigrams)

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

Word (Type) z	Raw Count count(z)	Normalization	Probability p(z)
The	1	16	
film	2		
got	1		
a	2		
great	1		
opening	1		
and	1		
the	1		
went	1		
on	1		
to	1		
become	1		
hit	1		
.	1		

Count-Based N-Grams (Unigrams)

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

Word (Type) z	Raw Count count(z)	Normalization	Probability p(z)
The	1	16	1/16
film	2		1/8
got	1		1/16
a	2		1/8
great	1		1/16
opening	1		1/16
and	1		1/16
the	1		1/16
went	1		1/16
on	1		1/16
to	1		1/16
become	1		1/16
hit	1		1/16
.	1		1/16

Count-Based N-Grams (Trigrams)

order matters in conditioning

order matters in *count*

$$p(\mathrm{z}|\mathrm{x},\mathrm{y}) \propto count(\mathrm{x},\mathrm{y},\mathrm{z})$$

Count of the sequence of items "x y z"

Count-Based N-Grams (Trigrams)

order matters in conditioning

order matters in *count*

$$p(\mathrm{z}|\mathrm{x},\mathrm{y}) \propto count(\mathrm{x},\mathrm{y},\mathrm{z})$$

count(x, y, z) ≠ count(x, z, y) ≠ count(y, x, z) ≠ …

Count-Based N-Grams (Trigrams)

$$p(\mathrm{z}|\mathrm{x},\mathrm{y}) \propto count(\mathrm{x},\mathrm{y},\mathrm{z})$$

$$= \frac{count(\mathrm{x},\mathrm{y},\mathrm{z})}{\sum_{v} count(\mathrm{x},\mathrm{y},\mathrm{v})}$$

Count-Based N-Grams (Trigrams)

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

Context: x y	Word (Type): z	Raw Count	Normalization	Probability p(z | x y)
The film	The	0	1	0/1
The film	film	0		0/1
The film	got	1		1/1
The film	went	0		0/1
…				
a great	great	0	1	0/1
a great	opening	1		1/1
a great	and	0		0/1
a great	the	0		0/1
…				

Count-Based N-Grams (Lowercased Trigrams)

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

Context: x y	Word (Type): z	Raw Count	Normalization	Probability: p(z | x y)
the film	the	0	2	0/2
the film	film	0		0/2
the film	got	1		1/2
the film	went	1		1/2
		…		
a great	great	0	1	0/1
a great	opening	1		1/1
a great	and	0		0/1
a great	the	0		0/1
		…		

Implementation: EOS Padding

Create an end of sentence ("chunk") token <EOS>

Don't estimate p(<BOS> | <EOS>)

Training & Evaluation:

1. Identify "chunks" that are relevant (sentences, paragraphs, documents)
2. Append the <EOS> token to the end of the chunk
3. Train or evaluate LM as normal

Implementation: Memory Issues

Let V = vocab size, W = number of ***observed*** n-grams

Often, $W \ll V$

Dense count representation: $O(V^n)$, but many entries will be zero

Sparse count representation: $O(W)$

Sometimes selective precomputation is helpful (e.g., normalizers)

Implementation: Unknown words

Create an unknown word token <UNK>

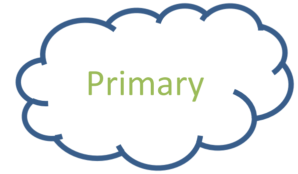
Training:

1. Create a fixed lexicon L of size V
2. Change any word not in L to <UNK>
3. Train LM as normal

Evaluation:

Use UNK probabilities for any word not in training

A Closer Look at Count-based $p($ 

$|$ 

$)$

This is a *class specific* language model

To learn $p($ 

$|$ 

$)$:

For each class Class:

- Get a bunch of Class documents D_{Class}
- Learn a new language model p_{Class} on just D_{Class}

Two Ways to Learn Class-specific Count-based Language Models

1. Create different count table(s) $c_{\text{Class}}(\ldots)$ for each Class

    e.g., record separate trigram counts for Primary vs. Social vs. Forums vs. Spam

Two Ways to Learn Class-specific Count-based Language Models

1. Create different count table(s) $c_{\text{Class}}(\dots)$ for each Class

   e.g., record separate trigram counts for Primary vs. Social vs. Forums vs. Spam

OR

2. Add a dimension to your existing tables $c(\text{Class}, \dots)$

   e.g., record how often each trigram occurs within Primary vs. Social vs. Forums vs. Spam documents

Evaluating Language Models

What is "correct?"

What is working "well?"

fine-tune any secondary (hyper)parameters

Training Data

Dev Data

Test Data

learn model parameters:
- acquire primary statistics
- learn feature weights

perform final evaluation

DO NOT TUNE ON THE TEST DATA

Evaluating Language Models

What is "correct?"

What is working "well?"

Extrinsic: Evaluate LM in downstream task

Test an MT, ASR, etc. system and see which LM does better

Propagate & conflate errors

$$p(\text{label } Y \mid \text{doc } X) \propto p(X \mid Y) * p(Y)$$

Evaluating Language Models

What is "correct?"

What is working "well?"

Extrinsic: Evaluate LM in downstream task

Test an MT, ASR, etc. system and see which LM does better

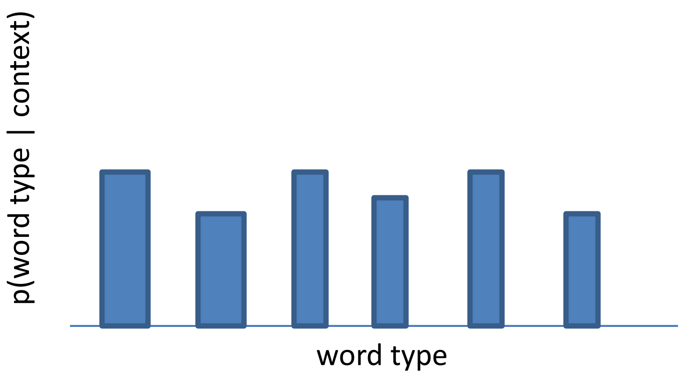
Propagate & conflate errors

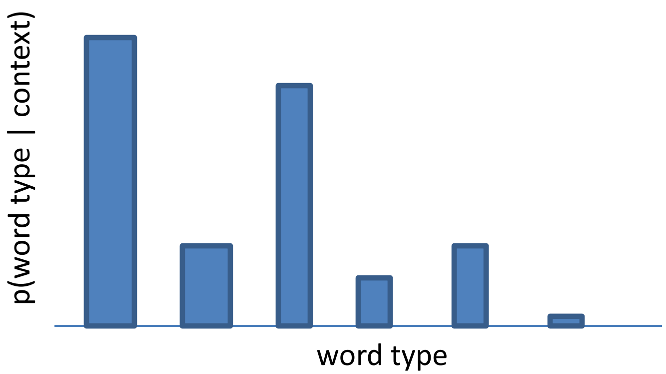
Intrinsic: Treat LM as its own downstream task

Use perplexity (from information theory)

Perplexity: Average "Surprisal"

Lower is better : lower perplexity ➔ less surprised





Less certain ➔
More surprised ➔
Higher perplexity

More certain ➔
Less surprised ➔
Lower perplexity

Perplexity

Lower is better : lower perplexity ➔ less surprised

$$\text{perplexity} = \exp(\text{avg xent})$$

Perplexity

Lower is better : lower perplexity ➔ less surprised

e.g., n-gram history (n-1 items)

$$\text{perplexity} = \exp\left(\frac{-1}{M} \sum_{i=1}^{M} \log p(w_i \mid h_i)\right)$$

Perplexity

Lower is better : lower perplexity ➔ less surprised

$$\text{perplexity} = \exp\left(\frac{-1}{M} \sum_{i=1}^{M} \log \underbrace{p(w_i \mid h_i)}_{\geq 0,\ \leq 1:\ \text{higher}}\right)$$

Perplexity

Lower is better : lower perplexity ➔ less surprised

$$\text{perplexity} = \exp\left(\frac{-1}{M}\sum_{i=1}^{M} \overbrace{\log \underbrace{p(w_i \mid h_i)}_{\geq 0,\ \leq 1:\ \text{higher}}}^{\leq 0:\ \text{higher}}\right)$$

Perplexity

Lower is better : lower perplexity ➔ less surprised

$$\text{perplexity} = \exp\Bigg(\frac{-1}{M}\underbrace{\sum_{i=1}^{M}\overbrace{\log \underbrace{p(w_i \mid h_i)}_{\geq 0,\ \leq 1\text{: higher}}}^{\leq 0\text{: higher}}}_{\leq 0\text{, higher}}\Bigg)$$

Perplexity

Lower is better : lower perplexity ➔ less surprised

$$\text{perplexity} = \exp\Big(\overbrace{\frac{-1}{M}\underbrace{\sum_{i=1}^{M}\overbrace{\log \underbrace{p(w_i \mid h_i)}_{\geq 0,\ \leq 1\text{: higher}}}^{\leq 0\text{: higher}}}_{\leq 0\text{, higher}}}^{\geq 0\text{, lower is better}}\Big)$$

Perplexity

Lower is better : lower perplexity ➔ less surprised

$$\text{perplexity} = \underbrace{\exp\Big(\overbrace{\frac{-1}{M}\underbrace{\sum_{i=1}^{M}\overbrace{\log\underbrace{p(w_i \mid h_i)}_{\geq 0,\ \leq 1:\ \text{higher}}}^{\leq 0:\ \text{higher}}}_{\leq 0,\ \text{higher}}}^{\geq 0,\ \text{lower is better}}\Big)}_{\geq 0,\ \text{lower}}$$

Perplexity

Lower is better : lower perplexity ➔ less surprised

base must be the same

$$\text{perplexity} = \exp\Big(\underbrace{\overbrace{\frac{-1}{M}\underbrace{\sum_{i=1}^{M}\overbrace{\log\underbrace{p(w_i \mid h_i)}_{\geq 0,\ \leq 1:\ \text{higher}}}^{\leq 0:\ \text{higher}}}_{\leq 0,\ \text{higher}}}^{\geq 0,\ \text{lower is better}}}_{\geq 0,\ \text{lower}}\Big)$$

Perplexity

Lower is better : lower perplexity ➔ less surprised

$$\text{perplexity} = \exp\left(\frac{-1}{M}\sum_{i=1}^{M}\log p(w_i \mid h_i)\right)$$

$$= \sqrt[M]{\underbrace{\prod_{i=1} \frac{1}{p(w_i \mid h_i)}}_{\text{weighted geometric average}}}$$

How to Compute Average Perplexity

- If you have a list of the probabilities for each observed n-gram "token:"

```
numpy.exp(-numpy.mean(numpy.log(probs_per_trigram_token)))
```

- If you have a list of observed n-gram "types" t and counts c, and log-prob. function lp:

```
numpy.exp(-numpy.mean(c*lp(t) for (t, c) in ngram_types.items()))
```

- If you're computing a cross-entropy loss function (e.g., in Pytorch):

```
loss_fn = torch.nn.CrossEntropyLoss(reduction='mean')
torch.exp(loss_fn(input_data))
```

What are the tri-grams for “The film , a hit !”

Trigrams	MLE p(trigram)
<BOS> <BOS> The	1
<BOS> The film	1
The film ,	0
film , a	0
, a hit	0
a hit !	0
hit ! <EOS>	0
Perplexity	???

What are the tri-grams for “The film , a hit !”

Trigrams	MLE p(trigram)
<BOS> <BOS> The	1
<BOS> The film	1
The film ,	0
film , a	0
, a hit	0
a hit !	0
hit ! <EOS>	0
Perplexity	Infinity

What are the tri-grams for "The film , a hit !"

Trigrams	MLE p(trigram)	Smoothed p(trigram)
<BOS> <BOS> The	1	2/17
<BOS> The film	1	2/17
The film ,	0	1/17
film , a	0	1/16
, a hit	0	1/16
a hit !	0	1/17
hit ! <EOS>	0	1/16
Perplexity	Infinity	???

What are the tri-grams for "The film , a hit !"

Trigrams	MLE p(trigram)	Smoothed p(trigram)
<BOS> <BOS> The	1	2/17
<BOS> The film	1	2/17
The film ,	0	1/17
film , a	0	1/16
, a hit	0	1/16
a hit !	0	1/17
hit ! <EOS>	0	1/16
Perplexity	Infinity	13.59

0s Are Not Your (Language Model's) Friend

$$p(\text{item}) \propto count(\text{item}) = 0 \rightarrow$$
$$p(\text{item}) = 0$$

0 probability → item is *impossible*

0s annihilate: x*y*z*0 = 0

Language is creative:

new words keep appearing

existing words could appear in known contexts

How much do you trust your data?

Language Models & Smoothing

- ~~Maximum likelihood (MLE): simple counting~~
- Other count-based models
  - **Laplace smoothing, add- λ**
  - Interpolation models
  - Discounted backoff
  - Interpolated (modified) Kneser-Ney
  - Good-Turing
  - Witten-Bell
- Maxent n-gram models
- Neural n-gram models
- Recurrent/autoregressive NNs

Easy to implement

Advanced/ out of scope

Featureful LMs

Feedforward LMs

Super modern

Add-λ estimation

Other names: Laplace smoothing, Lidstone smoothing

Pretend we saw each word λ more times than we did

$$p(\mathrm{z}) \propto count(\mathrm{z}) + \lambda$$

Add λ to all the counts

Add-λ estimation

Other names: Laplace smoothing, Lidstone smoothing

Pretend we saw each word λ more times than we did

Add λ to all the counts

$$p(\mathrm{z}) \propto count(\mathrm{z}) + \lambda$$

$$= \frac{count(\mathrm{z}) + \lambda}{\sum_v (count(\mathrm{v}) + \lambda)}$$

Add-λ estimation

Other names: Laplace smoothing, Lidstone smoothing

Pretend we saw each word λ more times than we did

Add λ to all the counts

$$p(\mathrm{z}) \propto count(\mathrm{z}) + \lambda$$

$$= \frac{count(\mathrm{z}) + \lambda}{W + V\lambda}$$

tokens

types

Add-λ N-Grams (Unigrams)

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

Word (Type)	Raw Count	Norm	Prob.	Add-λ Count	Add-λ Norm.	Add-λ Prob.
The	1		1/16			
film	2		1/8			
got	1		1/16			
a	2		1/8			
great	1		1/16			
opening	1		1/16			
and	1		1/16			
the	1	16	1/16			
went	1		1/16			
on	1		1/16			
to	1		1/16			
become	1		1/16			
hit	1		1/16			
.	1		1/16			

Add-1 N-Grams (Unigrams)

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

Word (Type)	Raw Count	Norm	Prob.	Add-1 Count	Add-1 Norm.	Add-1 Prob.
The	1	16	1/16	2		
film	2		1/8	3		
got	1		1/16	2		
a	2		1/8	3		
great	1		1/16	2		
opening	1		1/16	2		
and	1		1/16	2		
the	1		1/16	2		
went	1		1/16	2		
on	1		1/16	2		
to	1		1/16	2		
become	1		1/16	2		
hit	1		1/16	2		
.	1		1/16	2		

Add-1 N-Grams (Unigrams)

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

Word (Type)	Raw Count	Norm	Prob.	Add-1 Count	Add-1 Norm.	Add-1 Prob.
The	1	16	1/16	2	16 + 14*1 = 30	
film	2		1/8	3		
got	1		1/16	2		
a	2		1/8	3		
great	1		1/16	2		
opening	1		1/16	2		
and	1		1/16	2		
the	1		1/16	2		
went	1		1/16	2		
on	1		1/16	2		
to	1		1/16	2		
become	1		1/16	2		
hit	1		1/16	2		
.	1		1/16	2		

Add-1 N-Grams (Unigrams)

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

Word (Type)	Raw Count	Norm	Prob.	Add-1 Count	Add-1 Norm.	Add-1 Prob.
The	1	16	1/16	2	16 + 14*1 = 30	=1/15
film	2		1/8	3		=1/10
got	1		1/16	2		=1/15
a	2		1/8	3		=1/10
great	1		1/16	2		=1/15
opening	1		1/16	2		=1/15
and	1		1/16	2		=1/15
the	1		1/16	2		=1/15
went	1		1/16	2		=1/15
on	1		1/16	2		=1/15
to	1		1/16	2		=1/15
become	1		1/16	2		=1/15
hit	1		1/16	2		=1/15
.	1		1/16	2		=1/15

An Extended Trigram Example

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

Q: With OOV, EOS, and BOS, how many types (for normalization)?

Context: x y	Word (Type): z	Raw Count	Add-1 count	Norm.	Probability p(z | x y)
The film	The	0			
The film	film	0			
The film	got	1			
The film	went	0			
The film	OOV	0			
The film	EOS	0			
		…			
a great	great	0			
a great	opening	1			
a great	and	0			
a great	the	0			
		…			

An Extended Trigram Example

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

Q: With OOV, EOS, and BOS, how many types (for normalization)?

A: 16
(why don't we count BOS?)

Context: x y	Word (Type): z	Raw Count	Add-1 count	Norm.	Probability p(z | x y)
The film	The	0			
The film	film	0			
The film	got	1			
The film	went	0			
The film	OOV	0			
The film	EOS	0			
		…			
a great	great	0			
a great	opening	1			
a great	and	0			
a great	the	0			
		…			

An Extended Trigram Example

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

Q: With OOV, EOS, and BOS, how many types (for normalization)?

A: 16
(why don't we count BOS?)

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
…					

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
		…			

Q: What is the perplexity for the sentence "The film , a hit !"

What are the tri-grams for “The film , a hit !”

Trigrams	MLE p(trigram)
<BOS> <BOS> The	1
<BOS> The film	1
The film ,	0
film , a	0
, a hit	0
a hit !	0
hit ! <EOS>	0

What are the tri-grams for “The film , a hit !”

Trigrams	MLE p(trigram)
<BOS> <BOS> The	1
<BOS> The film	1
The film ,	0
film , a	0
, a hit	0
a hit !	0
hit ! <EOS>	0

What are the tri-grams for "The film , a hit !"

Trigrams	MLE p(trigram)	UNK-ed trigrams
<BOS> <BOS> The	1	<BOS> <BOS> The
<BOS> The film	1	<BOS> The film
The film ,	0	The film <UNK>
film , a	0	film <UNK> a
, a hit	0	<UNK> a hit
a hit !	0	a hit <UNK>
hit ! <EOS>	0	hit <UNK> <EOS>

What are the tri-grams for “The film , a hit !”

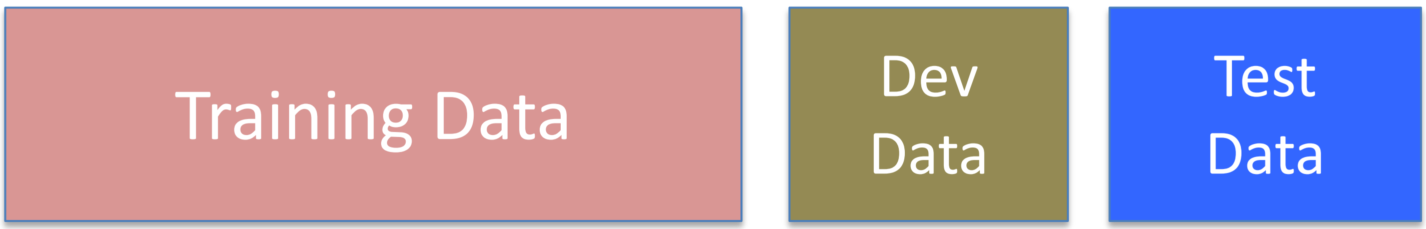
Trigrams	MLE p(trigram)	UNK-ed trigrams	Smoothed p(trigram)
<BOS> <BOS> The	1	<BOS> <BOS> The	2/17
<BOS> The film	1	<BOS> The film	2/17
The film ,	0	The film <UNK>	1/17
film , a	0	film <UNK> a	1/16
, a hit	0	<UNK> a hit	1/16
a hit !	0	a hit <UNK>	1/17
hit ! <EOS>	0	hit <UNK> <EOS>	1/16

What are the tri-grams for “The film , a hit !”

Trigrams	MLE p(trigram)	UNK-ed trigrams	Smoothed p(trigram)
<BOS> <BOS> The	1	<BOS> <BOS> The	2/17
<BOS> The film	1	<BOS> The film	2/17
The film ,	0	The film <UNK>	1/17
film , a	0	film <UNK> a	1/16
, a hit	0	<UNK> a hit	1/16
a hit !	0	a hit <UNK>	1/17
hit ! <EOS>	0	hit <UNK> <EOS>	1/16

Setting Hyperparameters

Use a **development** corpus



Choose λs to maximize the probability of dev data:

- Fix the N-gram probabilities (on the training data)
- Then search for λs that give largest probability to held-out set:

Advanced topic

Other Kinds of Smoothing

- ~~Maximum likelihood (MLE): simple counting~~
- Laplace smoothing, add- λ
- Interpolation models
- Discounted backoff
- Interpolated (modified) Kneser-Ney
- Good-Turing
- Witten-Bell

Interpolated (modified) Kneser-Ney

Idea: How "productive" is a context?

How many different word *types v* appear in a context *x, y*

Good-Turing

Partition words into classes of occurrence

Smooth class statistics

Properties of classes are likely to predict properties of other classes

Witten-Bell

Idea: Every observed type was at some point novel

Give MLE prediction for novel *type* occurring

Language Models & Smoothing

- ~~Maximum likelihood (MLE): simple counting~~
- ~~Other count-based models~~
  - ~~Laplace smoothing, add-λ~~
  - ~~Interpolation models~~
  - ~~Discounted backoff~~
  - ~~Interpolated (modified) Kneser-Ney~~
  - ~~Good-Turing~~
  - ~~Witten-Bell~~
- **Maxent n-gram models**
- Neural n-gram models
- Recurrent/autoregressive NNs

Easy to implement

Advanced/ out of scope

Featureful LMs

Feedforward LMs

Super modern

Maxent Models as Featureful n-gram Language Models

p(Colorless green ideas sleep furiously | Label) =
p(Colorless | Label, <BOS>) * … * p(<EOS> | Label , furiously)

Model each n-gram term with a maxent model

$$p(x_i \mid y, x_{i-N+1:i-1}) = \text{maxent}(y, x_{i-N+1:i-1}, x_i)$$

generatively trained:
learn to model *(class-specific) language*

Language Model with Maxent n-grams

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

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

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

What Should These Features Do?

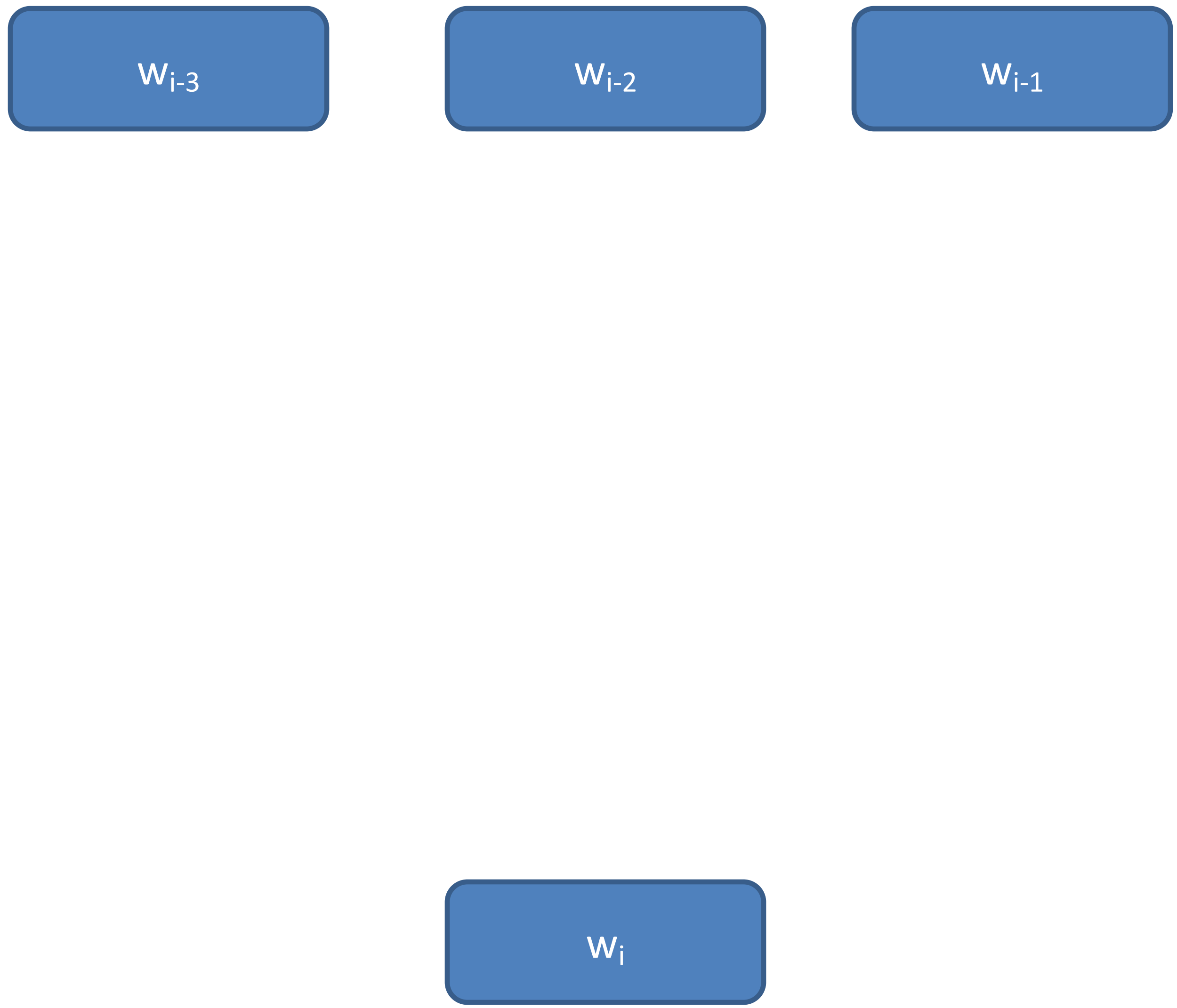
$p(x_i \mid y, x_{i-N+1:i-1}) = \text{maxent}(y, x_{i-N+1:i-1}, x_i)$, e.g.,

$$p(\text{sleep} \mid y, \text{green}, \text{ideas}) = \\ \text{maxent}\left(y, x_{i-2,i-1} = (\text{green}, \text{ideas}), x_i = \text{sleep}\right) \\ \propto \exp(\theta_{x_i=\text{sleep}}{}^T f(y, x_{i-2,i-1} = (\text{green}, \text{ideas})))$$

(in-class discussion)

N-gram Language Models

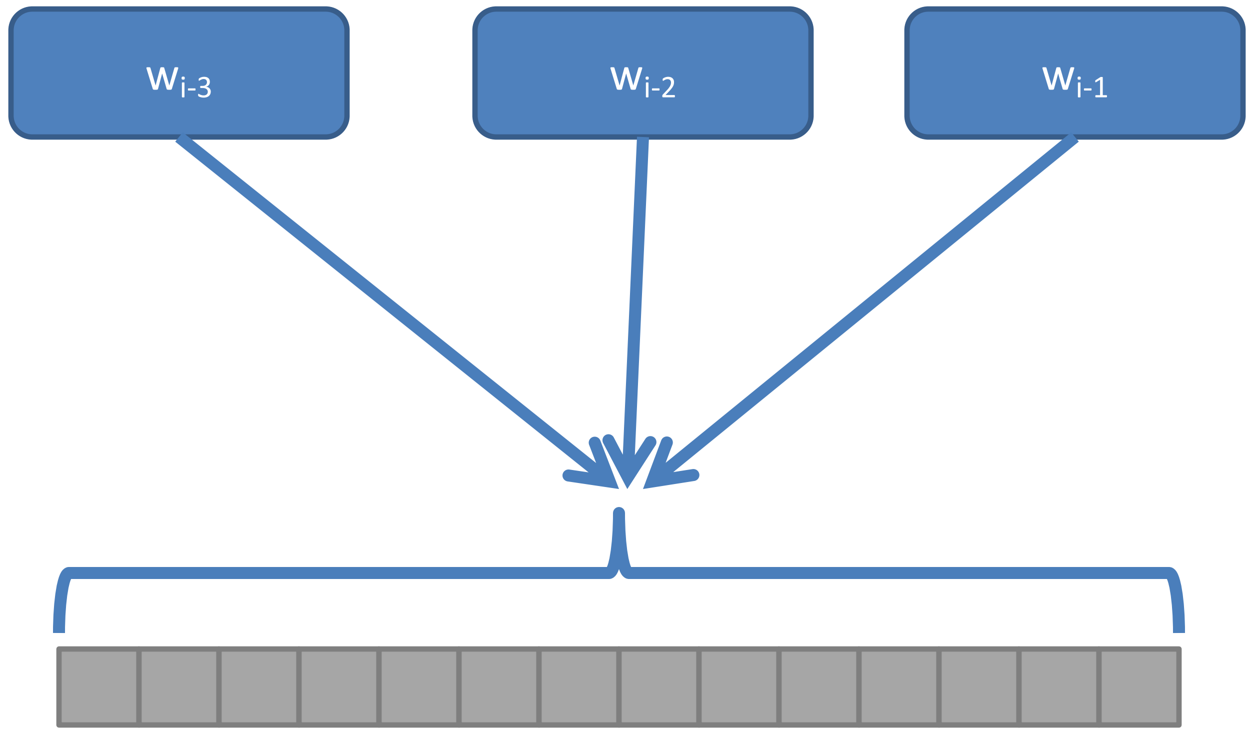
given some context…



predict the next word

N-gram Language Models

given some context…



compute beliefs about what is likely…

$$p(w_i | \, w_{i-3}, w_{i-2}, w_{i-1}) \propto count(w_{i-3}, w_{i-2}, w_{i-1}, w_i)$$

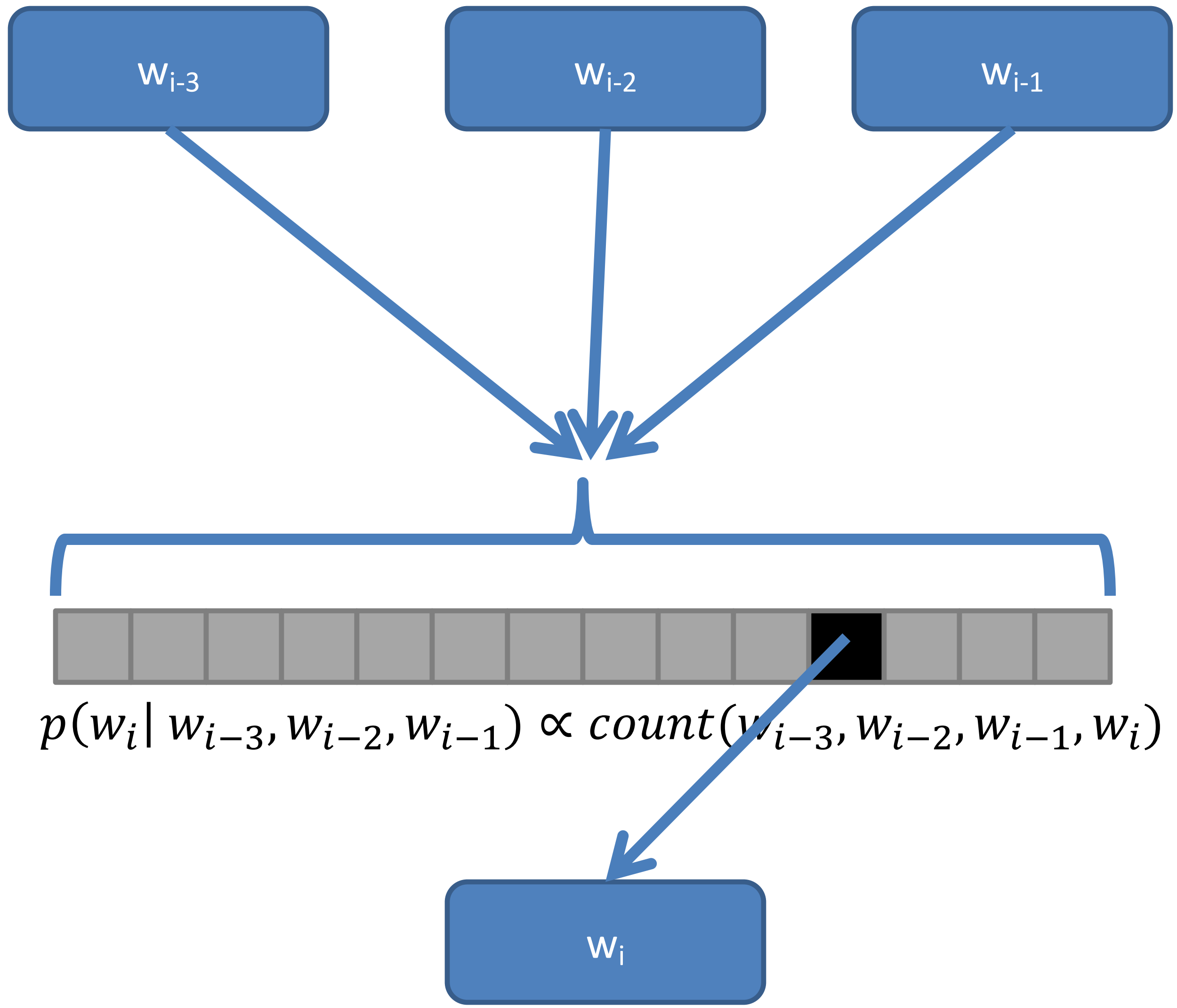
predict the next word

N-gram Language Models

given some context…

compute beliefs about what is likely…

predict the next word

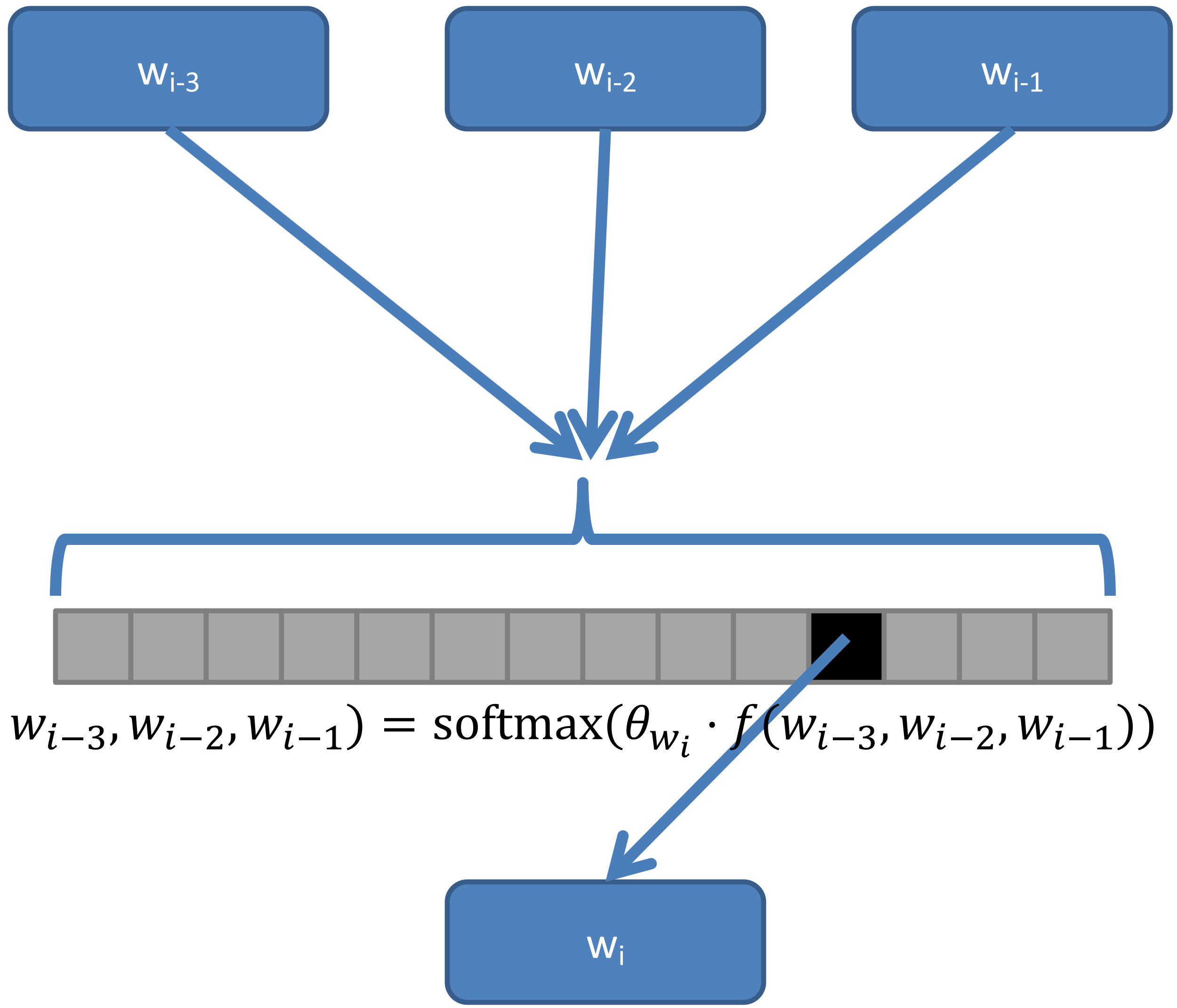


Maxent Language Models

given some context…

compute beliefs about what is likely…

predict the next word



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

A Closer Look at Maxent $p($ Won't you please donate? $|$ Primary $)$

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

To learn $p($ Won't you please donate? $|$ Class $)$:

Define features f that make use of the specific label Class

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

Language Models & Smoothing

- ~~Maximum likelihood (MLE): simple counting~~
- ~~Other count-based models~~
  - ~~Laplace smoothing, add-λ~~ ← Easy to implement
  - ~~Interpolation models~~
  - ~~Discounted backoff~~
  - ~~Interpolated (modified) Kneser-Ney~~
  - ~~Good-Turing~~
  - ~~Witten-Bell~~

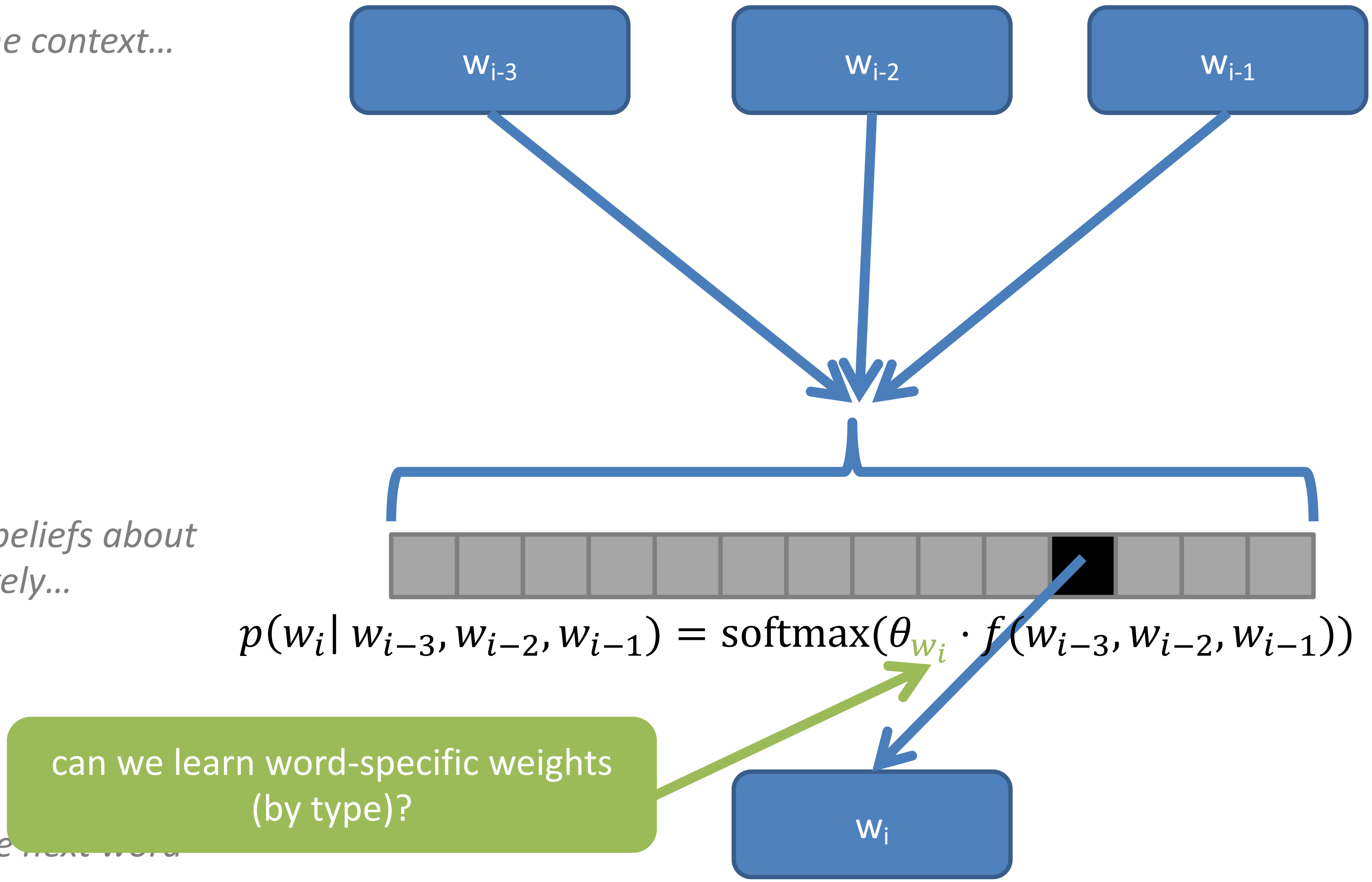
  Advanced/ out of scope
- ~~Maxent n-gram models~~ ← Featureful LMs
- **Neural n-gram models** ← Feedforward LMs
- Recurrent/autoregressive NNs ← Super modern

Maxent Language Models

given some context…

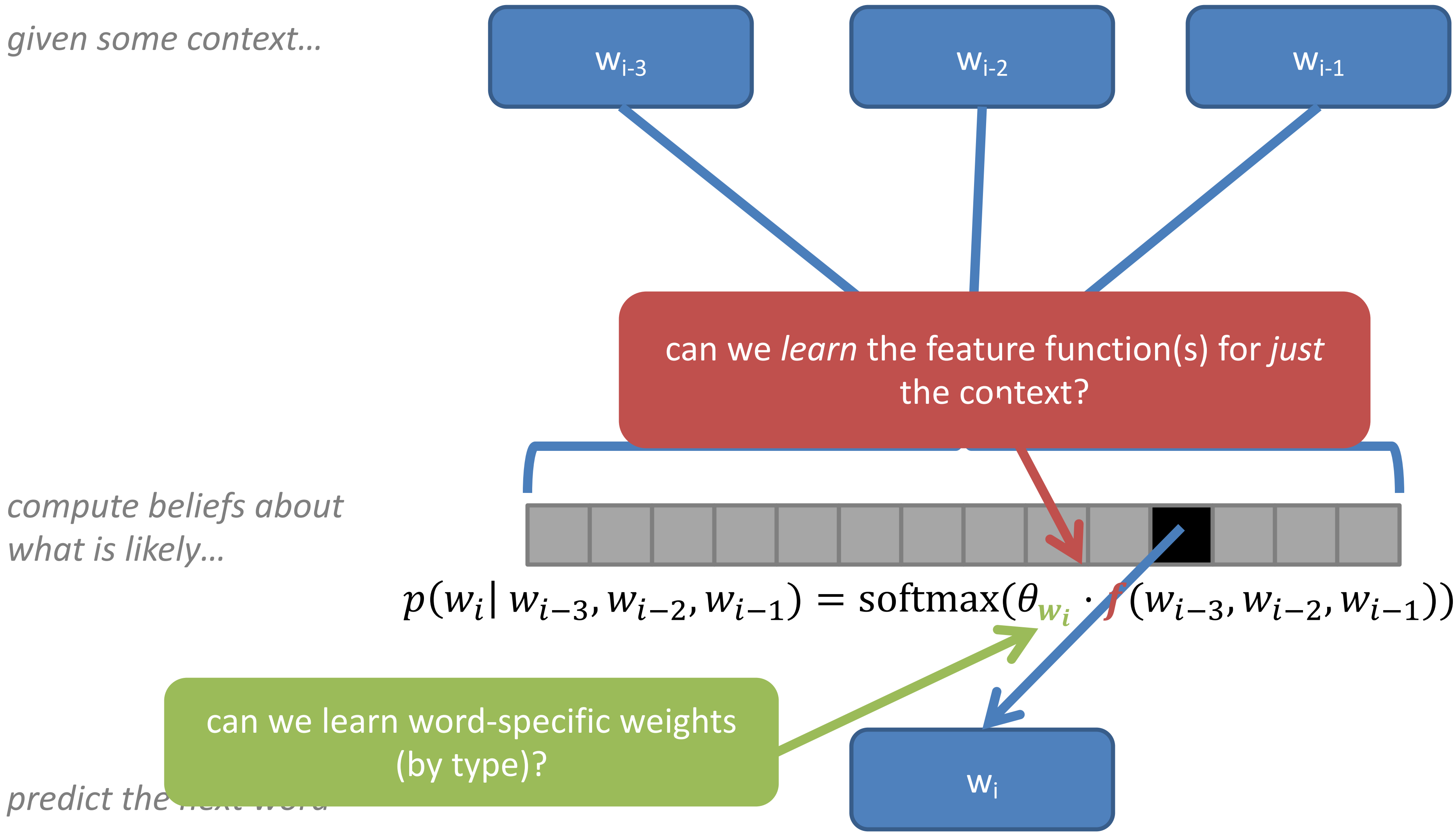
compute beliefs about what is likely…

predict the next word



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

Neural Language Models

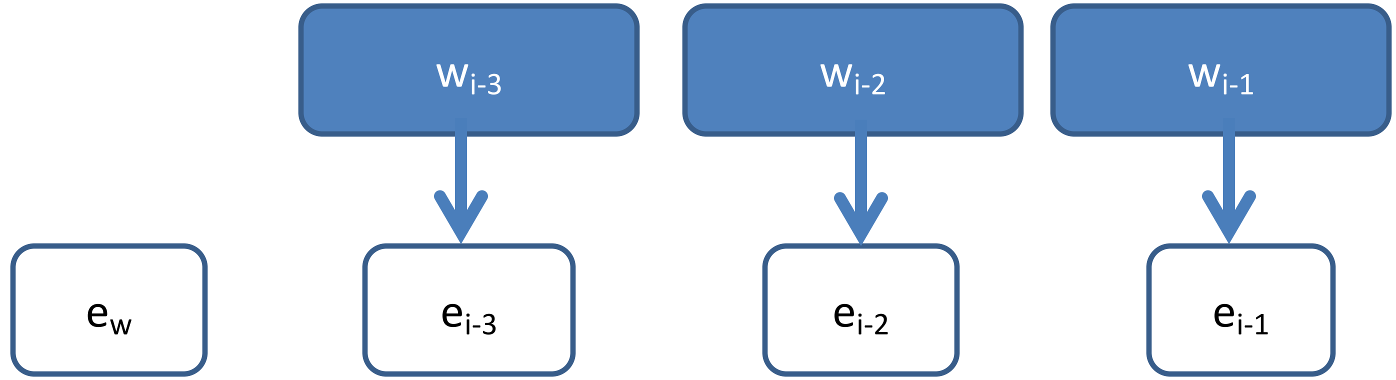


$$p(w_i \mid w_{i-3}, w_{i-2}, w_{i-1}) = \text{softmax}(\theta_{w_i} \cdot f(w_{i-3}, w_{i-2}, w_{i-1}))$$

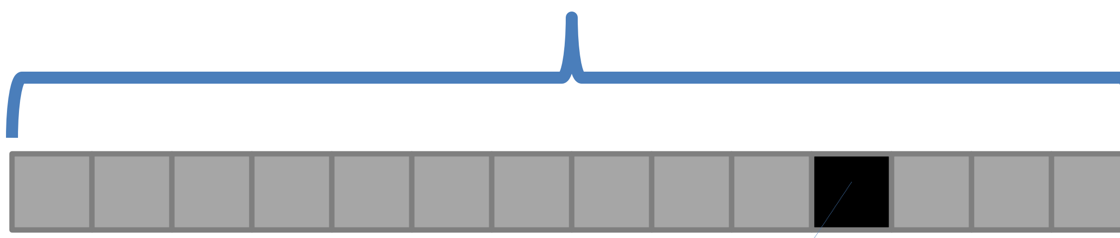
Neural Language Models

given some context…

create/use "distributed representations"…

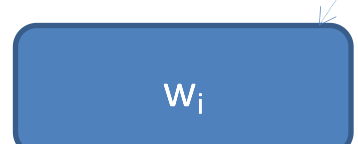


compute beliefs about what is likely…



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

predict the next word



Neural Language Models

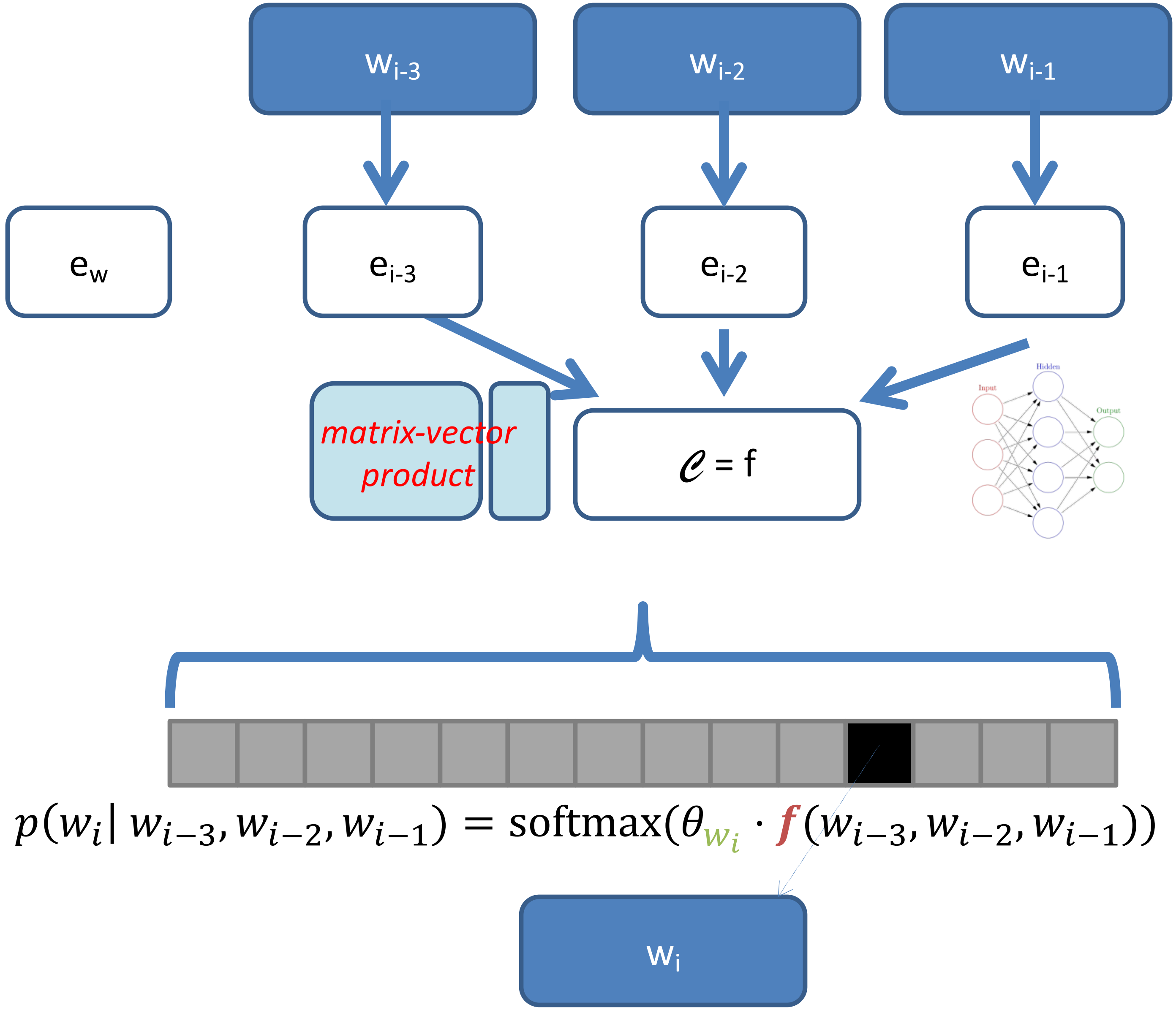
given some context…

create/use “distributed representations”…

combine these representations…

compute beliefs about what is likely…

predict the next word



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

Neural Language Models

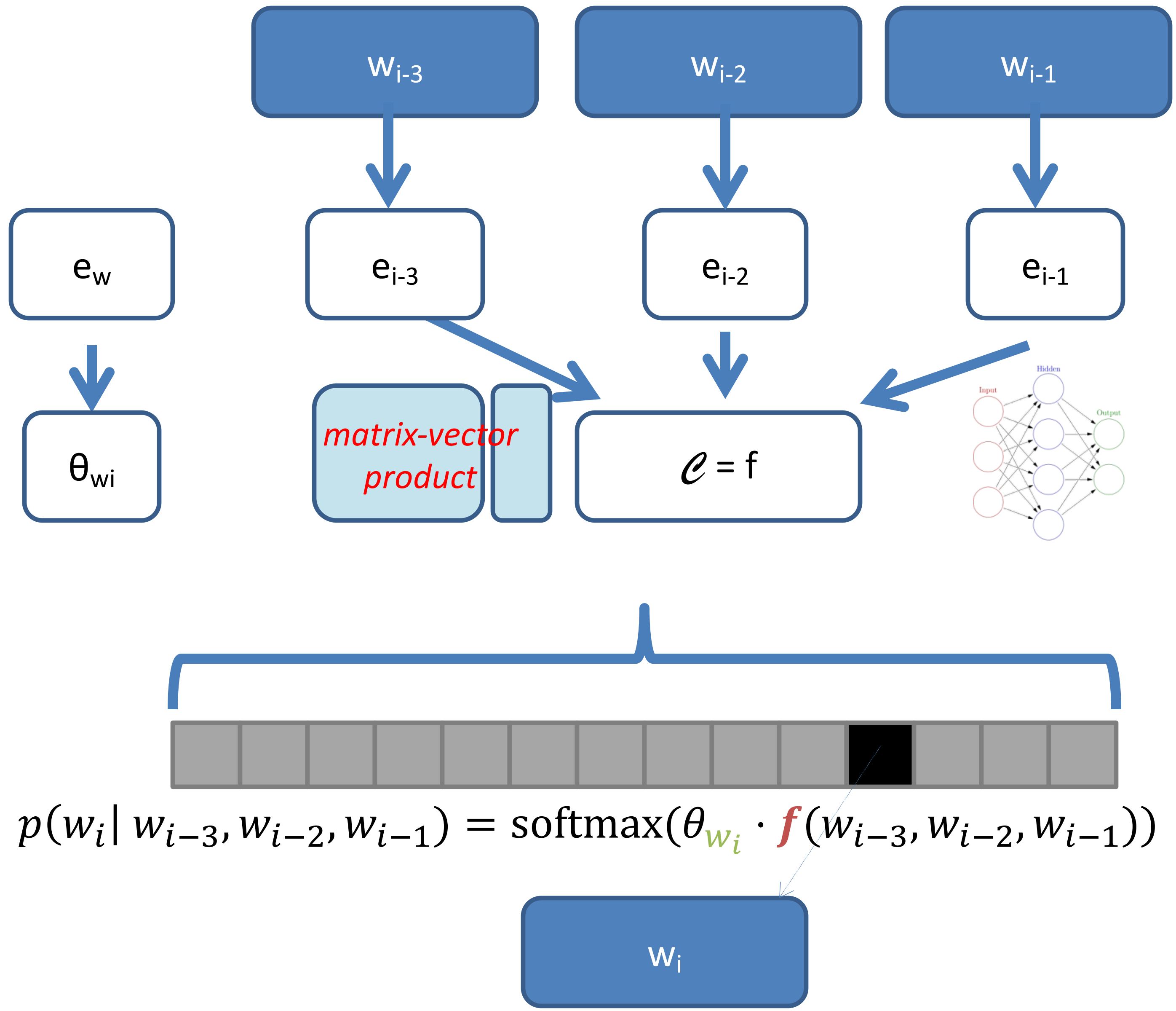
given some context…

create/use “distributed representations”…

combine these representations…

compute beliefs about what is likely…

predict the next word



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

Neural Language Models

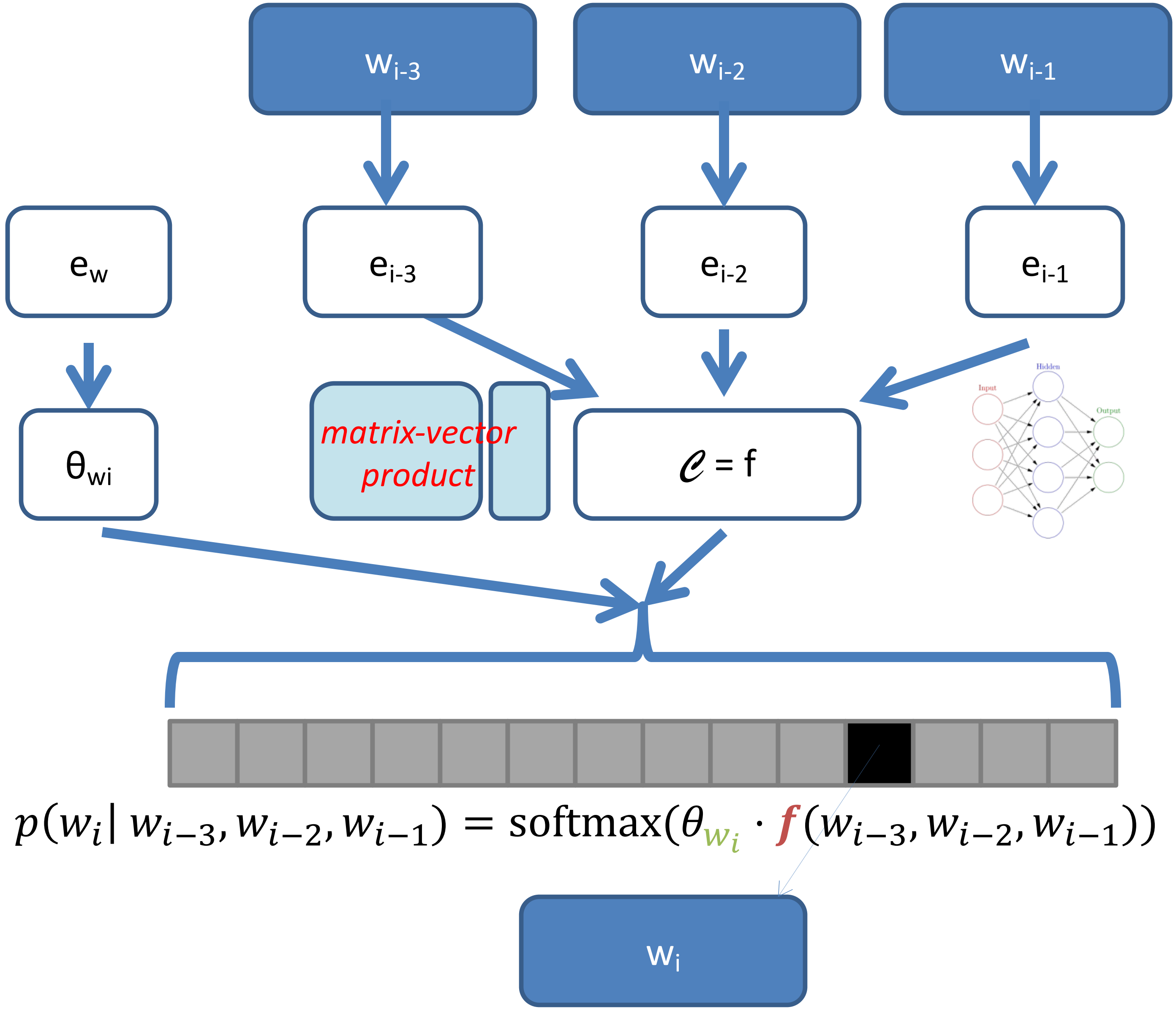
given some context…

create/use "distributed representations"…

combine these representations…

compute beliefs about what is likely…

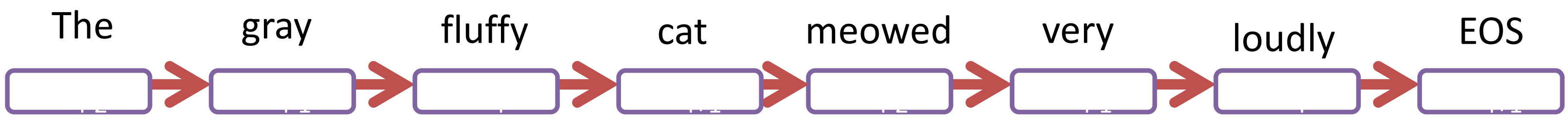
predict the next word



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

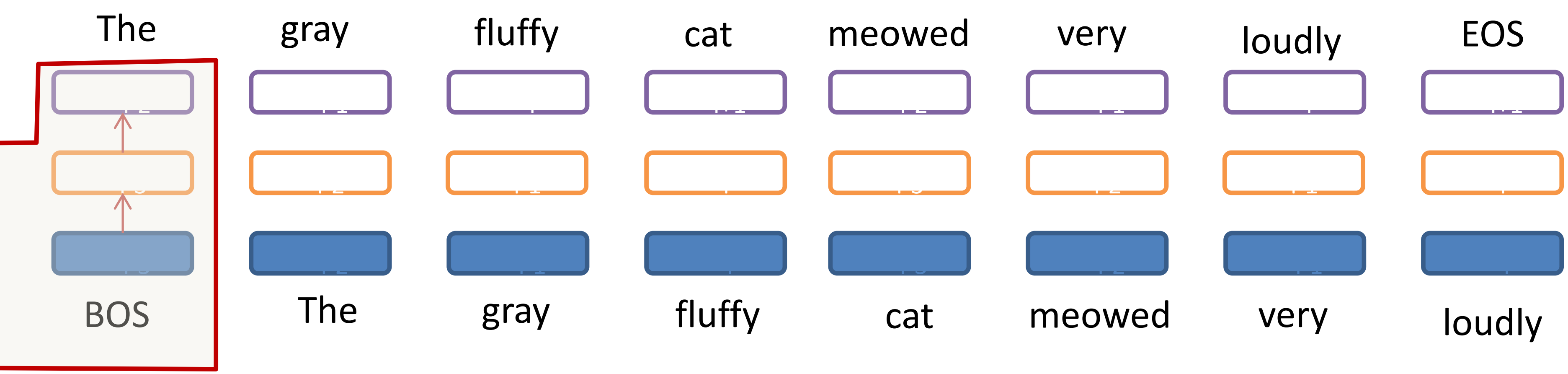
A Neural N-Gram Model

The gray fluffy cat meowed very loudly



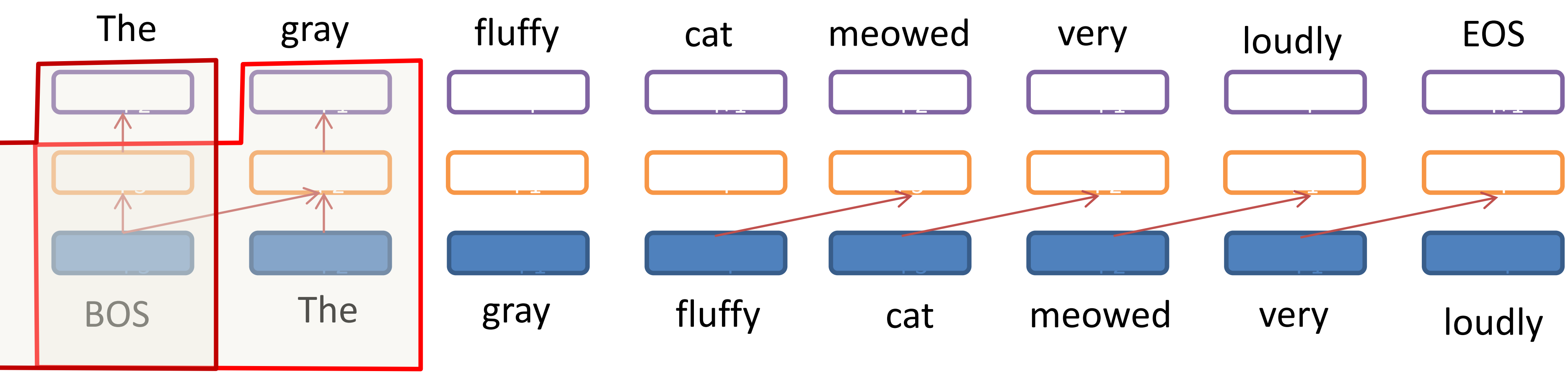
A Neural N-Gram Model (N=3)

The gray fluffy cat meowed very loudly



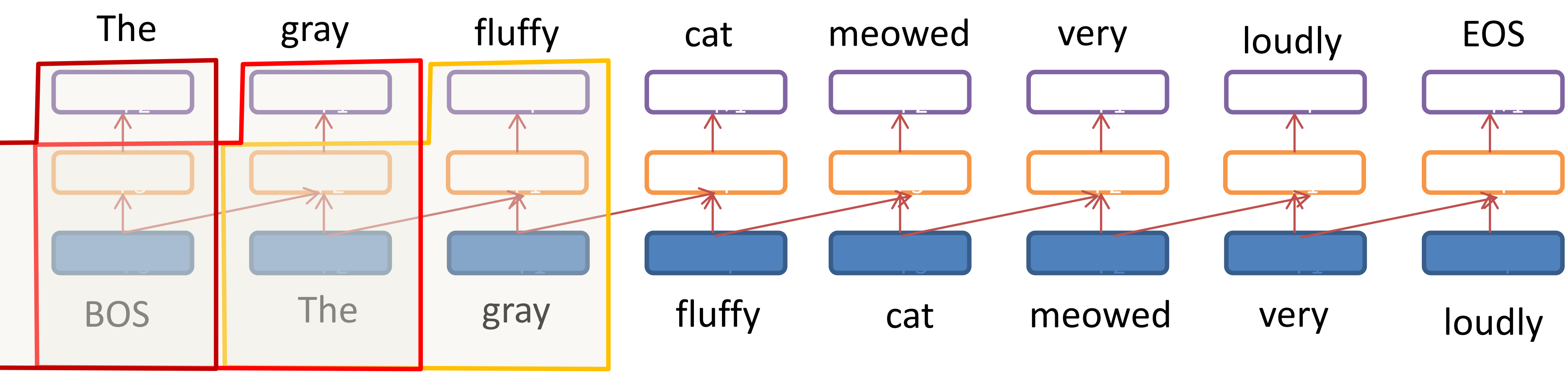
A Neural N-Gram Model (N=3)

The gray fluffy cat meowed very loudly



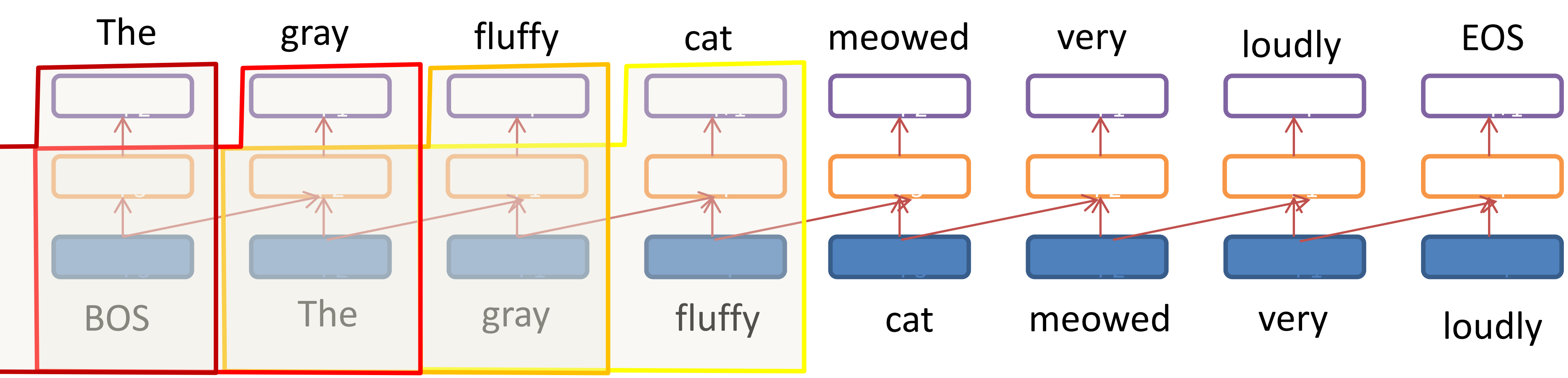
A Neural N-Gram Model (N=3)

The gray fluffy cat meowed very loudly



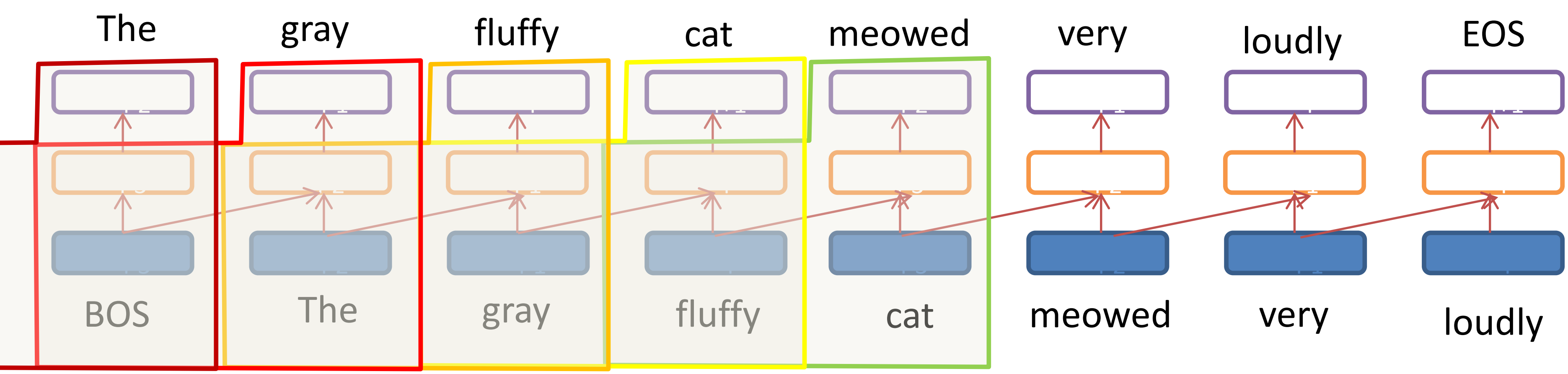
A Neural N-Gram Model (N=3)

The gray fluffy cat meowed very loudly



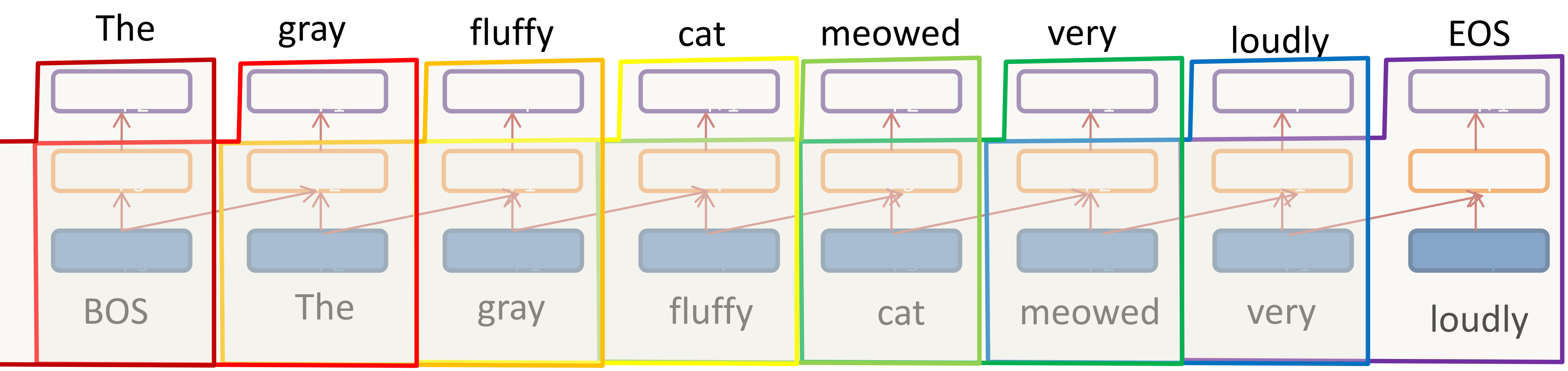
A Neural N-Gram Model (N=3)

The gray fluffy cat meowed very loudly



A Neural N-Gram Model (N=3)

The gray fluffy cat meowed very loudly



“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

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

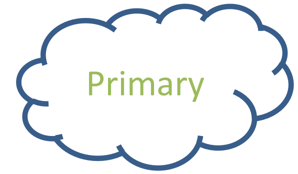
Baselines

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

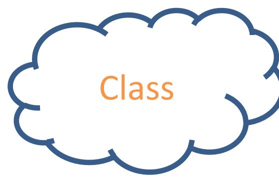
NPLM

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 $p($  $|$  $)$

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

To learn $p($  $|$  $)$:

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

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

Language Models & Smoothing

- ~~Maximum likelihood (MLE): simple counting~~
- ~~Other count-based models~~
  - ~~Laplace smoothing, add-λ~~ — Easy to implement
  - ~~Interpolation models~~
  - ~~Discounted backoff~~
  - ~~Interpolated (modified) Kneser-Ney~~
  - ~~Good-Turing~~
  - ~~Witten-Bell~~

  Advanced/ out of scope
- ~~Maxent n-gram models~~ — Featureful LMs
- ~~Neural n-gram models~~ — Feedforward LMs
- **Recurrent/autoregressive NNs** — Super modern

Recurrent/Autoregressive LMs

- coming next class…