

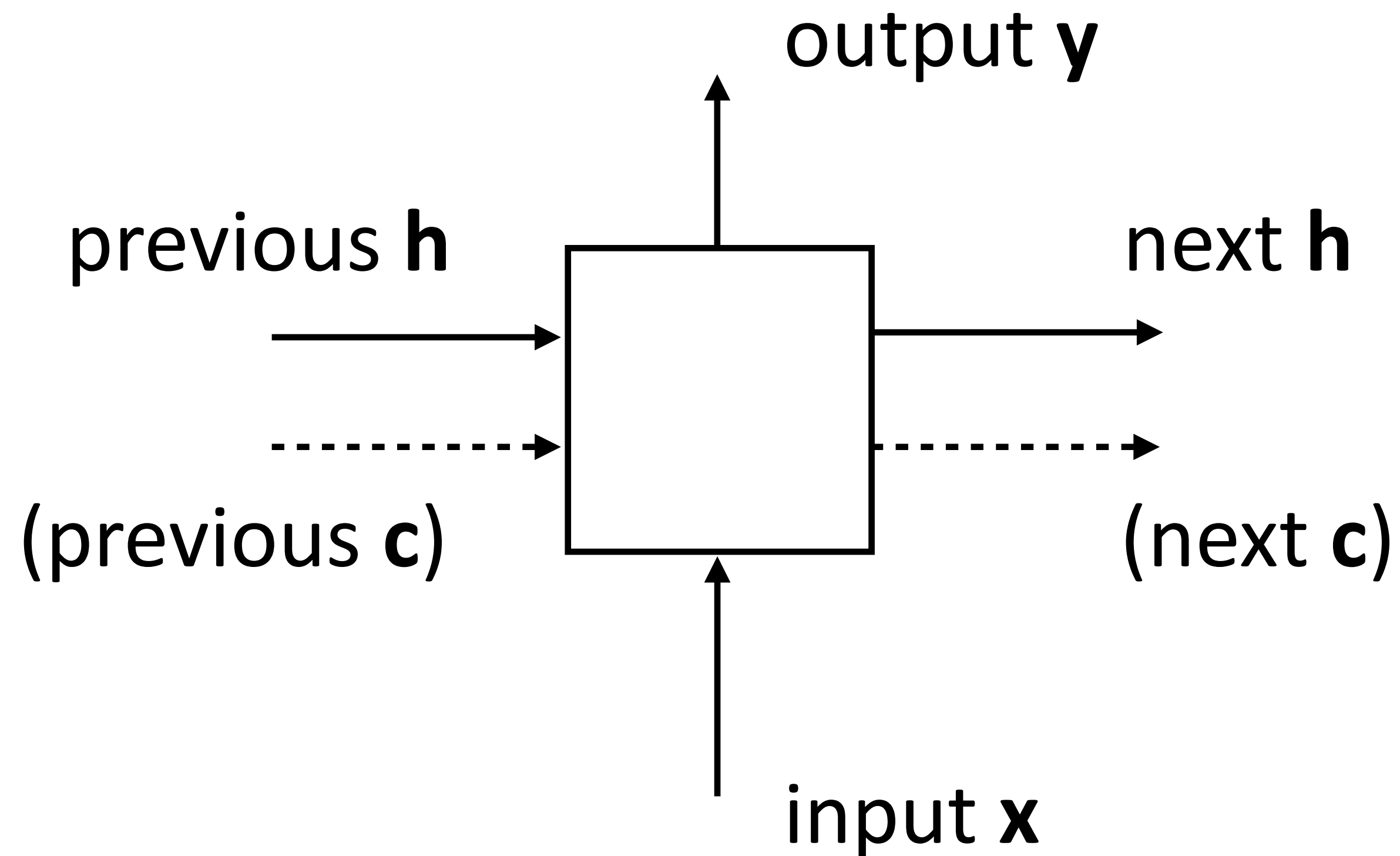
Lecture 9: CNNs, Neural CRFs

Alan Ritter

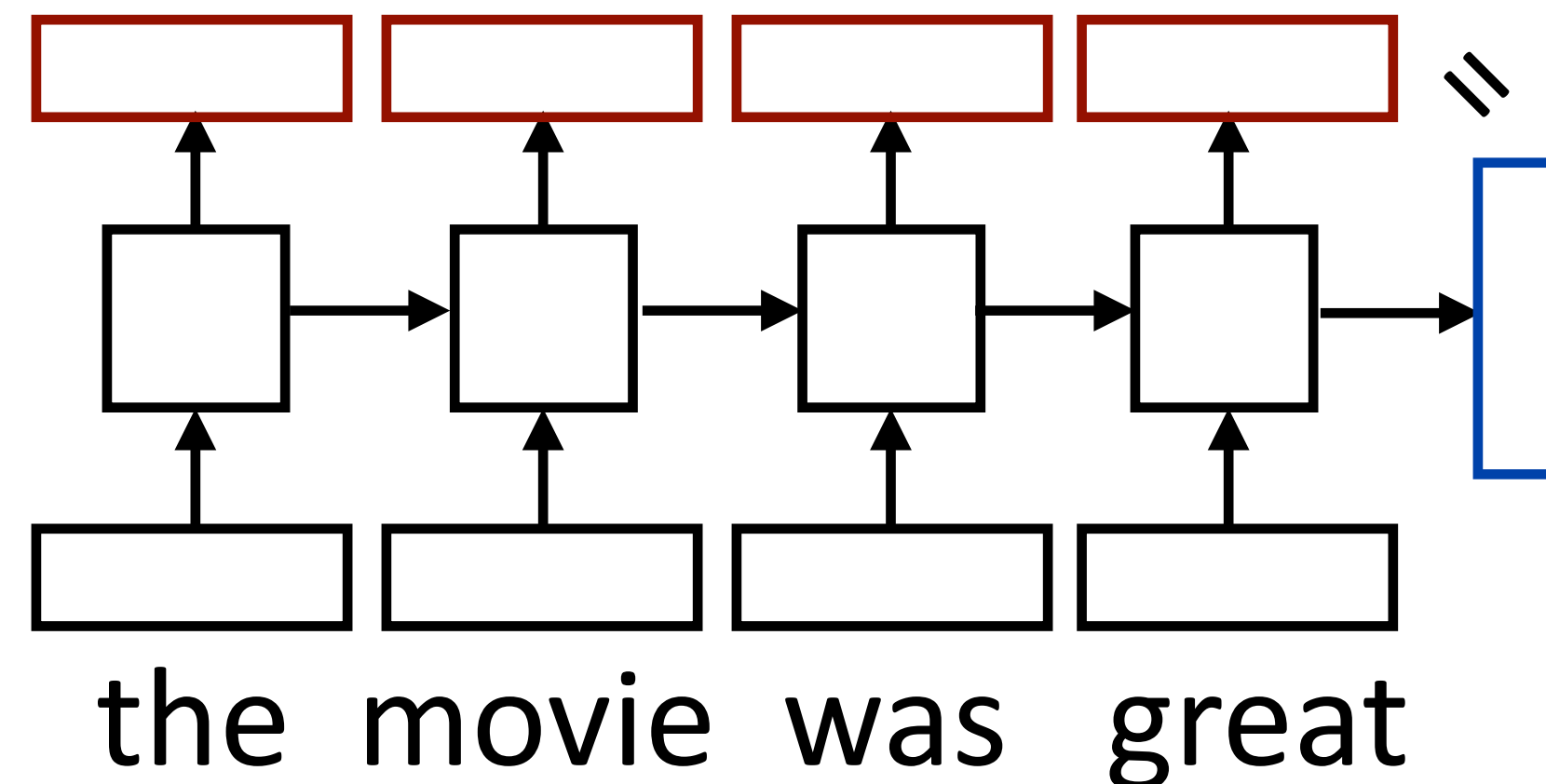
(many slides from Greg Durrett)

Recall: RNNs

- ▶ Cell that takes some input \mathbf{x} , has some hidden state \mathbf{h} , and updates that hidden state and produces output \mathbf{y} (all vector-valued)



Recall: RNN Abstraction



- ▶ **Encoding of the sentence** — can pass this a decoder or make a classification decision about the sentence
- ▶ **Encoding of each word** — can pass this to another layer to make a prediction (can also pool these to get a different sentence encoding)
- ▶ RNN can be viewed as a transformation of a sequence of vectors into a sequence of context-dependent vectors

What can LSTMs model?

- ▶ Sentiment
 - ▶ Encode one sentence, predict
- ▶ Language models
 - ▶ Move left-to-right, per-token prediction
- ▶ Translation
 - ▶ Encode sentence + then decode, use token predictions for attention weights (next lecture)

What can LSTMs model?

- ▶ Sentiment
 - ▶ Encode one sentence, predict
- ▶ Language models
 - ▶ Move left-to-right, per-token prediction
- ▶ Translation
 - ▶ Encode sentence + then decode, use token predictions for attention weights (next lecture)
- ▶ Textual entailment

What can LSTMs model?

- ▶ Sentiment
 - ▶ Encode one sentence, predict
- ▶ Language models
 - ▶ Move left-to-right, per-token prediction
- ▶ Translation
 - ▶ Encode sentence + then decode, use token predictions for attention weights (next lecture)
- ▶ Textual entailment
 - ▶ Encode two sentences, predict

Natural Language Inference

Premise

A boy plays in the snow

Hypothesis

A boy is outside

Natural Language Inference

Premise

A boy plays in the snow

entails

Hypothesis

A boy is outside

Natural Language Inference

Premise

Hypothesis

A boy plays in the snow

entails

A boy is outside

A man inspects the uniform of a figure

The man is sleeping

Natural Language Inference

Premise

Hypothesis

A boy plays in the snow

entails

A boy is outside

A man inspects the uniform of a figure

contradicts

The man is sleeping

Natural Language Inference

Premise

Hypothesis

A boy plays in the snow

entails

A boy is outside

A man inspects the uniform of a figure

contradicts

The man is sleeping

An older and younger man smiling

Two men are smiling and
laughing at cats playing

Natural Language Inference

Premise

Hypothesis

A boy plays in the snow

entails

A boy is outside

A man inspects the uniform of a figure

contradicts

The man is sleeping

An older and younger man smiling

neutral

Two men are smiling and laughing at cats playing

Natural Language Inference

Premise

Hypothesis

A boy plays in the snow

entails

A boy is outside

A man inspects the uniform of a figure

contradicts

The man is sleeping

An older and younger man smiling

neutral

Two men are smiling and laughing at cats playing

- ▶ Long history of this task: “Recognizing Textual Entailment” challenge in 2006 (Dagan, Glickman, Magnini)

Natural Language Inference

Premise

Hypothesis

A boy plays in the snow

entails

A boy is outside

A man inspects the uniform of a figure

contradicts

The man is sleeping

An older and younger man smiling

neutral

Two men are smiling and laughing at cats playing

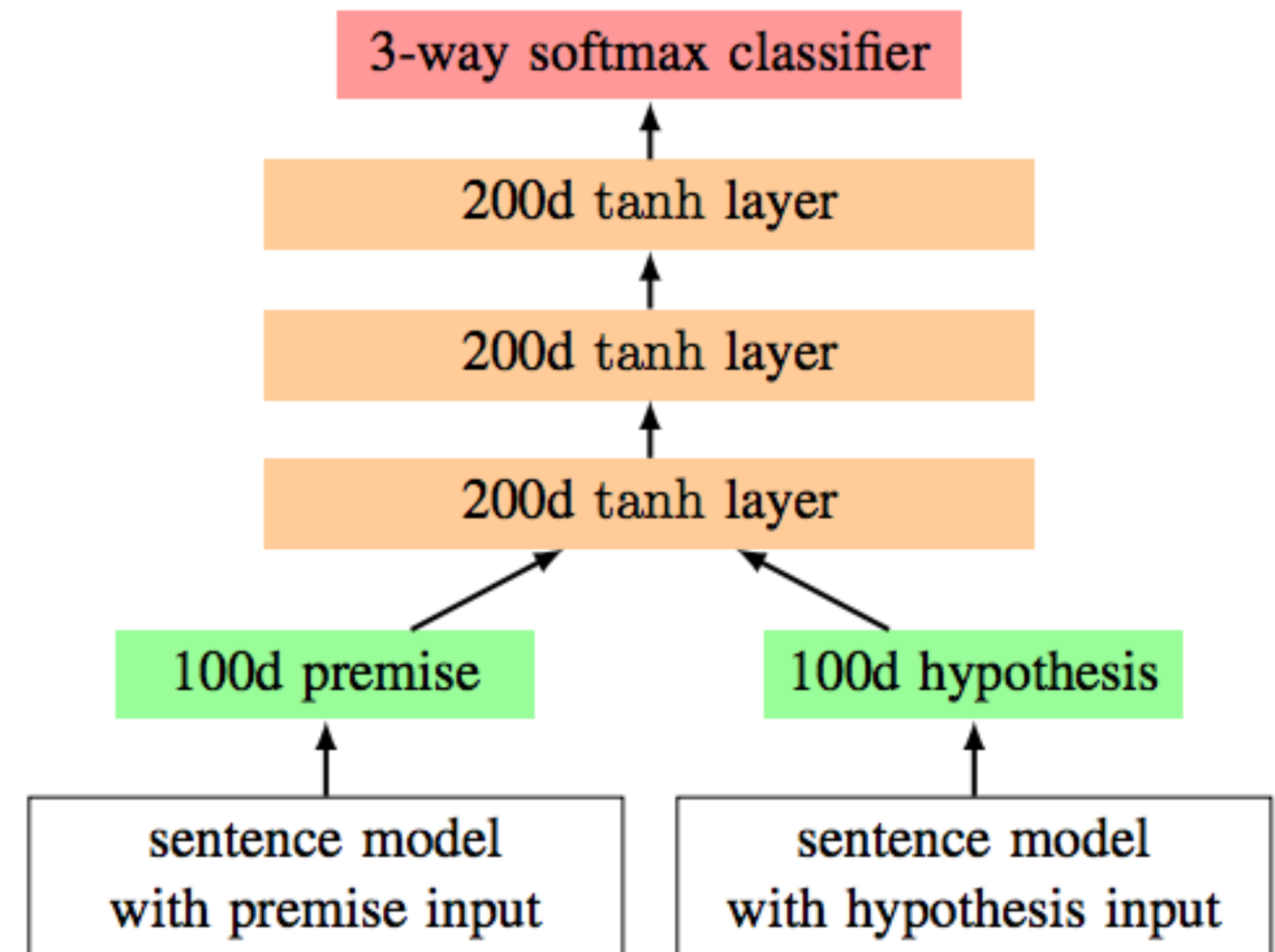
- ▶ Long history of this task: “Recognizing Textual Entailment” challenge in 2006 (Dagan, Glickman, Magnini)
- ▶ Early datasets: small (hundreds of pairs), very ambitious (lots of world knowledge, temporal reasoning, etc.)

SNLI Dataset

- ▶ Show people captions for (unseen) images and solicit entailed / neural / contradictory statements
- ▶ >500,000 sentence pairs

SNLI Dataset

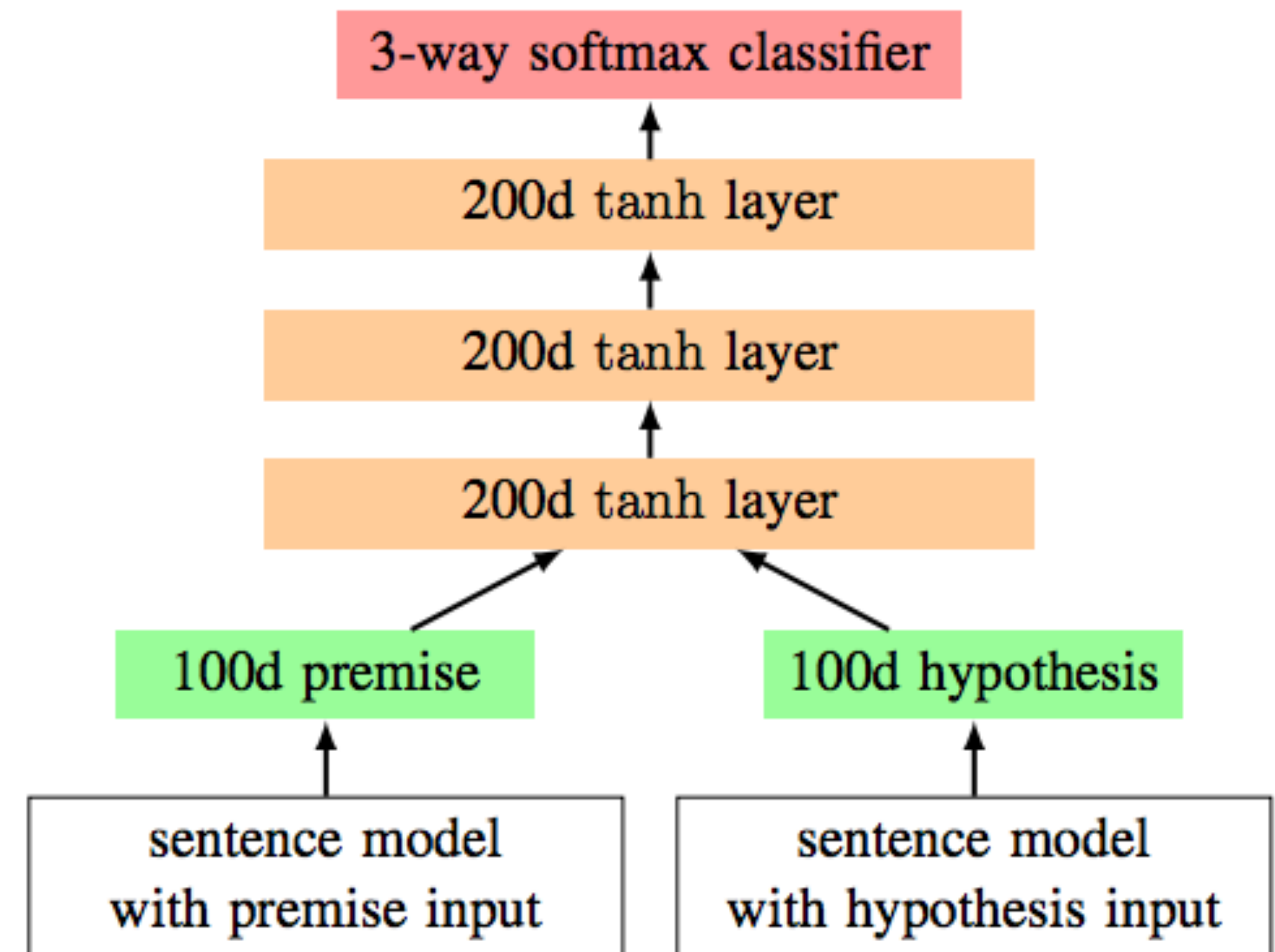
- ▶ Show people captions for (unseen) images and solicit entailed / neural / contradictory statements
- ▶ >500,000 sentence pairs
- ▶ Encode each sentence and process



Bowman et al. (2015)

SNLI Dataset

- ▶ Show people captions for (unseen) images and solicit entailed / neural / contradictory statements
 - ▶ >500,000 sentence pairs
 - ▶ Encode each sentence and process
- 100D LSTM: 78% accuracy



Bowman et al. (2015)

SNLI Dataset

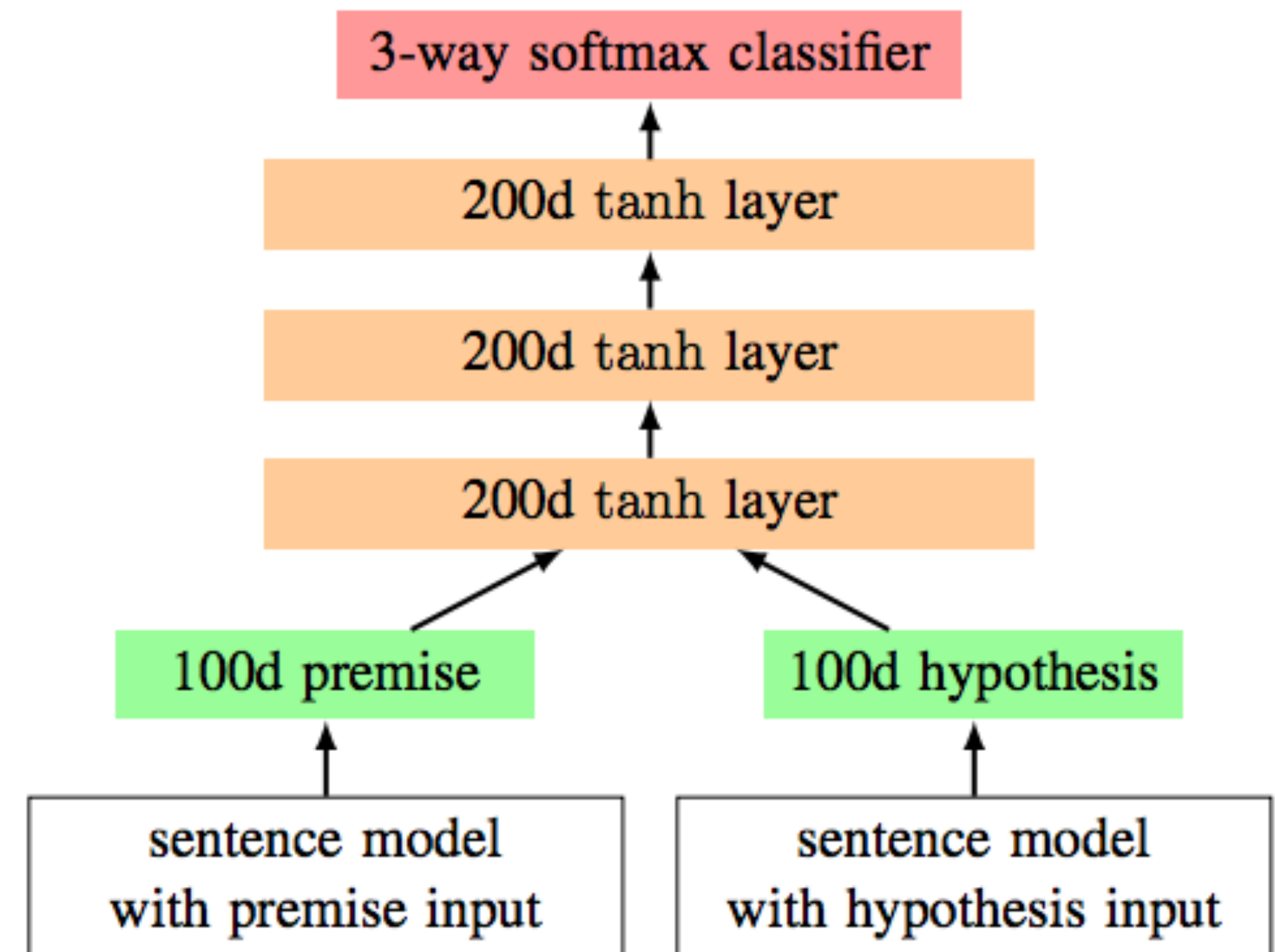
- ▶ Show people captions for (unseen) images and solicit entailed / neural / contradictory statements
- ▶ >500,000 sentence pairs

- ▶ Encode each sentence and process

100D LSTM: 78% accuracy

300D LSTM: 80% accuracy

(Bowman et al., 2016)



Bowman et al. (2015)

SNLI Dataset

- ▶ Show people captions for (unseen) images and solicit entailed / neural / contradictory statements
- ▶ >500,000 sentence pairs

- ▶ Encode each sentence and process

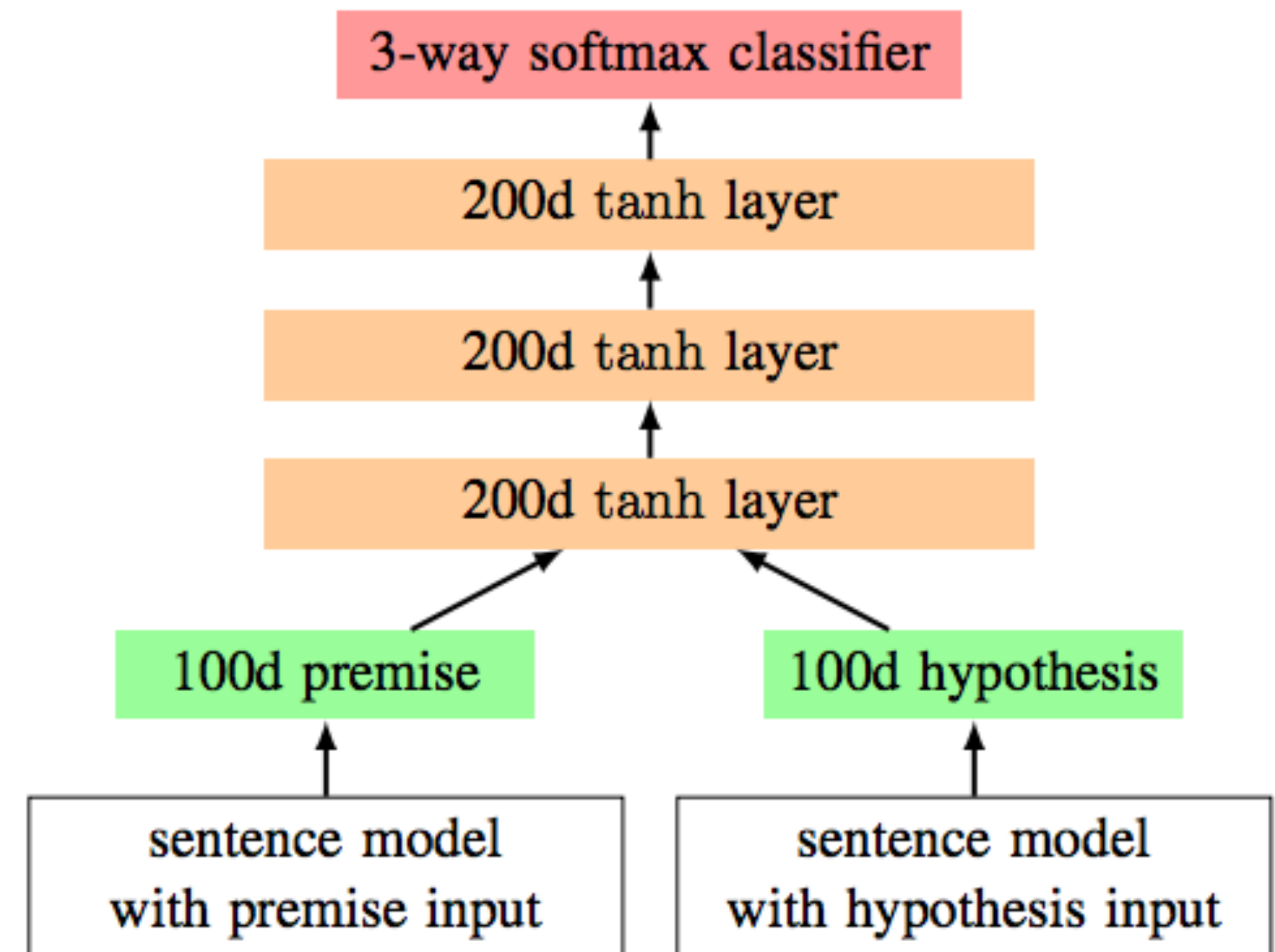
100D LSTM: 78% accuracy

300D LSTM: 80% accuracy

(Bowman et al., 2016)

300D BiLSTM: 83% accuracy

(Liu et al., 2016)



Bowman et al. (2015)

SNLI Dataset

- ▶ Show people captions for (unseen) images and solicit entailed / neural / contradictory statements
- ▶ >500,000 sentence pairs

- ▶ Encode each sentence and process

100D LSTM: 78% accuracy

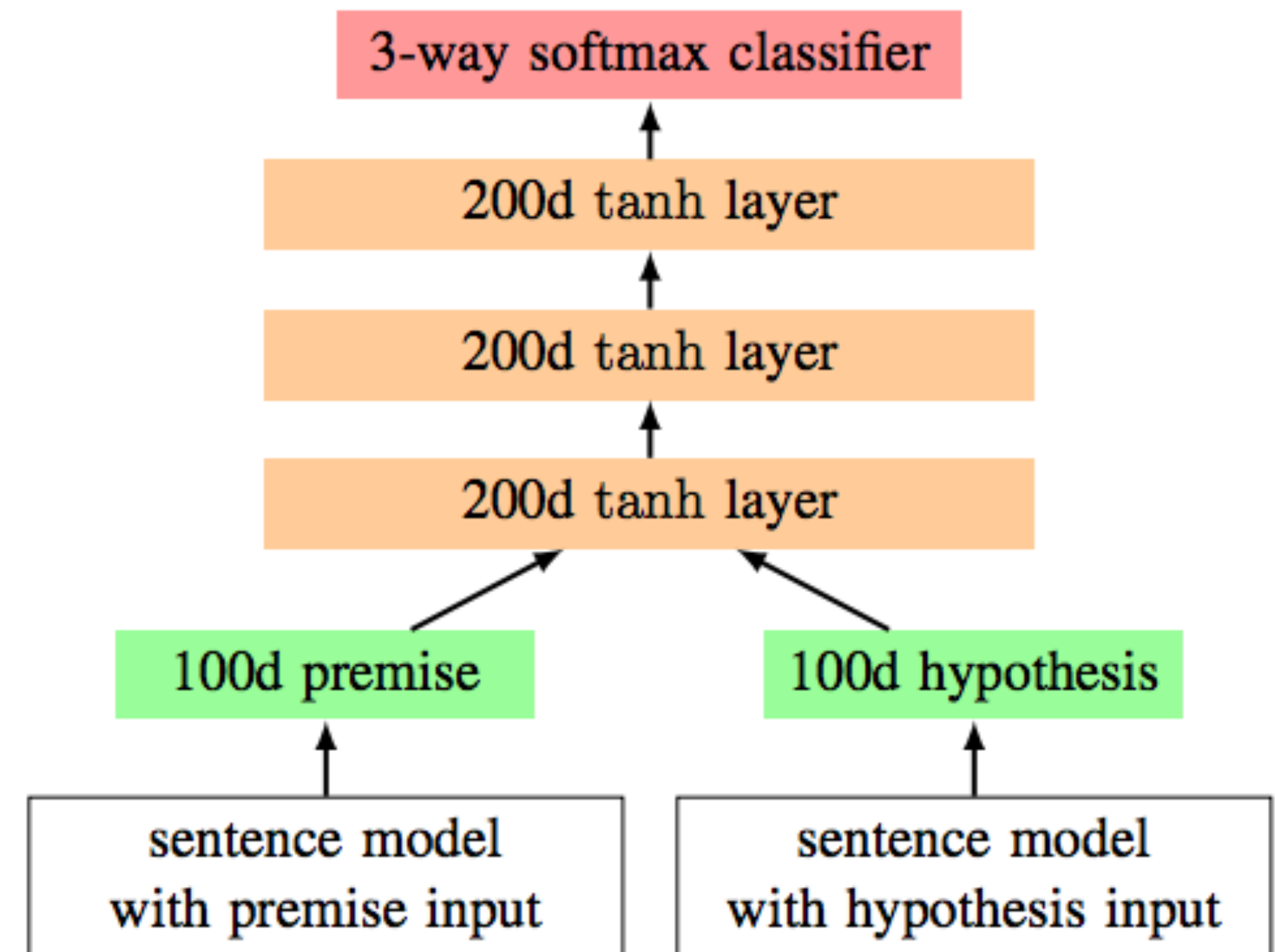
300D LSTM: 80% accuracy

(Bowman et al., 2016)

300D BiLSTM: 83% accuracy

(Liu et al., 2016)

- ▶ Later: better models for this



Bowman et al. (2015)

This Lecture

- ▶ CNNs
- ▶ CNNs for Sentiment
- ▶ Neural CRFs

CNNs

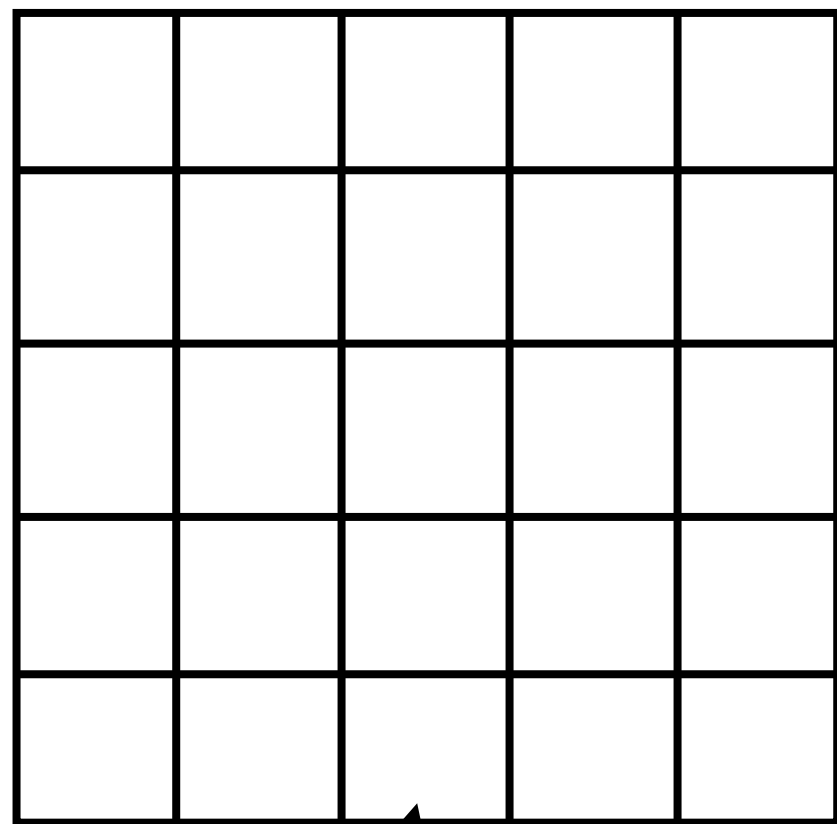
Convolutional Layer

- ▶ Applies a *filter* over patches of the input and returns that filter's activations
- ▶ Convolution: take dot product of filter with a patch of the input

Convolutional Layer

- ▶ Applies a *filter* over patches of the input and returns that filter's activations
- ▶ Convolution: take dot product of filter with a patch of the input

image: $n \times n \times k$

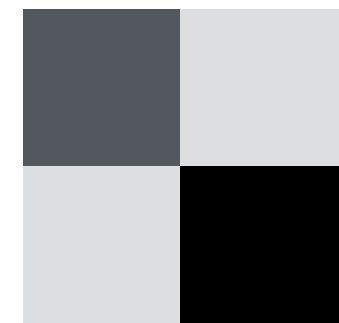
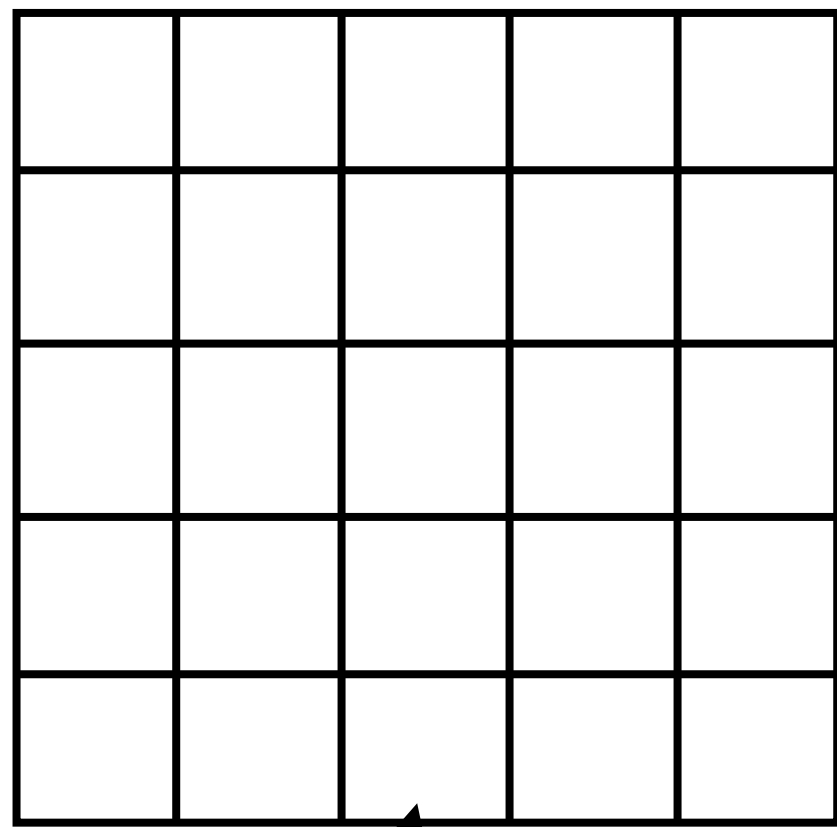


Each of these cells is a vector with multiple values
Images: RGB values (3 dim)

Convolutional Layer

- ▶ Applies a *filter* over patches of the input and returns that filter's activations
- ▶ Convolution: take dot product of filter with a patch of the input

image: $n \times n \times k$ filter: $m \times m \times k$

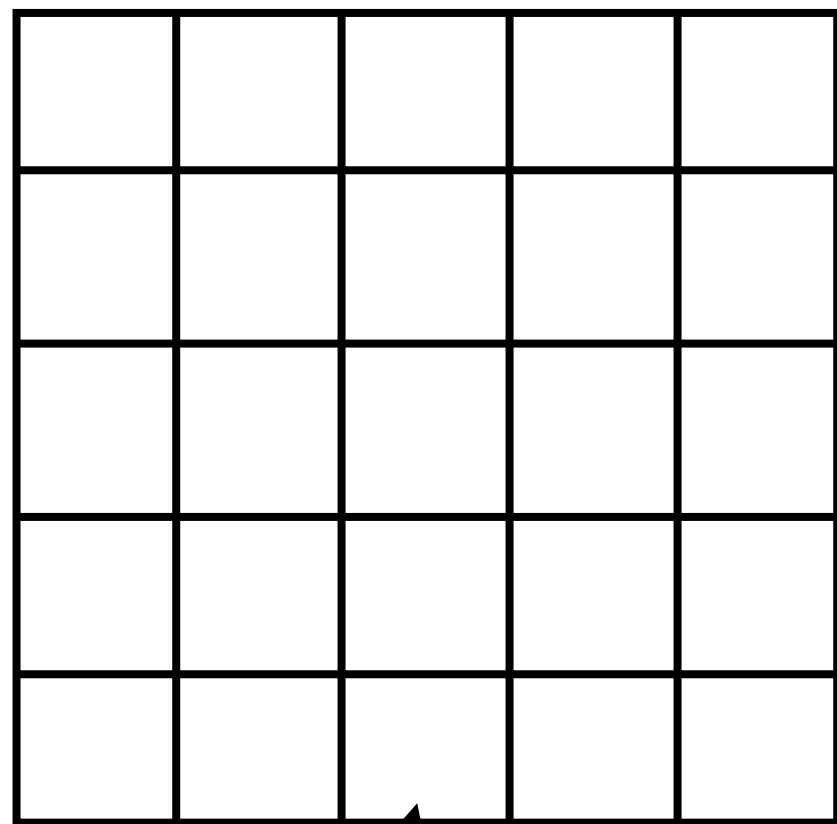


Each of these cells is a vector with multiple values
Images: RGB values (3 dim)

Convolutional Layer

- ▶ Applies a *filter* over patches of the input and returns that filter's activations
- ▶ Convolution: take dot product of filter with a patch of the input

image: $n \times n \times k$ filter: $m \times m \times k$



sum over dot products

$$\text{activation}_{ij} = \sum_{i_o=0}^{k-1} \sum_{j_o=0}^{k-1} \text{image}(i + i_o, j + j_o) \cdot \text{filter}(i_o, j_o)$$

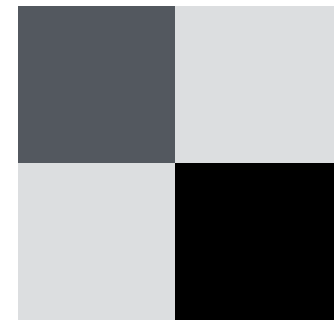
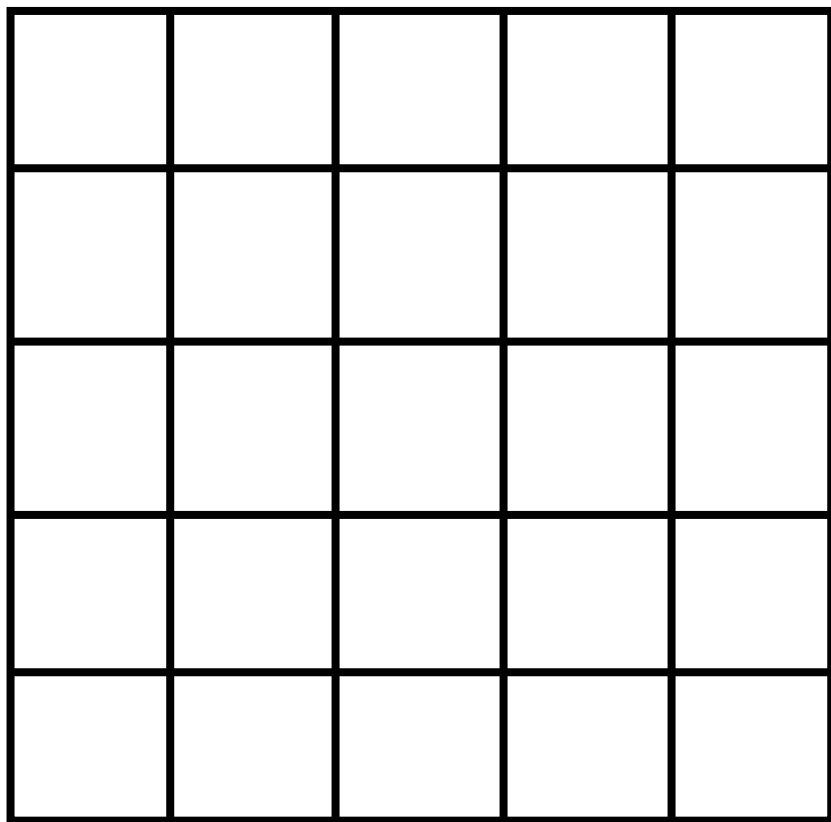
↑
offsets

Each of these cells is a vector with multiple values
Images: RGB values (3 dim)

Convolutional Layer

- ▶ Applies a *filter* over patches of the input and returns that filter's activations
- ▶ Convolution: take dot product of filter with a patch of the input

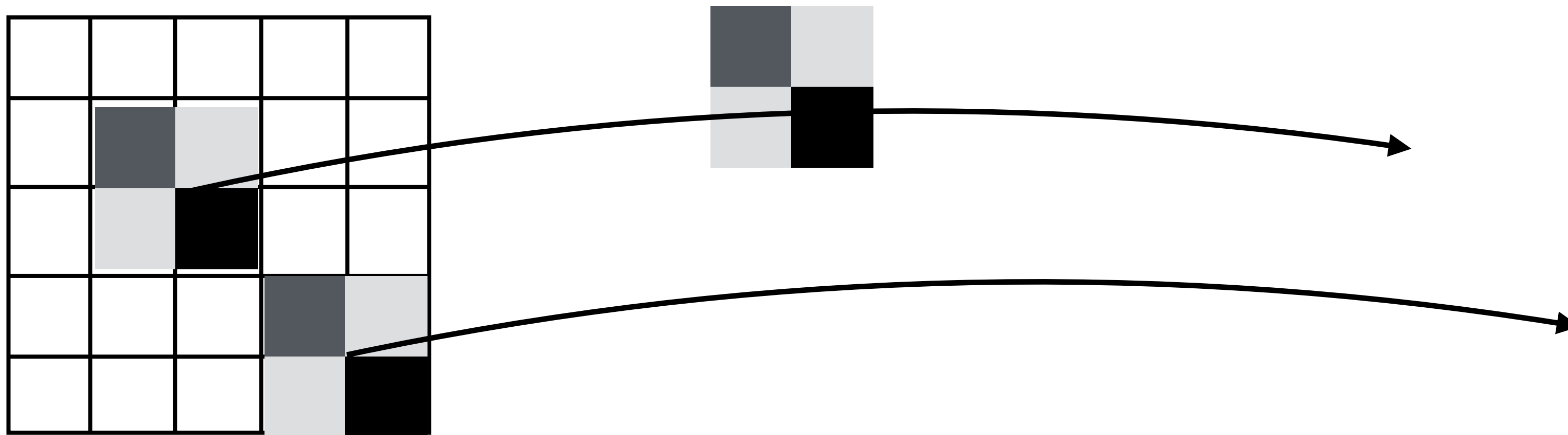
image: $n \times n \times k$ filter: $m \times m \times k$



Convolutional Layer

- ▶ Applies a *filter* over patches of the input and returns that filter's activations
- ▶ Convolution: take dot product of filter with a patch of the input

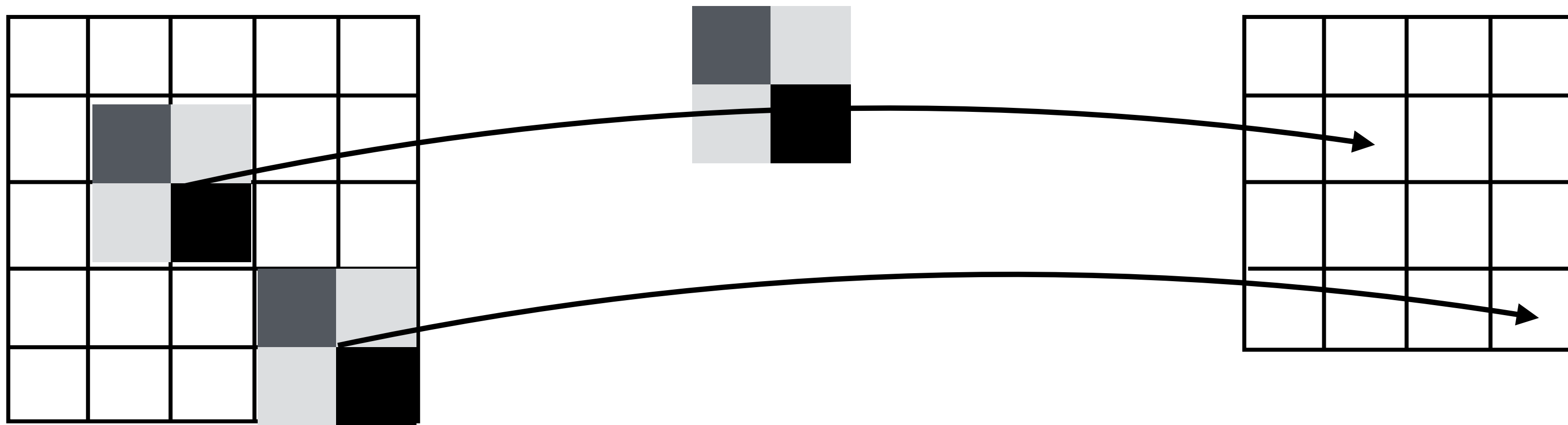
image: $n \times n \times k$ filter: $m \times m \times k$



Convolutional Layer

- ▶ Applies a *filter* over patches of the input and returns that filter's activations
- ▶ Convolution: take dot product of filter with a patch of the input

image: $n \times n \times k$ filter: $m \times m \times k$ activations: $(n - m + 1) \times (n - m + 1) \times 1$



Convolutions for NLP

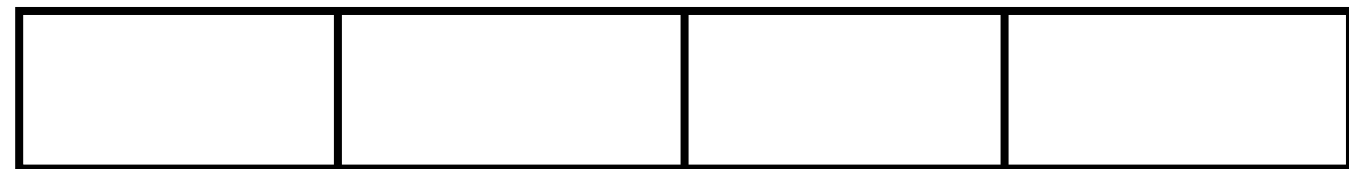
- ▶ Input and filter are 2-dimensional instead of 3-dimensional

Convolutions for NLP

- ▶ Input and filter are 2-dimensional instead of 3-dimensional

sentence: n words \times k vec dim

the movie was good



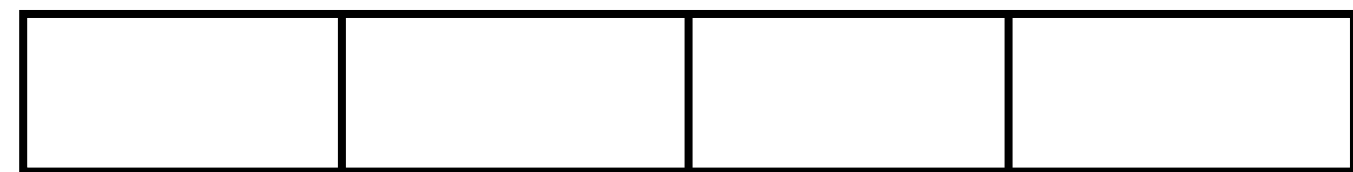
 vector for each word

Convolutions for NLP

- ▶ Input and filter are 2-dimensional instead of 3-dimensional

sentence: n words \times k vec dim filter: $m \times k$

the movie was good



 vector for each word

Convolutions for NLP

- ▶ Input and filter are 2-dimensional instead of 3-dimensional

sentence: n words \times k vec dim filter: $m \times k$

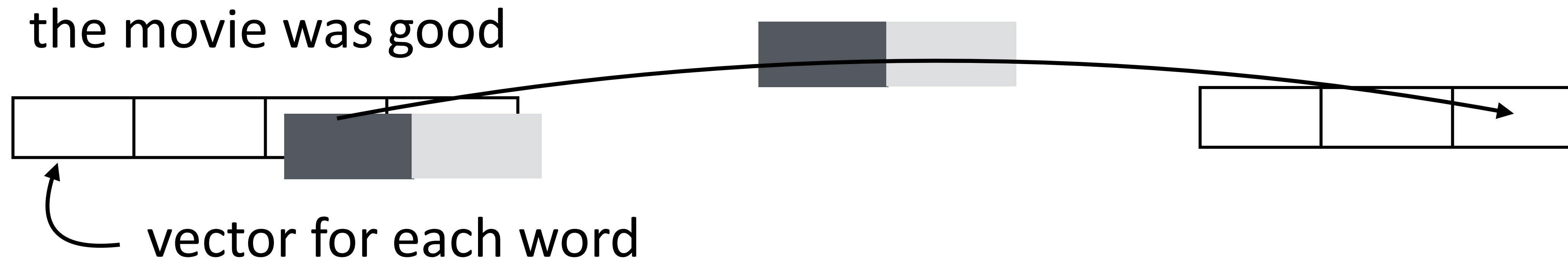
the movie was good



Convolutions for NLP

- ▶ Input and filter are 2-dimensional instead of 3-dimensional

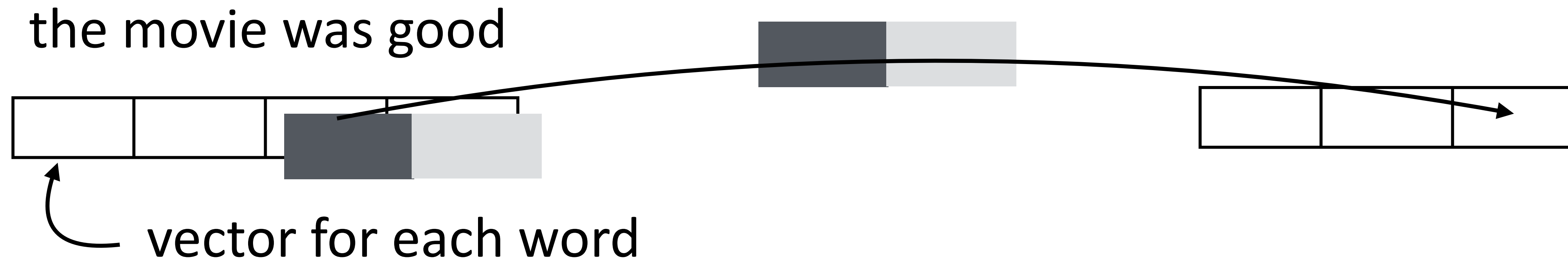
sentence: n words \times k vec dim filter: $m \times k$ activations: $(n - m + 1) \times 1$



Convolutions for NLP

- ▶ Input and filter are 2-dimensional instead of 3-dimensional

sentence: n words \times k vec dim filter: $m \times k$ activations: $(n - m + 1) \times 1$



- ▶ Combines evidence locally in a sentence and produces a new (but still variable-length) representation

Compare: CNNs vs. LSTMs



$n \times k$

the movie was good

Compare: CNNs vs. LSTMs



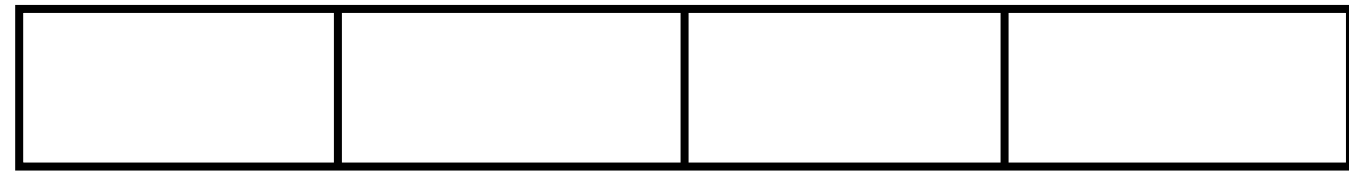
c filters,
 $m \times k$ each



$n \times k$

the movie was good

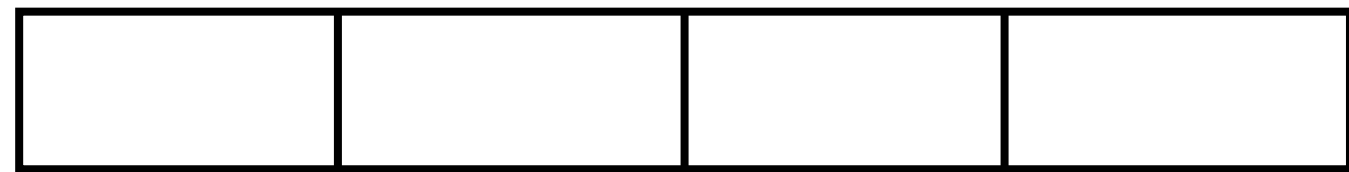
Compare: CNNs vs. LSTMs



$O(n) \times c$



c filters,
 $m \times k$ each



$n \times k$

the movie was good

Compare: CNNs vs. LSTMs



$O(n) \times c$

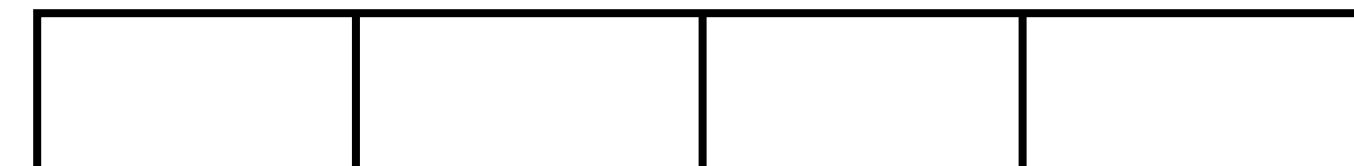


c filters,
 $m \times k$ each



$n \times k$

the movie was good



$n \times k$

the movie was good

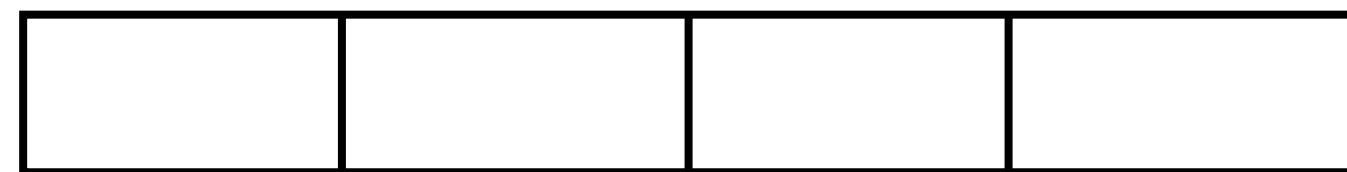
Compare: CNNs vs. LSTMs



$O(n) \times c$

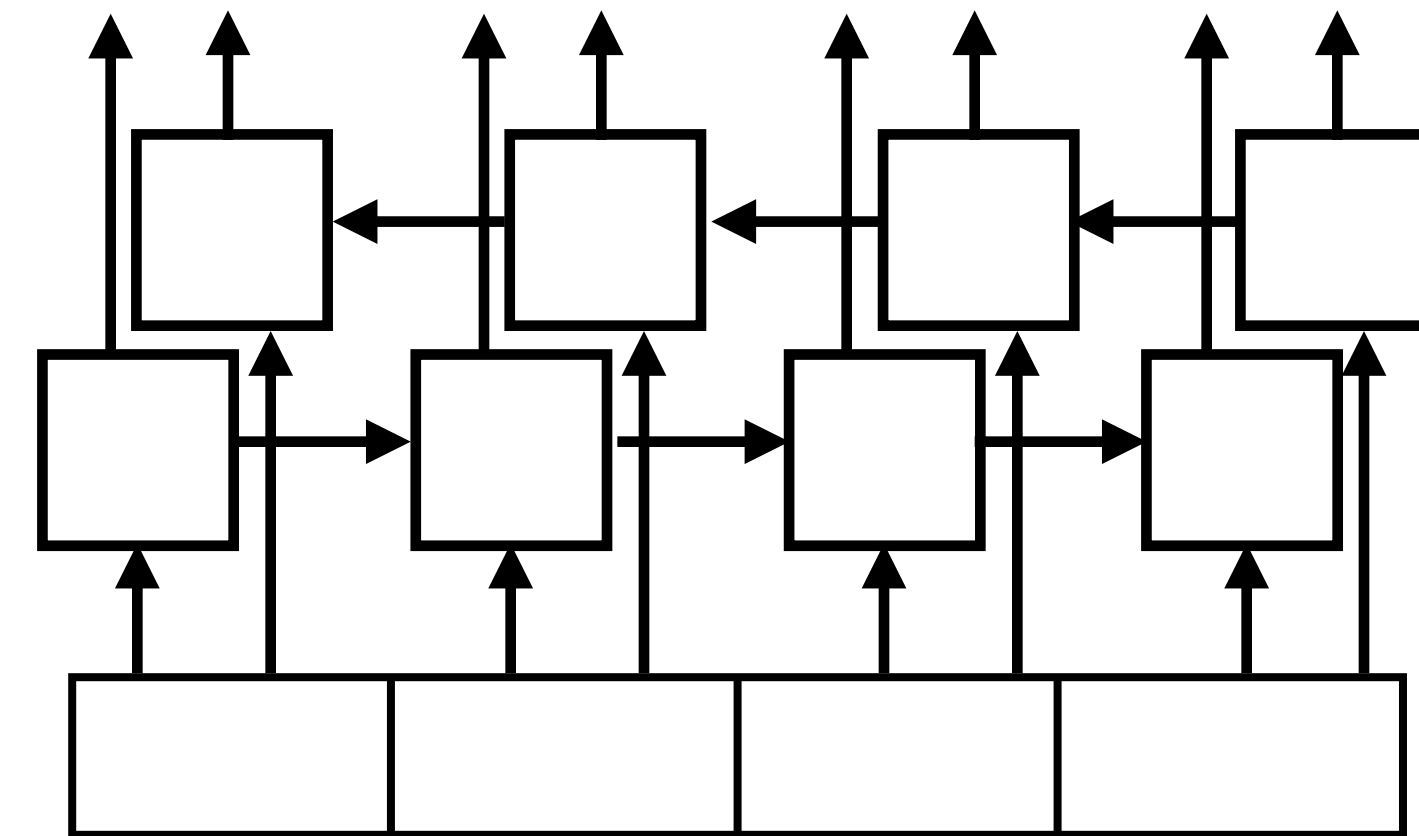


c filters,
 $m \times k$ each



$n \times k$

the movie was good



BiLSTM with
hidden size c

$n \times k$

the movie was good

Compare: CNNs vs. LSTMs



$O(n) \times c$

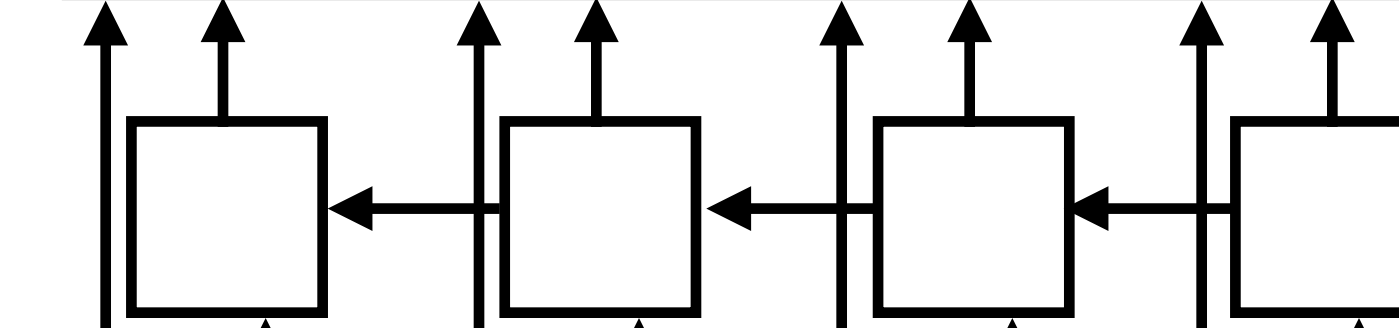
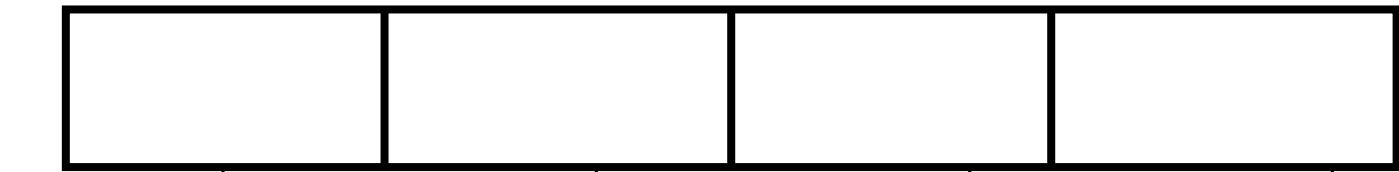


c filters,
 $m \times k$ each

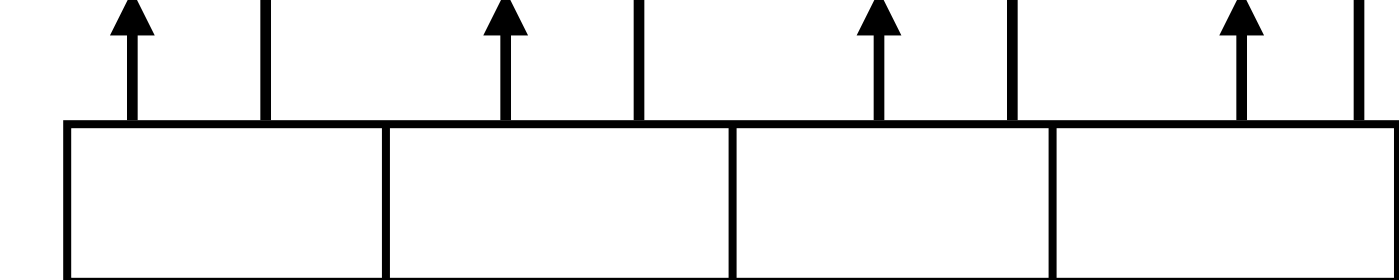
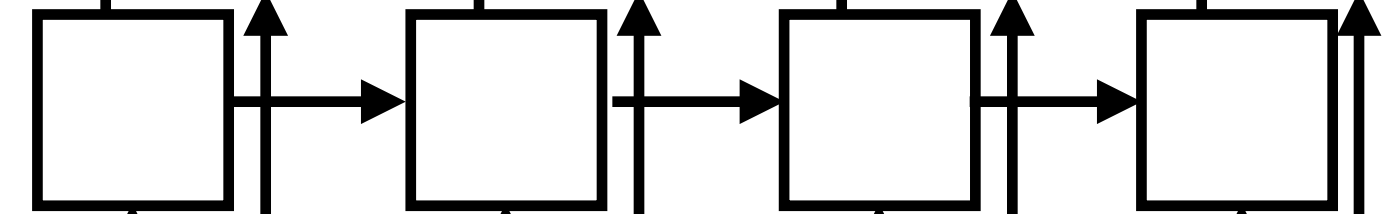


$n \times k$

the movie was good



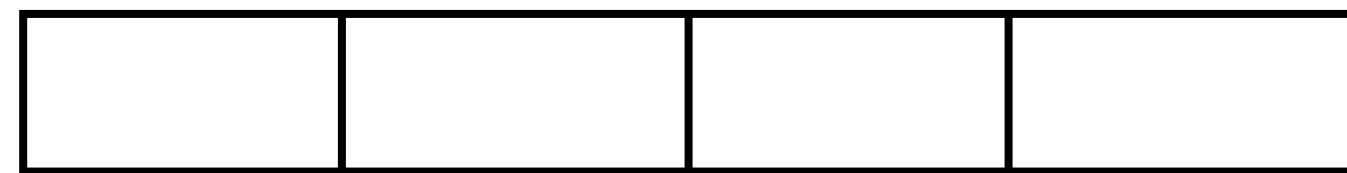
BiLSTM with
hidden size c



$n \times k$

the movie was good

Compare: CNNs vs. LSTMs



$O(n) \times c$

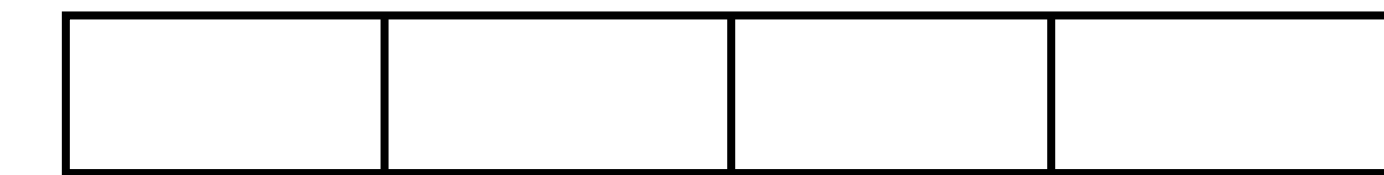


c filters,
 $m \times k$ each

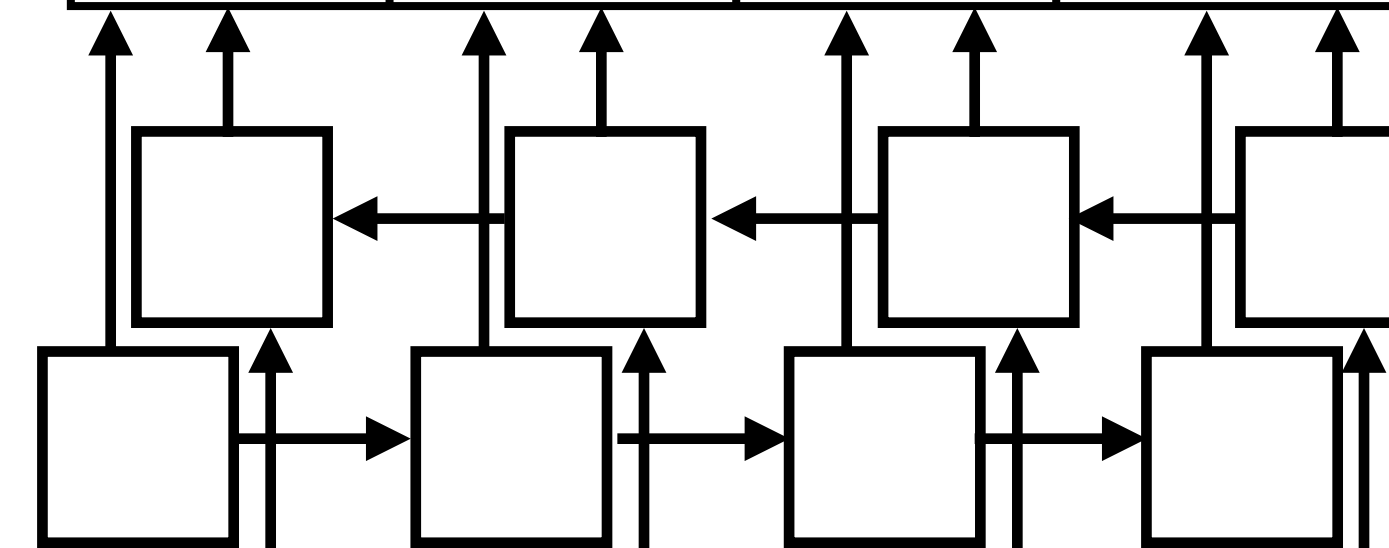


$n \times k$

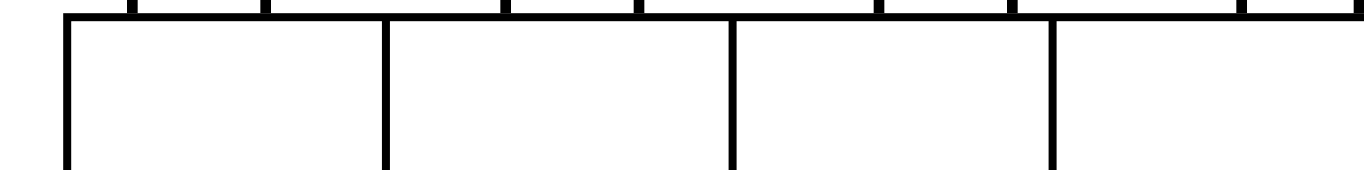
the movie was good



$n \times 2c$



BiLSTM with
hidden size c



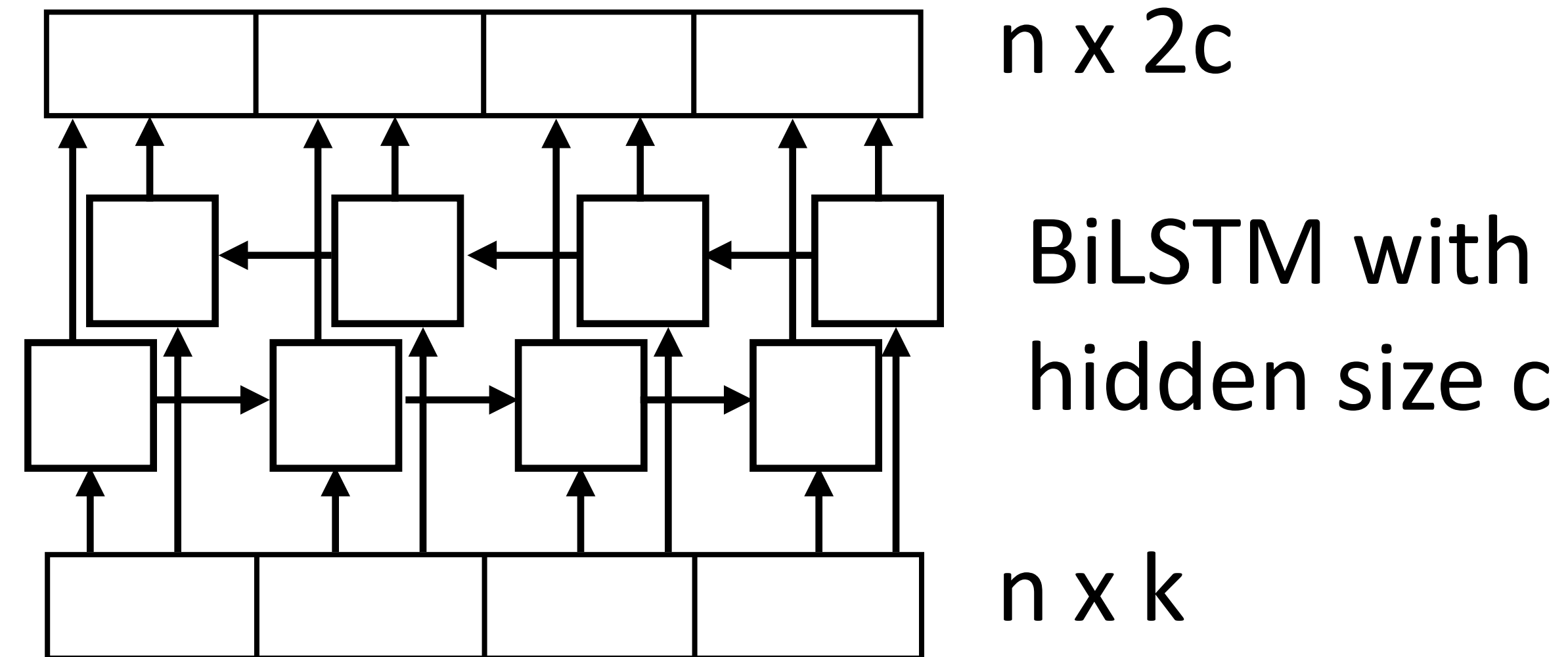
$n \times k$

the movie was good

Compare: CNNs vs. LSTMs



the movie was good



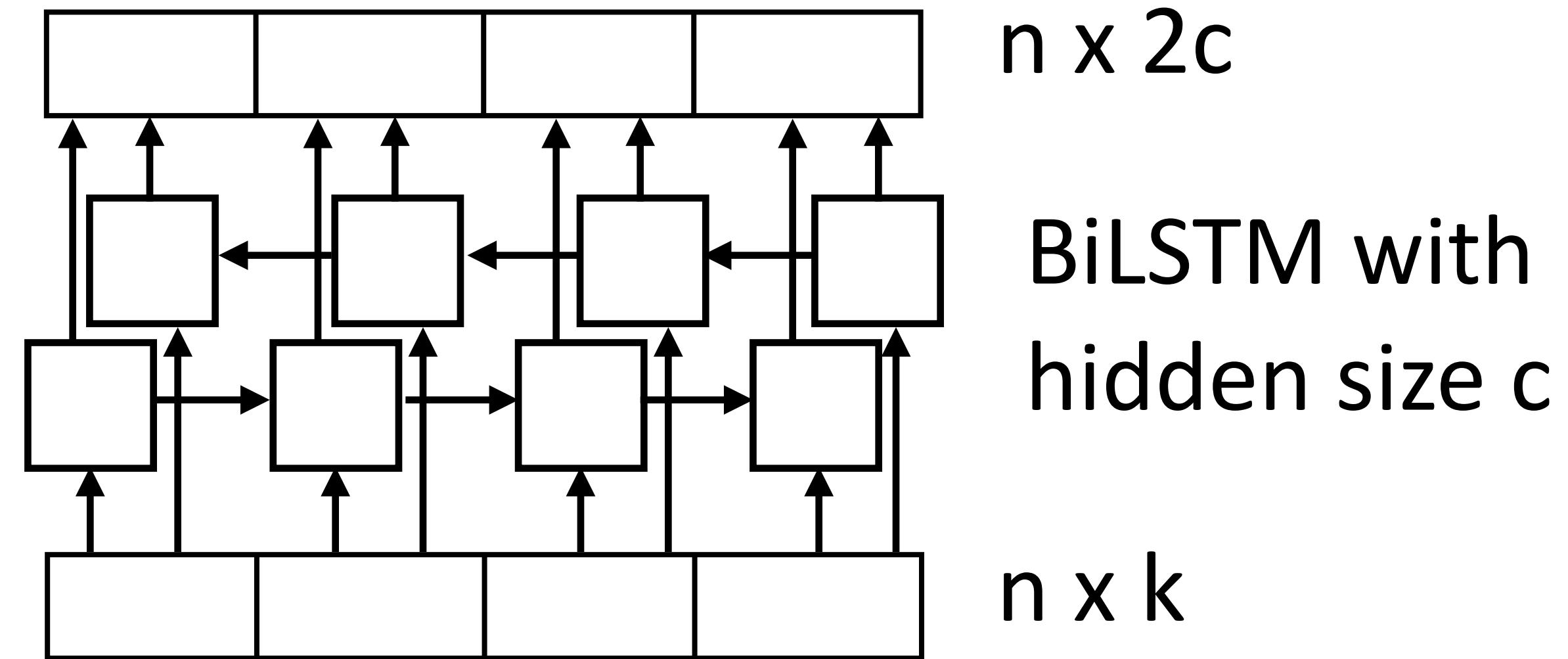
the movie was good

- ▶ Both LSTMs and convolutional layers transform the input using context

Compare: CNNs vs. LSTMs



the movie was good



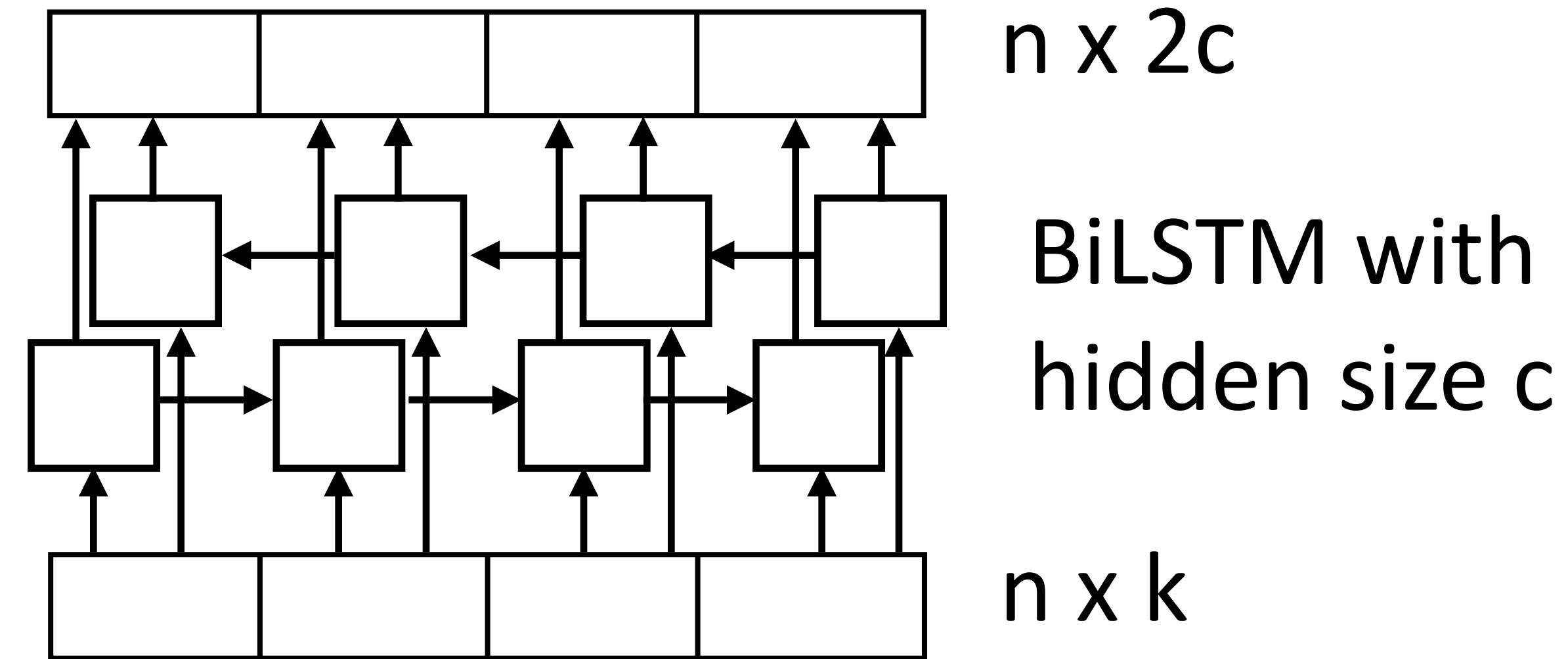
the movie was good

- ▶ Both LSTMs and convolutional layers transform the input using context
- ▶ LSTM: “globally” looks at the entire sentence (but local for many problems)

Compare: CNNs vs. LSTMs



the movie was good

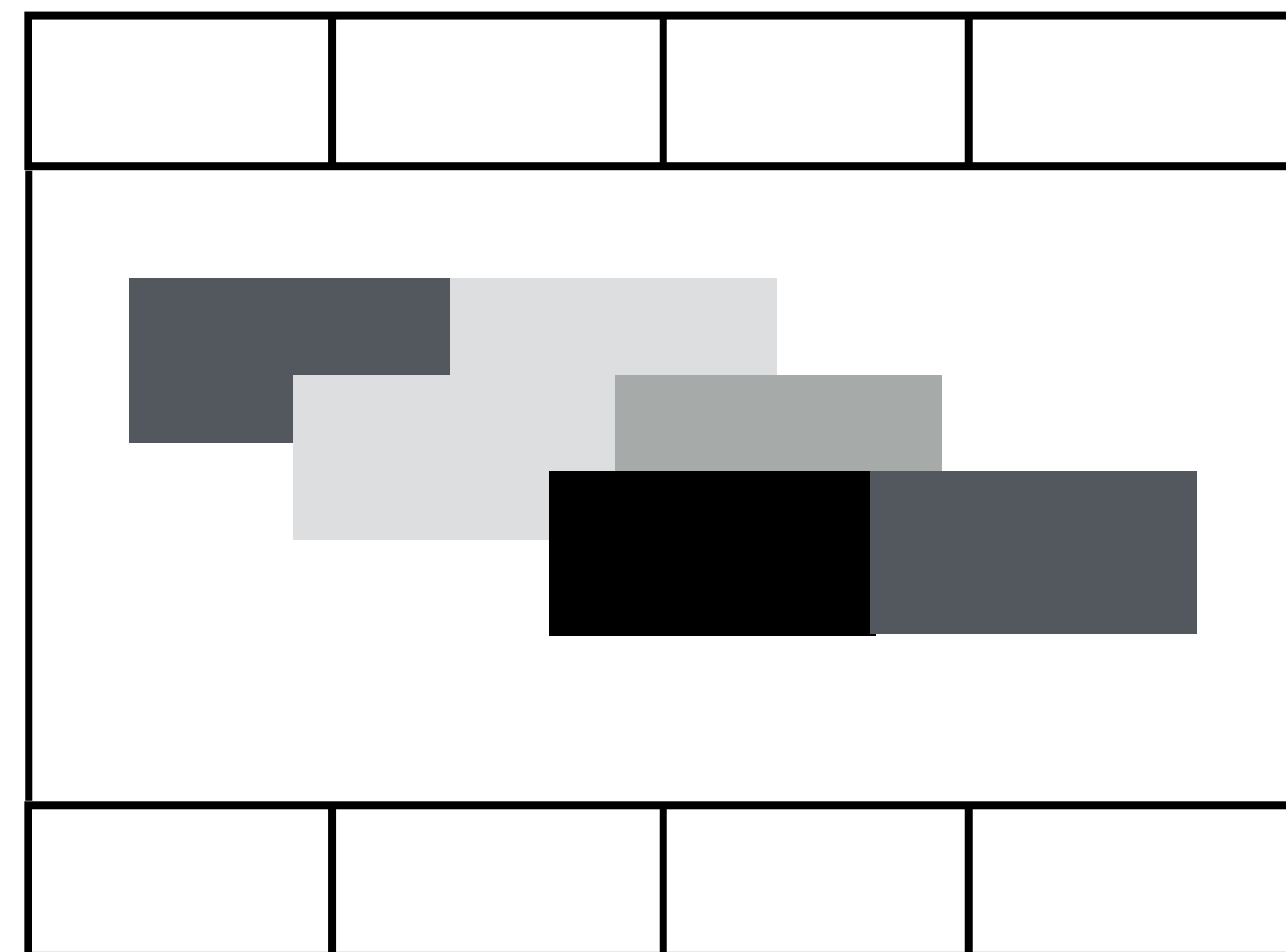


the movie was good

- ▶ Both LSTMs and convolutional layers transform the input using context
- ▶ LSTM: “globally” looks at the entire sentence (but local for many problems)
- ▶ CNN: local depending on filter width + number of layers

CNNs for Sentiment

CNNs for Sentiment Analysis



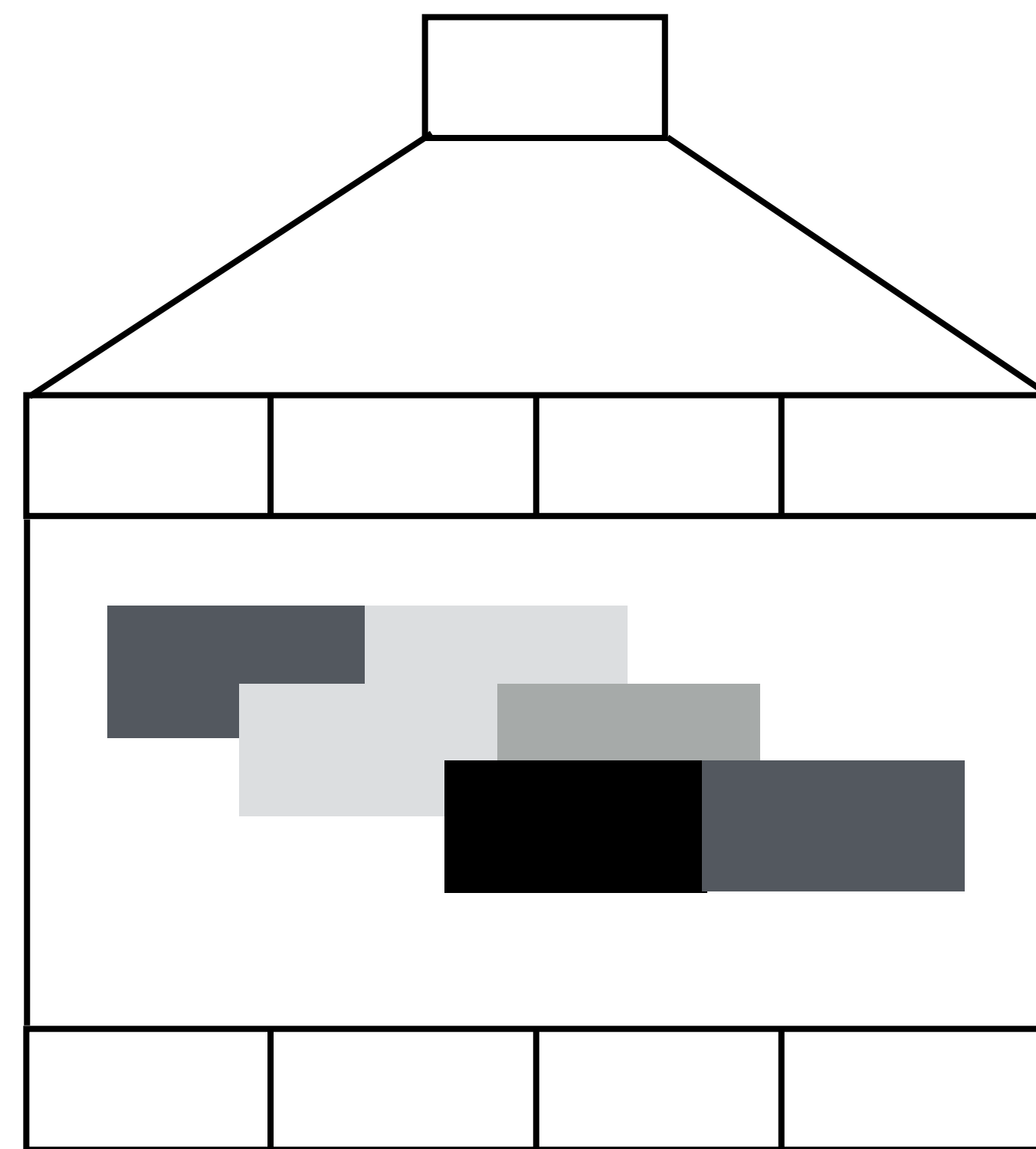
$n \times c$

c filters,
 $m \times k$ each

$n \times k$

the movie was good

CNNs for Sentiment Analysis



the movie was good

c-dimensional vector

max pooling over the sentence

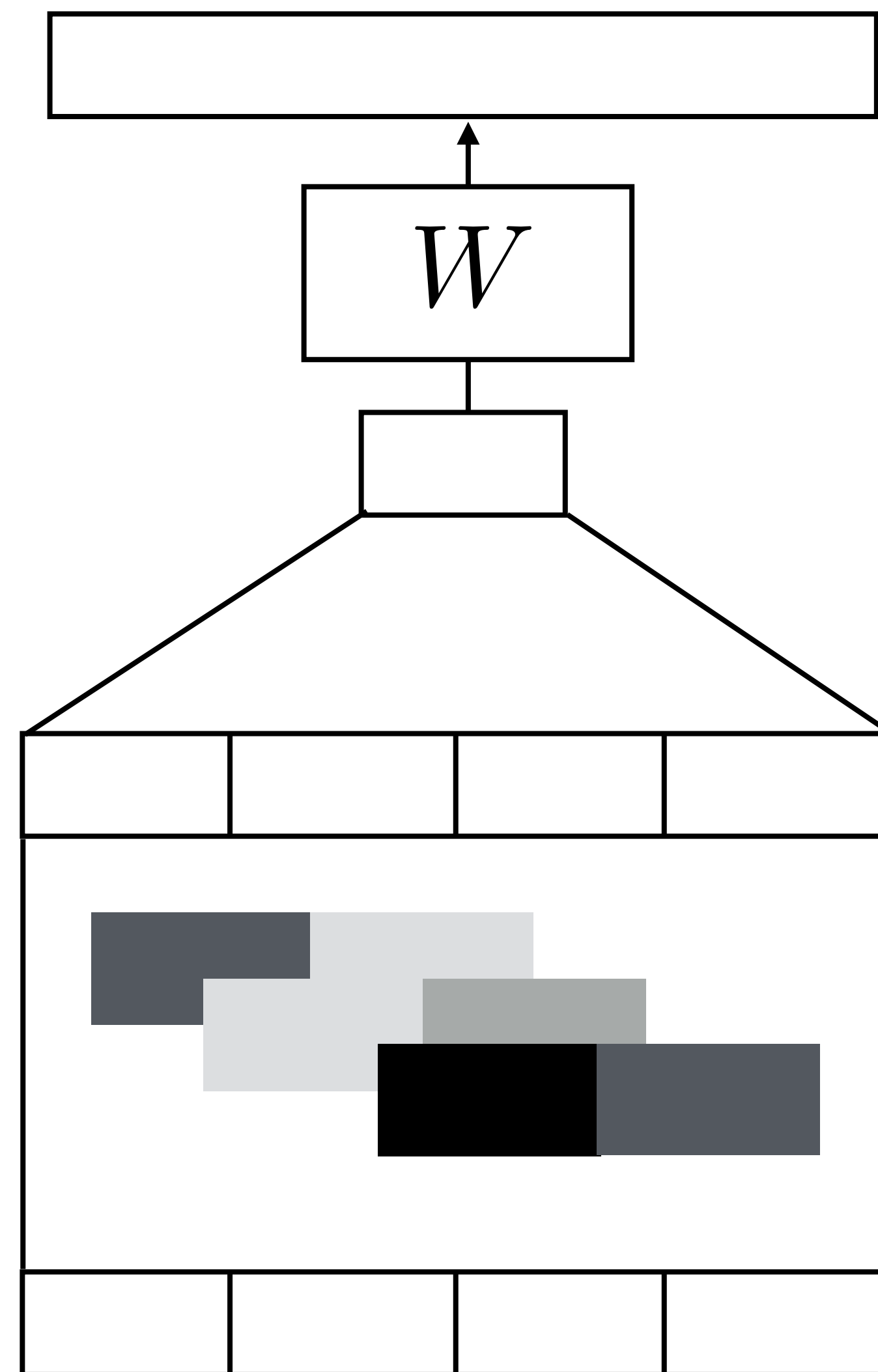
$n \times c$

c filters,
 $m \times k$ each

$n \times k$

- ▶ Max pooling: return the max activation of a given filter over the entire sentence; like a logical OR (sum pooling is like logical AND)

CNNs for Sentiment Analysis



$$P(y|\mathbf{x})$$

projection + softmax

c-dimensional vector

max pooling over the sentence

$n \times c$

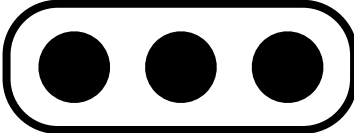
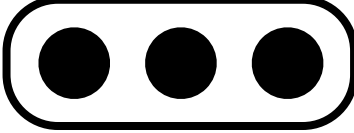


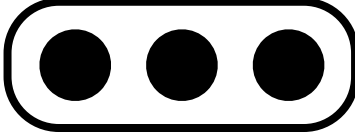
c filters,
 $m \times k$ each

$n \times k$

- ▶ Max pooling: return the max activation of a given filter over the entire sentence; like a logical OR (sum pooling is like logical AND)

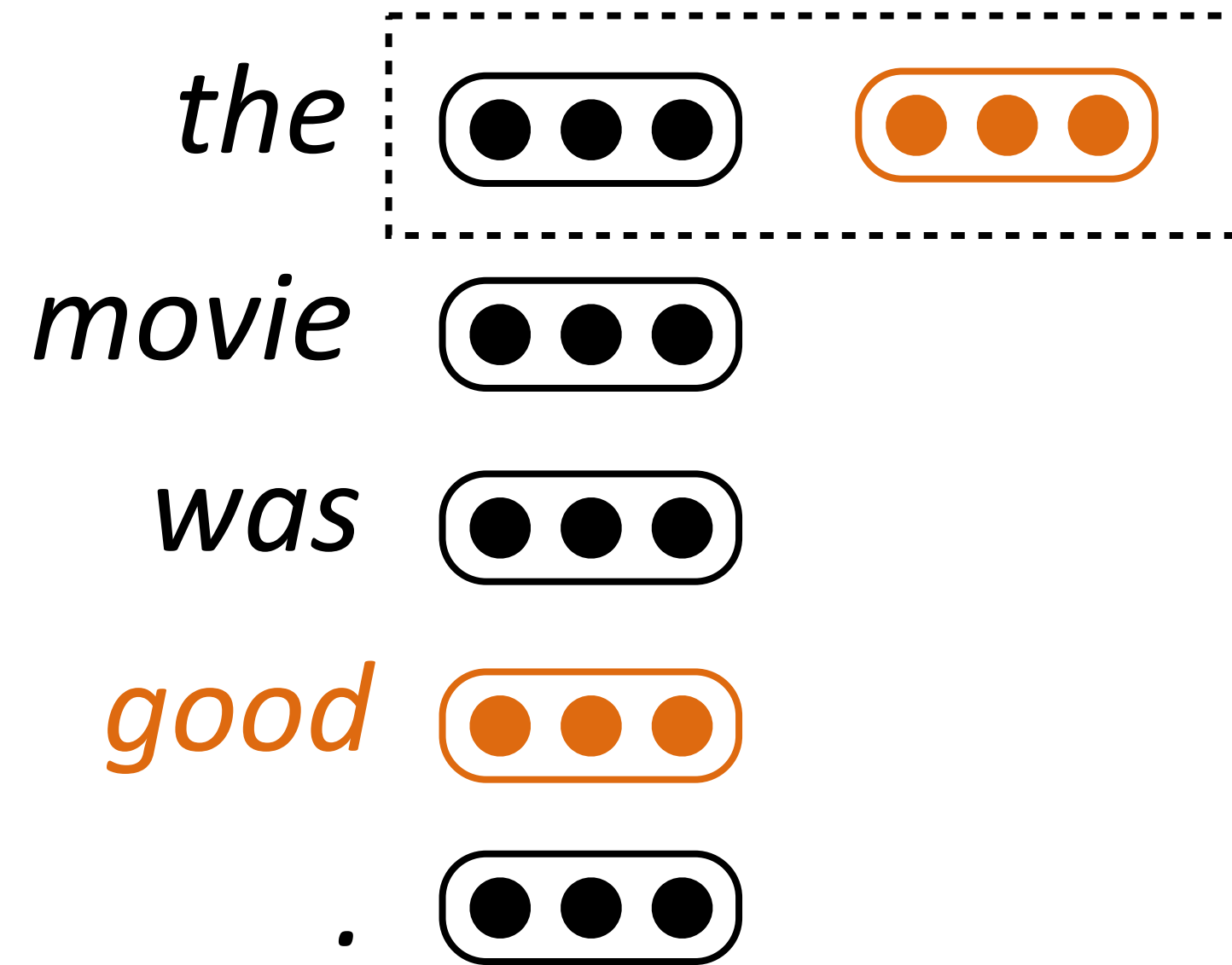
the movie was good

Understanding CNNs for Sentiment

the 
movie 
was 
good 
.


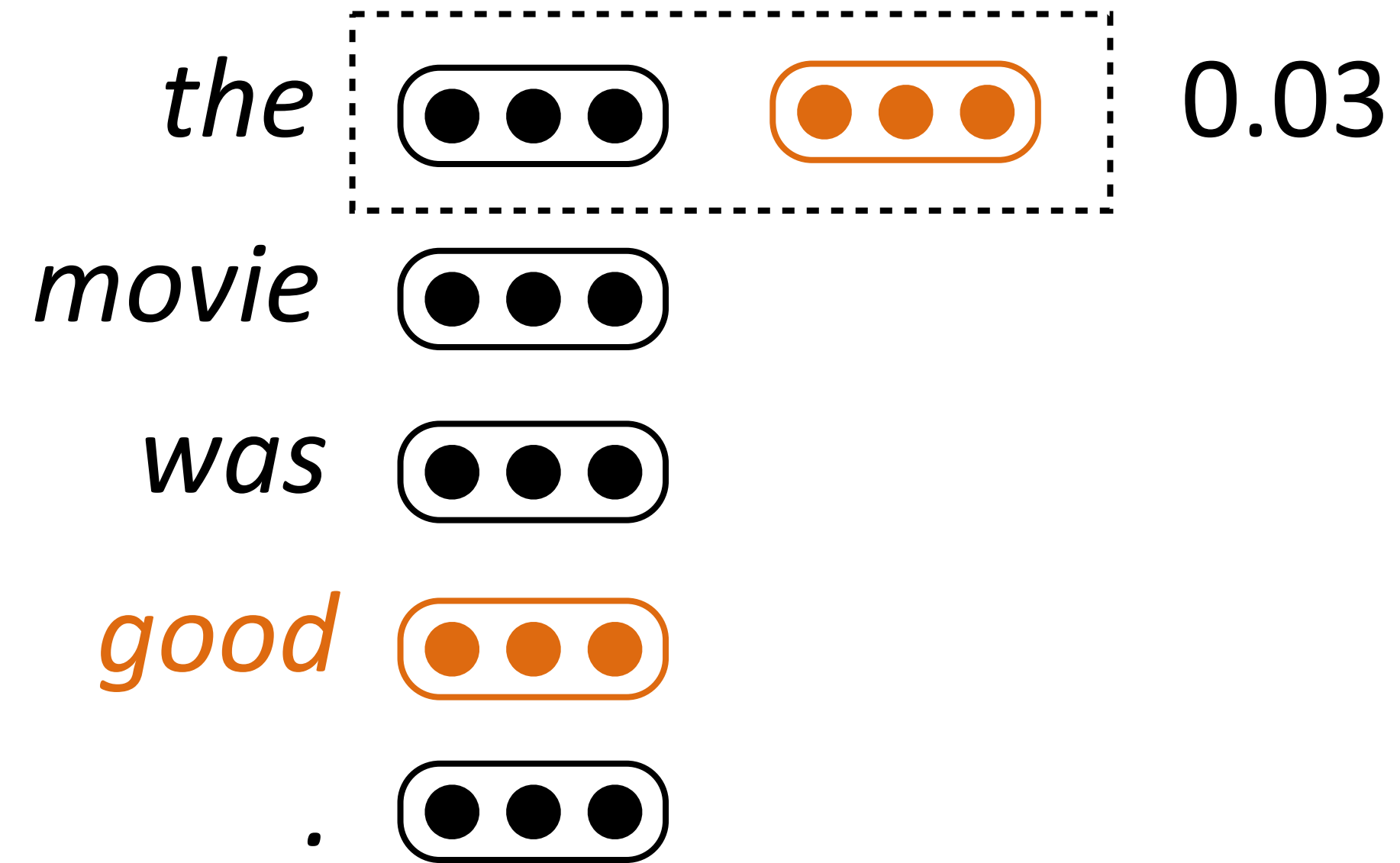
- ▶ Filter “looks like” the things that will cause it to have high activation

Understanding CNNs for Sentiment



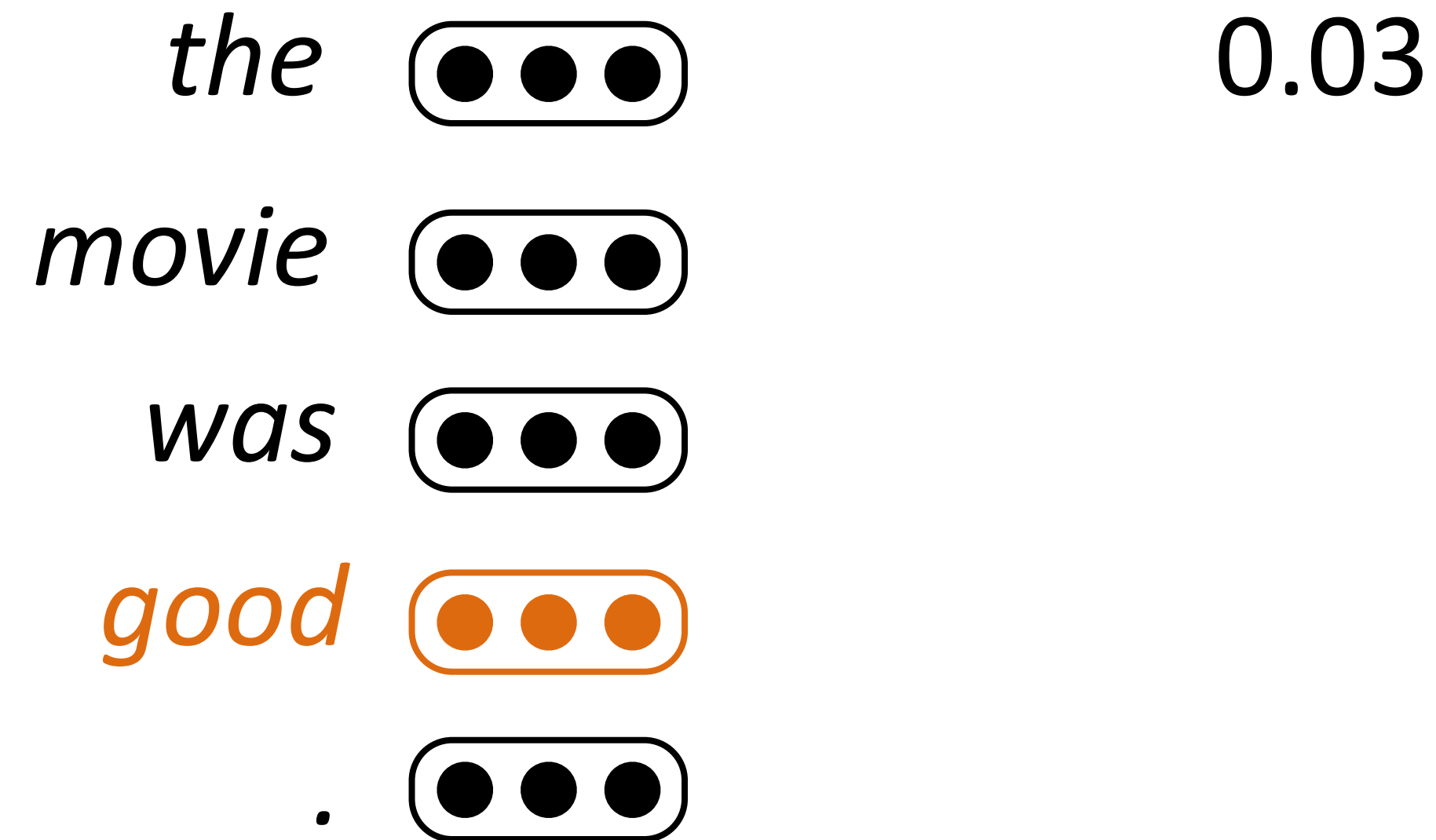
- ▶ Filter “looks like” the things that will cause it to have high activation

Understanding CNNs for Sentiment



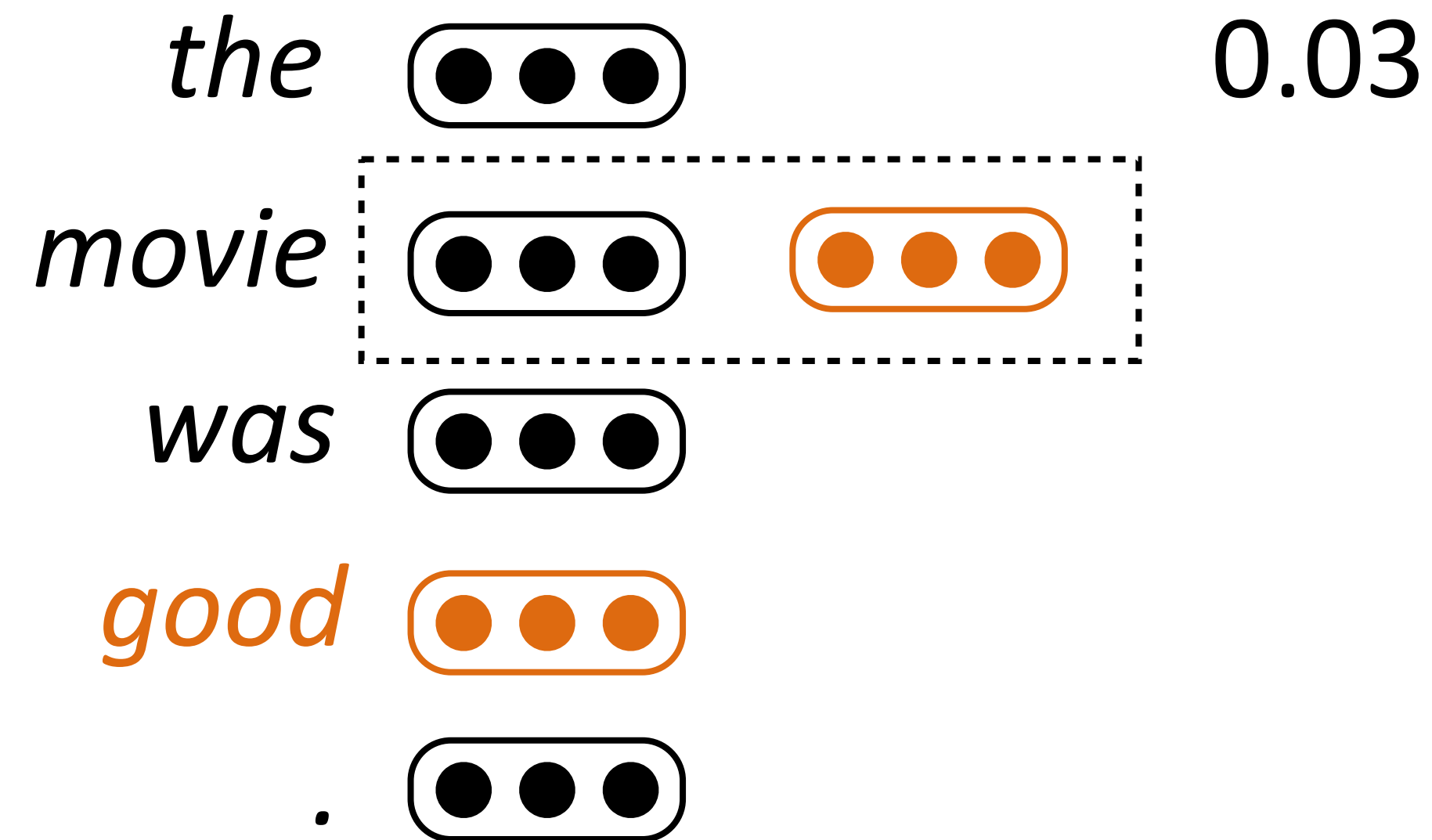
- ▶ Filter “looks like” the things that will cause it to have high activation

Understanding CNNs for Sentiment



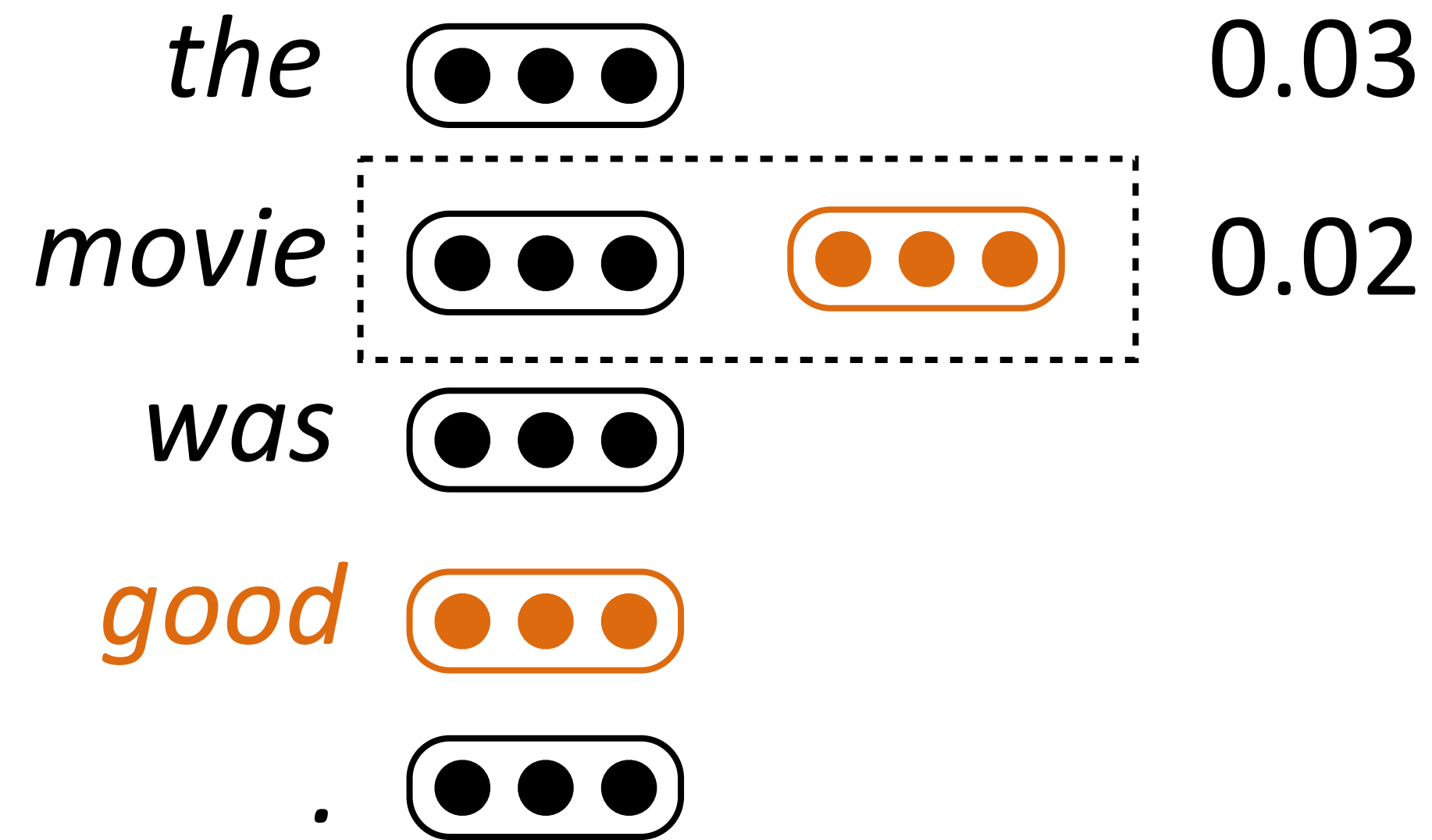
- ▶ Filter “looks like” the things that will cause it to have high activation

Understanding CNNs for Sentiment



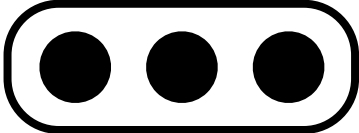
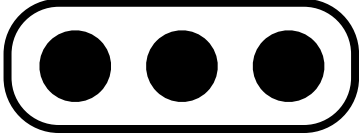
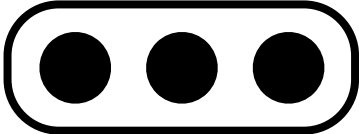

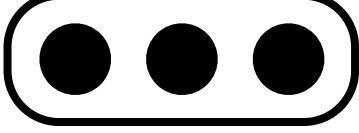
- ▶ Filter “looks like” the things that will cause it to have high activation

Understanding CNNs for Sentiment



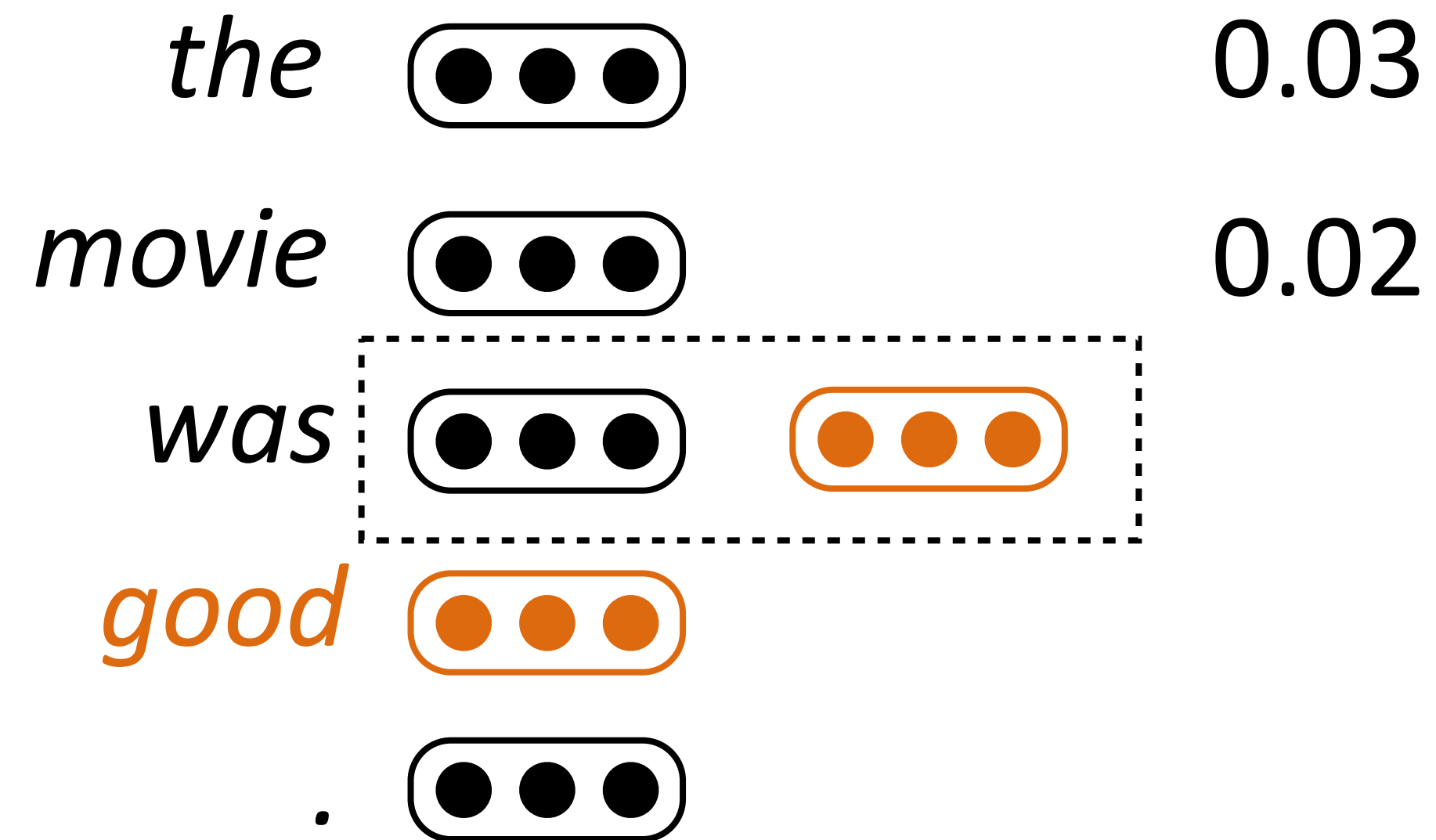
- ▶ Filter “looks like” the things that will cause it to have high activation

Understanding CNNs for Sentiment

<i>the</i>		0.03
<i>movie</i>		0.02
<i>was</i>		
<i>good</i>		
.		

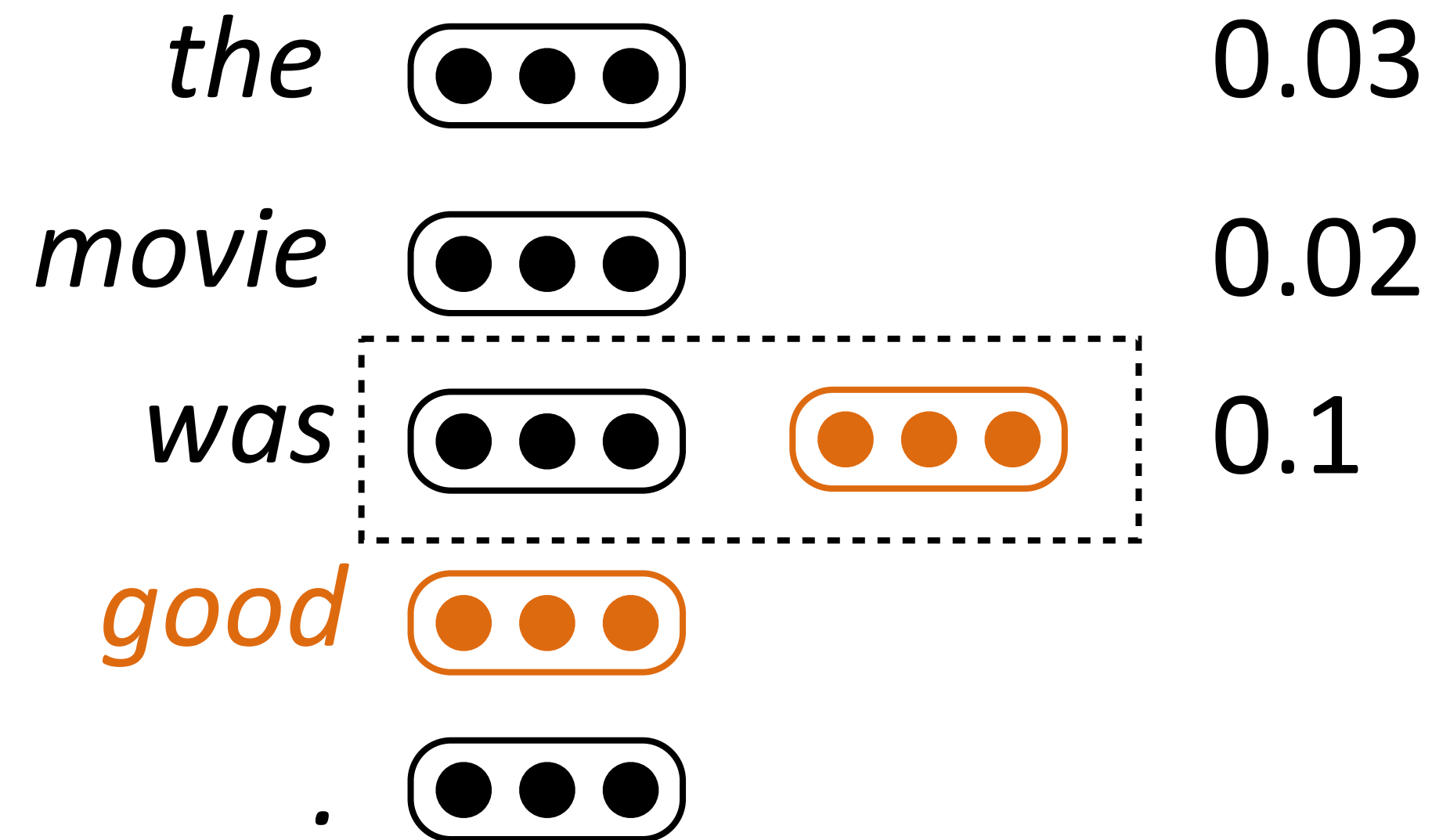
- ▶ Filter “looks like” the things that will cause it to have high activation

Understanding CNNs for Sentiment



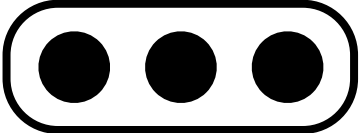
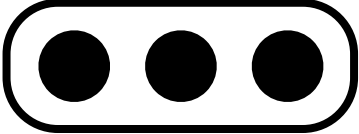
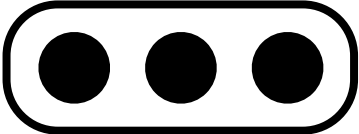

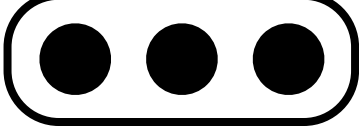
- ▶ Filter “looks like” the things that will cause it to have high activation

Understanding CNNs for Sentiment



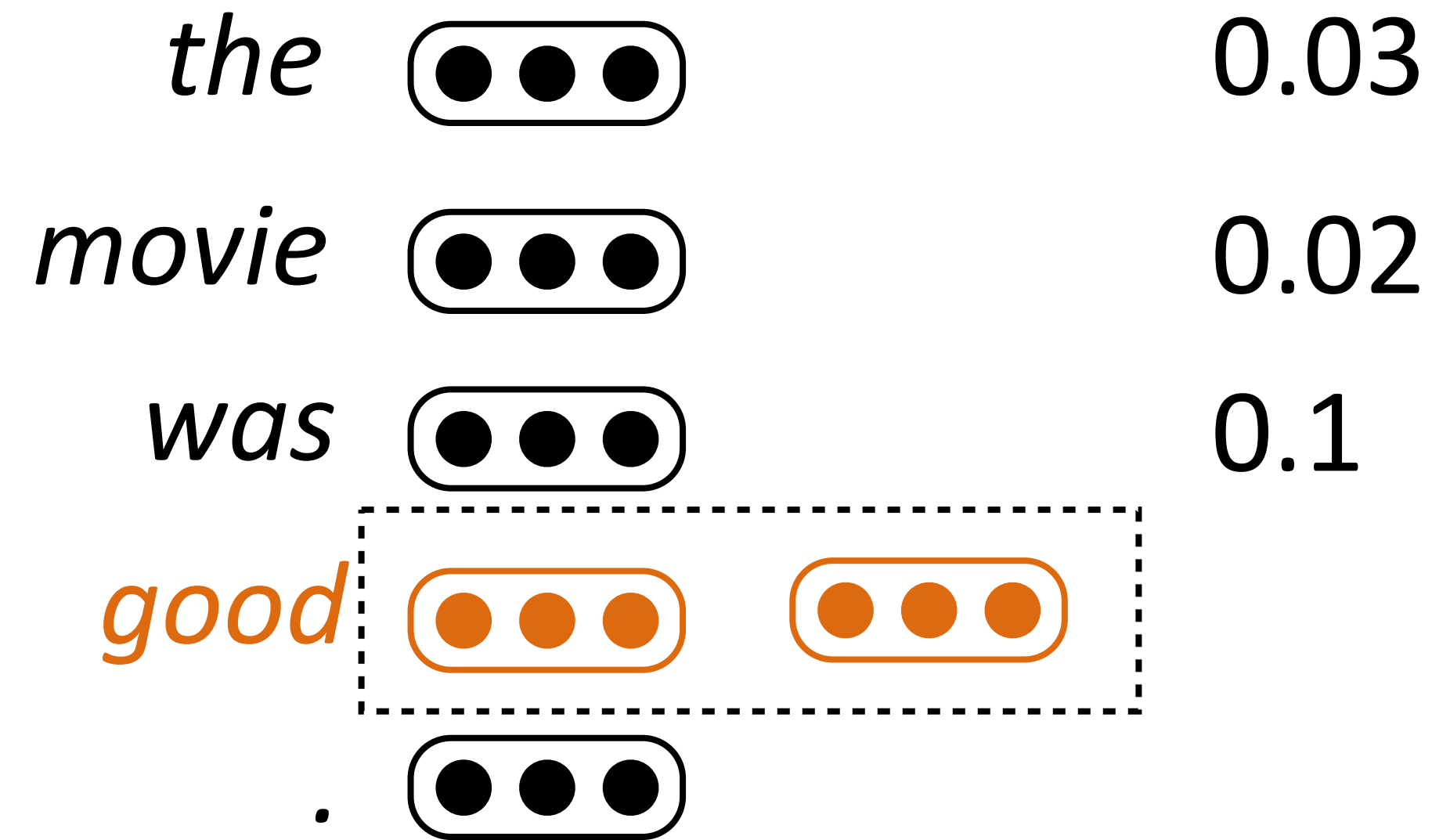
- ▶ Filter “looks like” the things that will cause it to have high activation

Understanding CNNs for Sentiment

<i>the</i>		0.03
<i>movie</i>		0.02
<i>was</i>		0.1
<i>good</i>		
.		

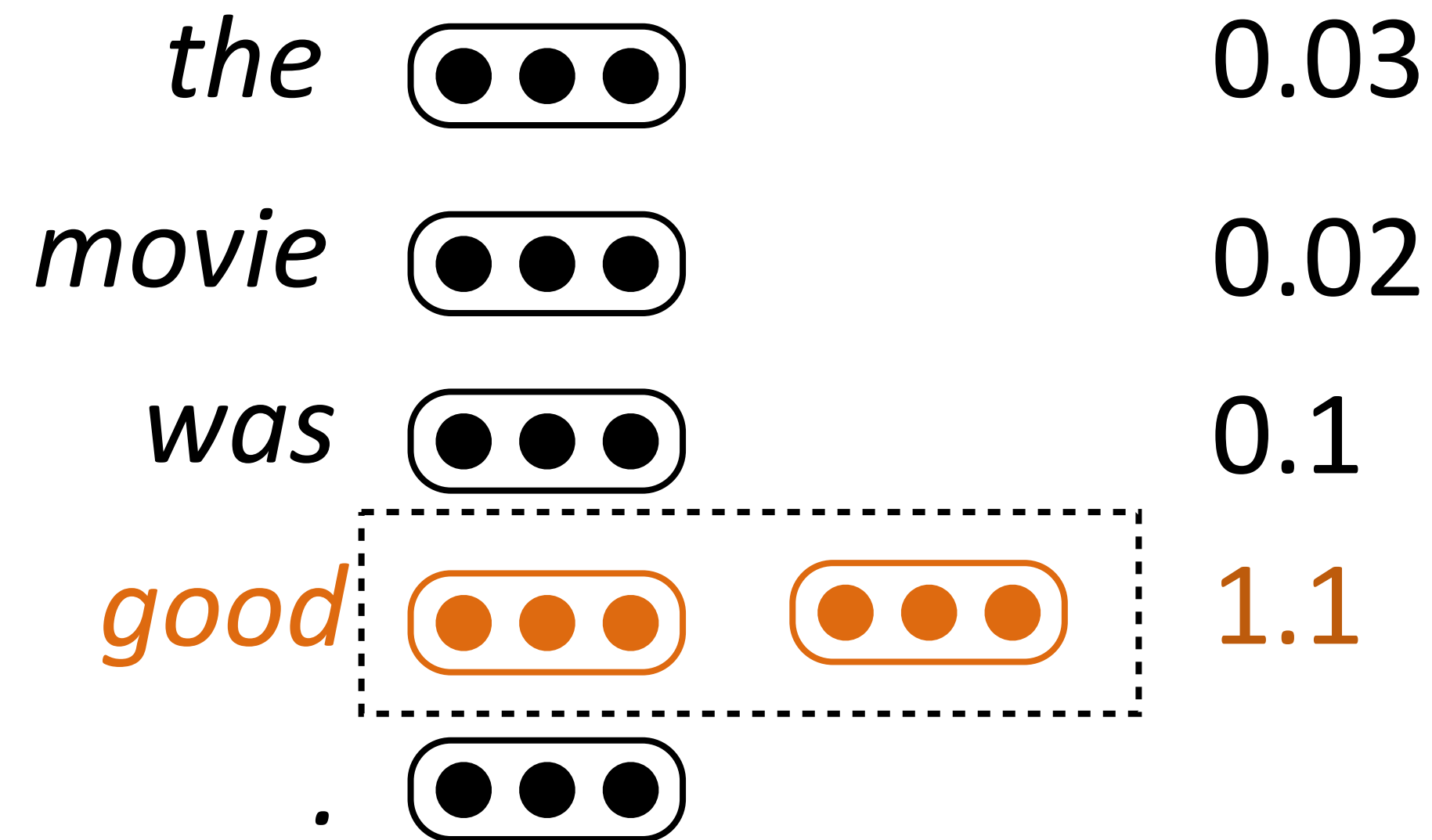
- ▶ Filter “looks like” the things that will cause it to have high activation

Understanding CNNs for Sentiment



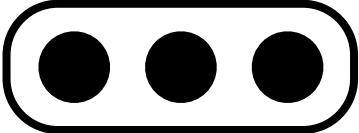
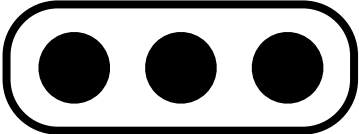

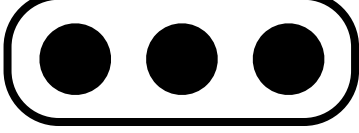
- ▶ Filter “looks like” the things that will cause it to have high activation

Understanding CNNs for Sentiment



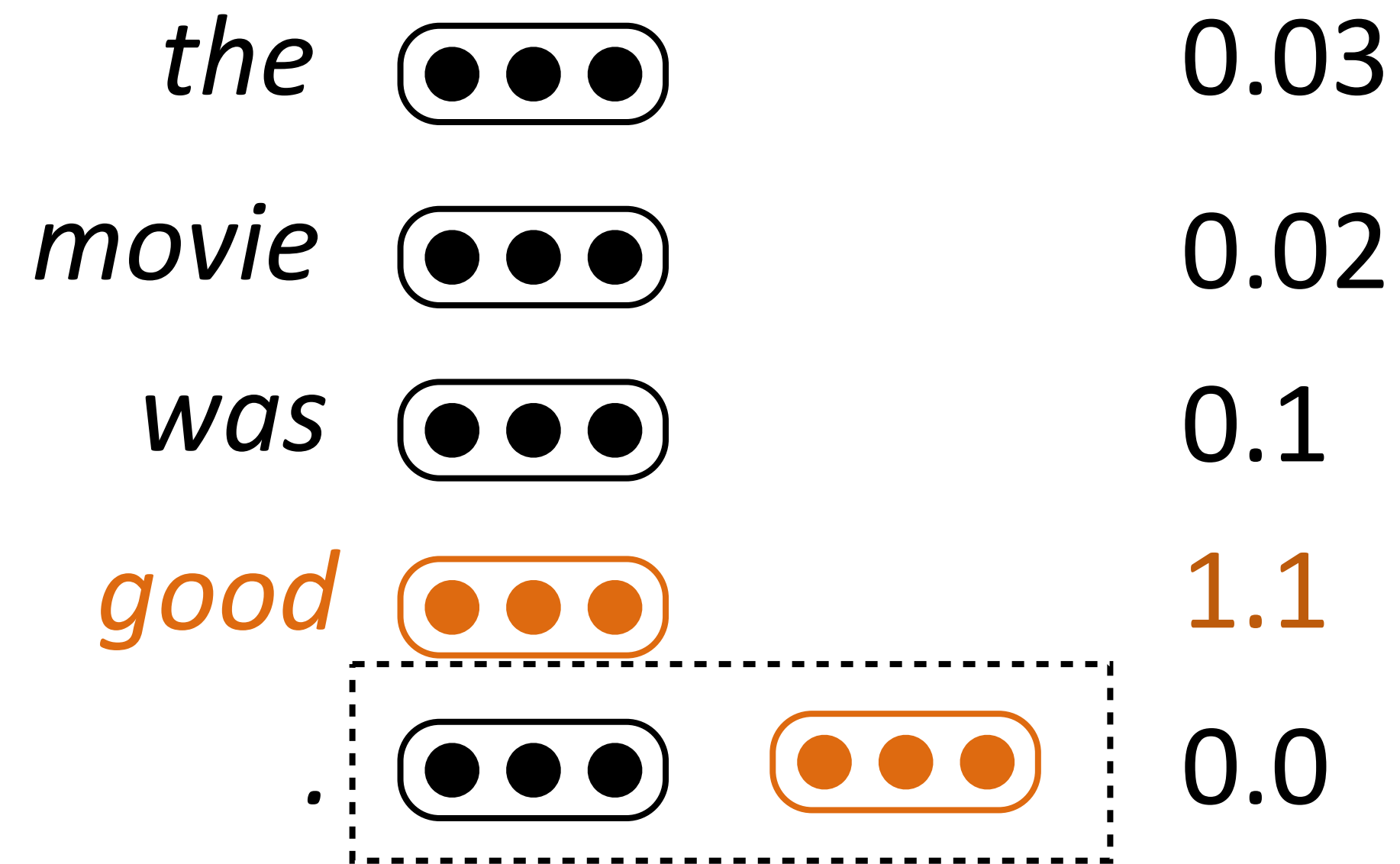
- ▶ Filter “looks like” the things that will cause it to have high activation

Understanding CNNs for Sentiment

<i>the</i>		0.03
<i>movie</i>		0.02
<i>was</i>		0.1
<i>good</i>		1.1
.		

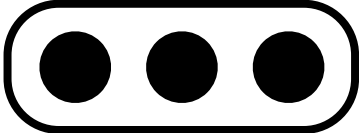
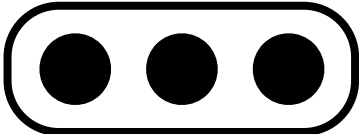


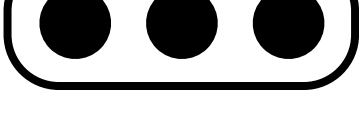
- ▶ Filter “looks like” the things that will cause it to have high activation

Understanding CNNs for Sentiment



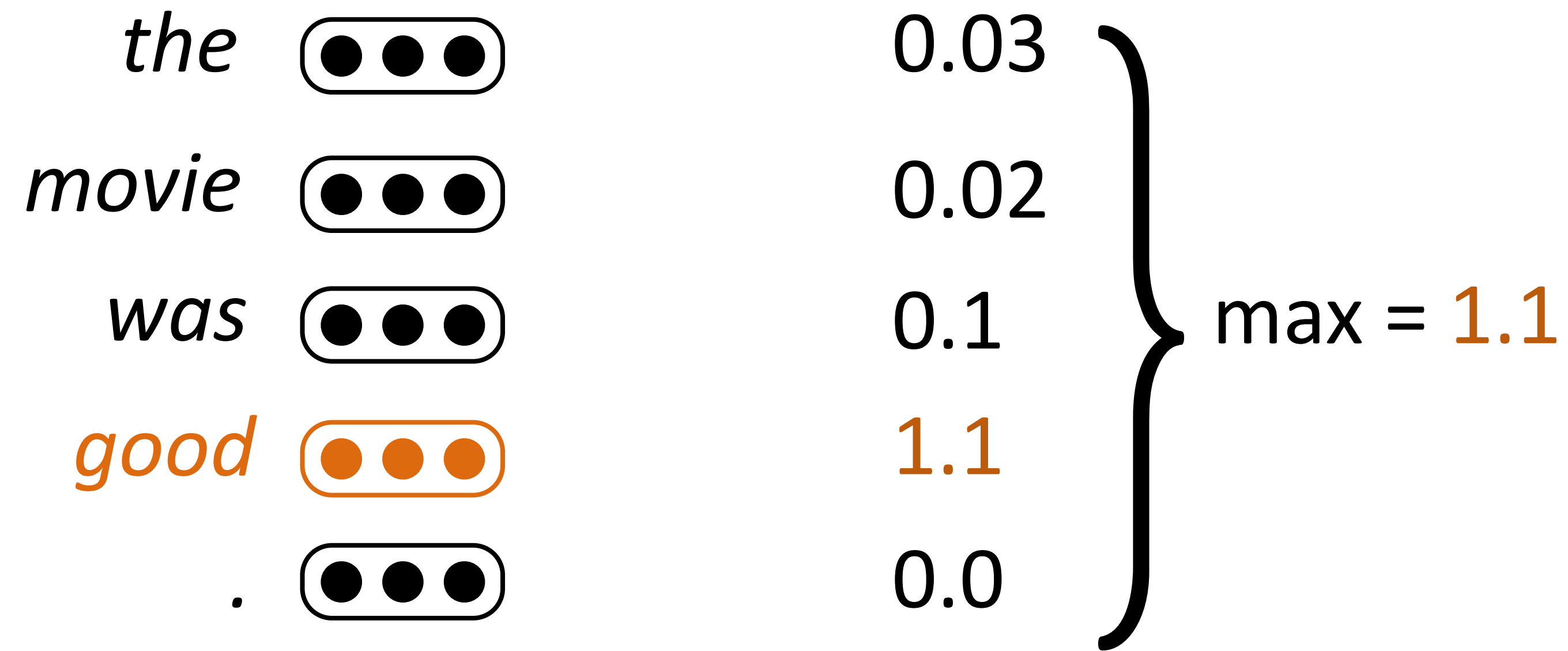
- ▶ Filter “looks like” the things that will cause it to have high activation

Understanding CNNs for Sentiment

<i>the</i>		0.03
<i>movie</i>		0.02
<i>was</i>		0.1
<i>good</i>		1.1
.		0.0

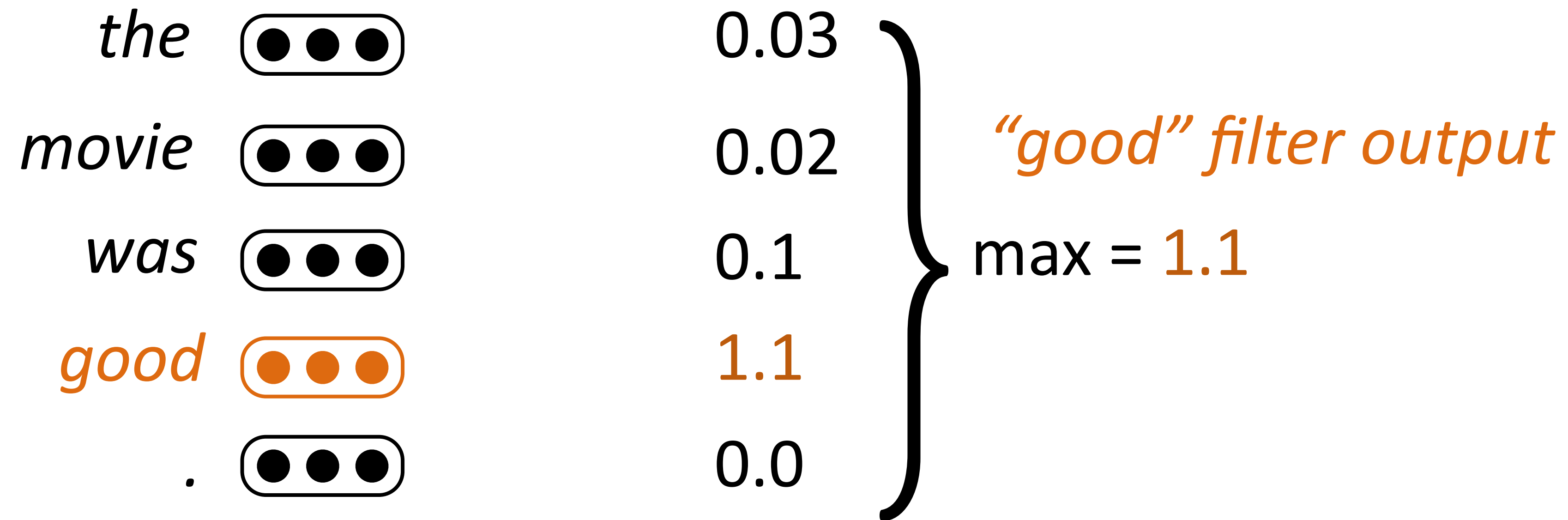
- ▶ Filter “looks like” the things that will cause it to have high activation

Understanding CNNs for Sentiment



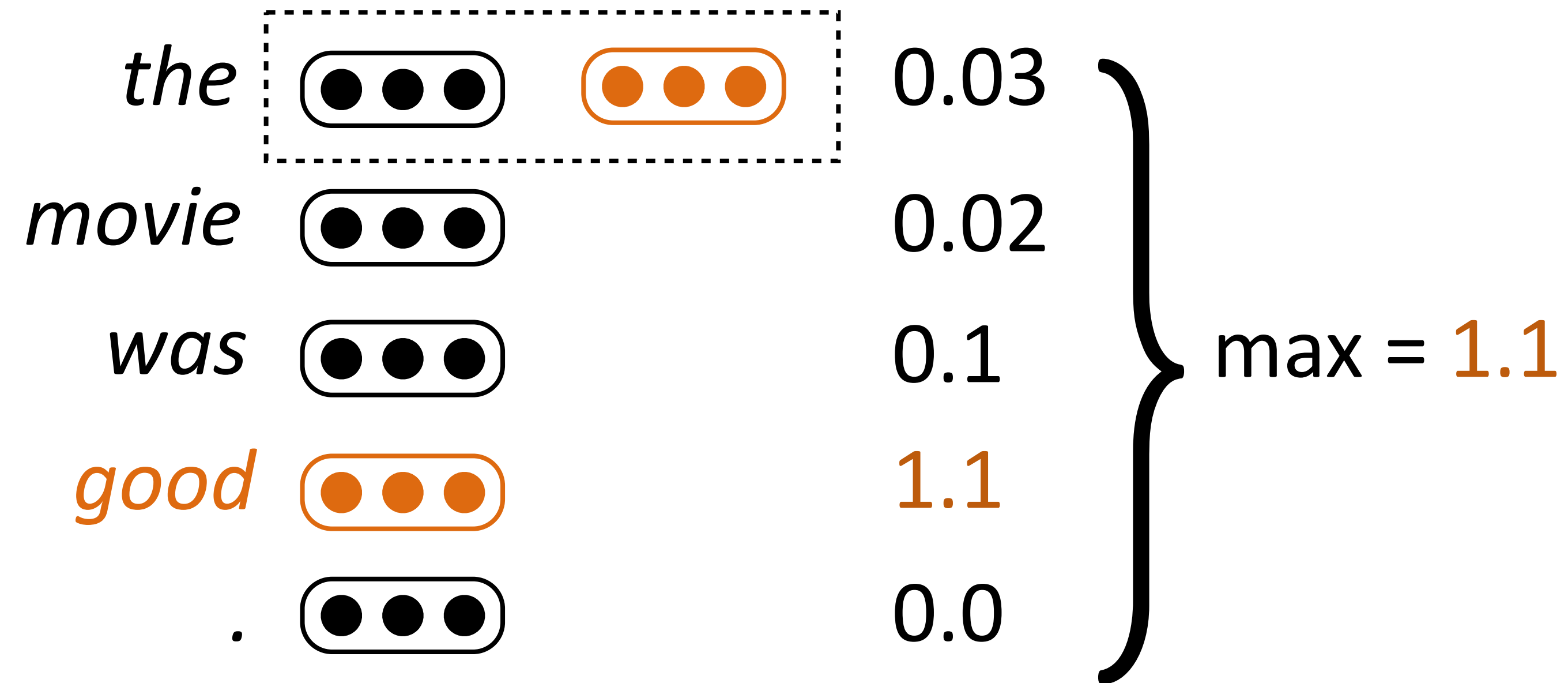
- ▶ Filter “looks like” the things that will cause it to have high activation

Understanding CNNs for Sentiment

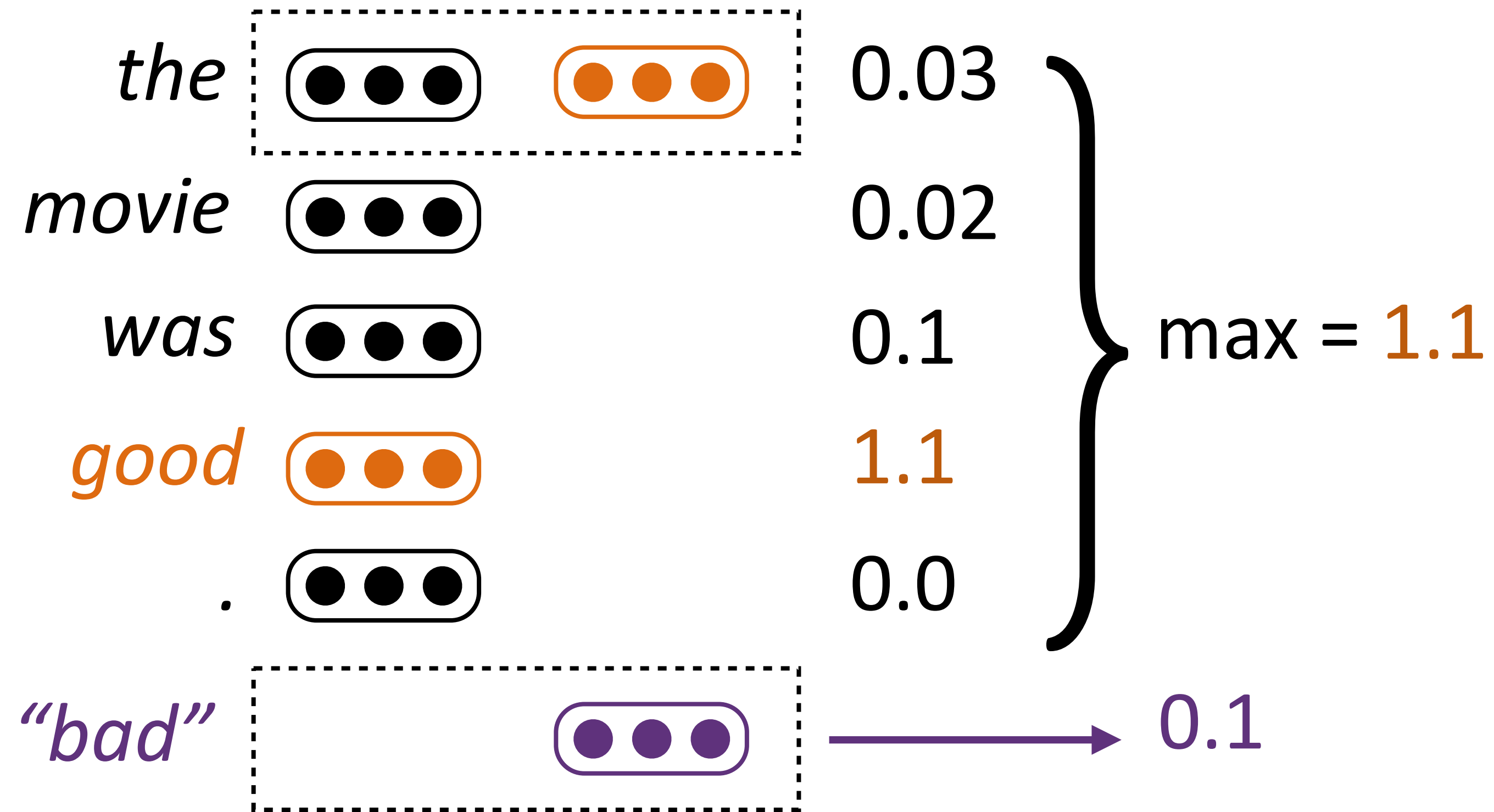


- ▶ Filter “looks like” the things that will cause it to have high activation

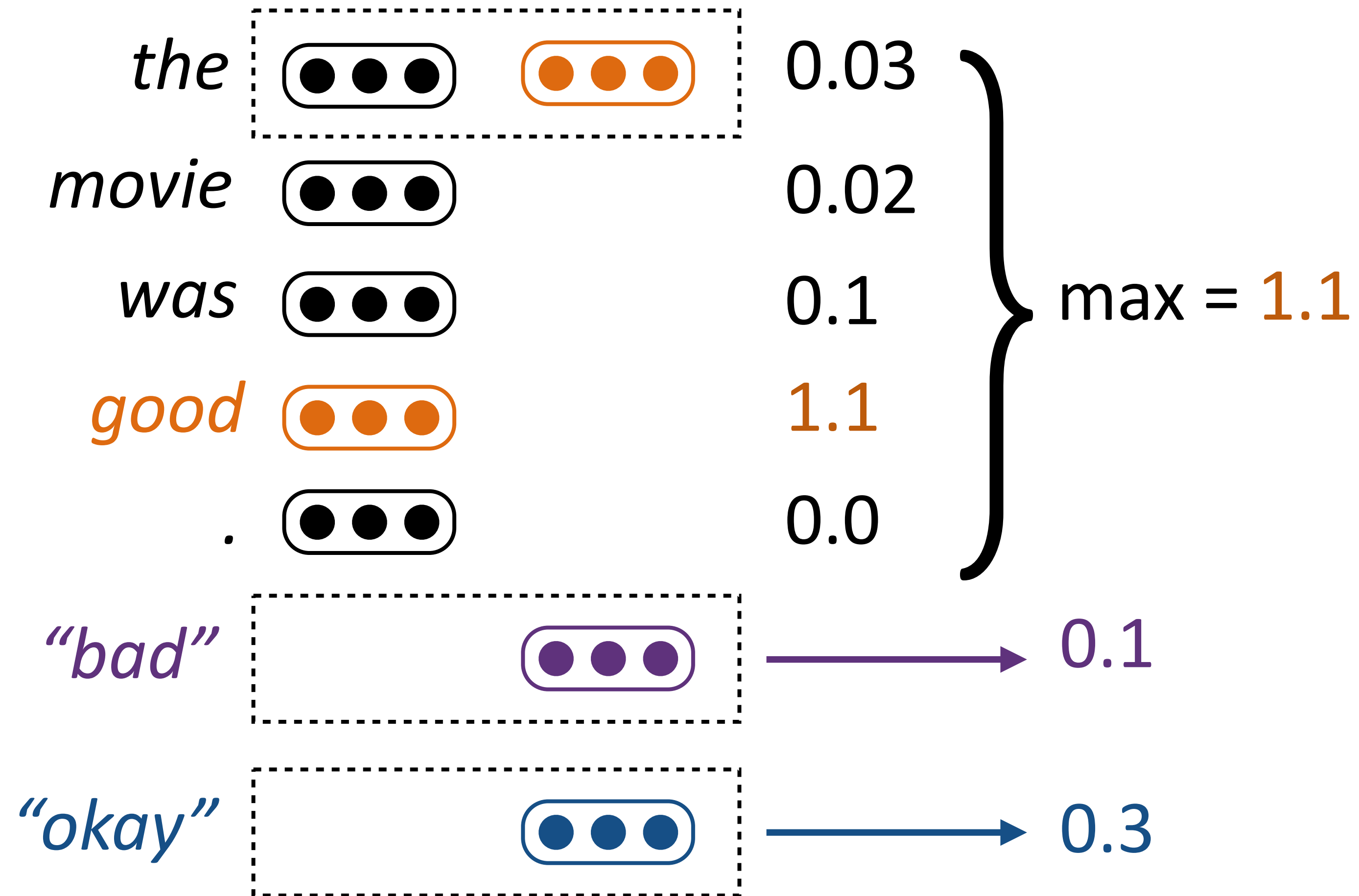
Understanding CNNs for Sentiment



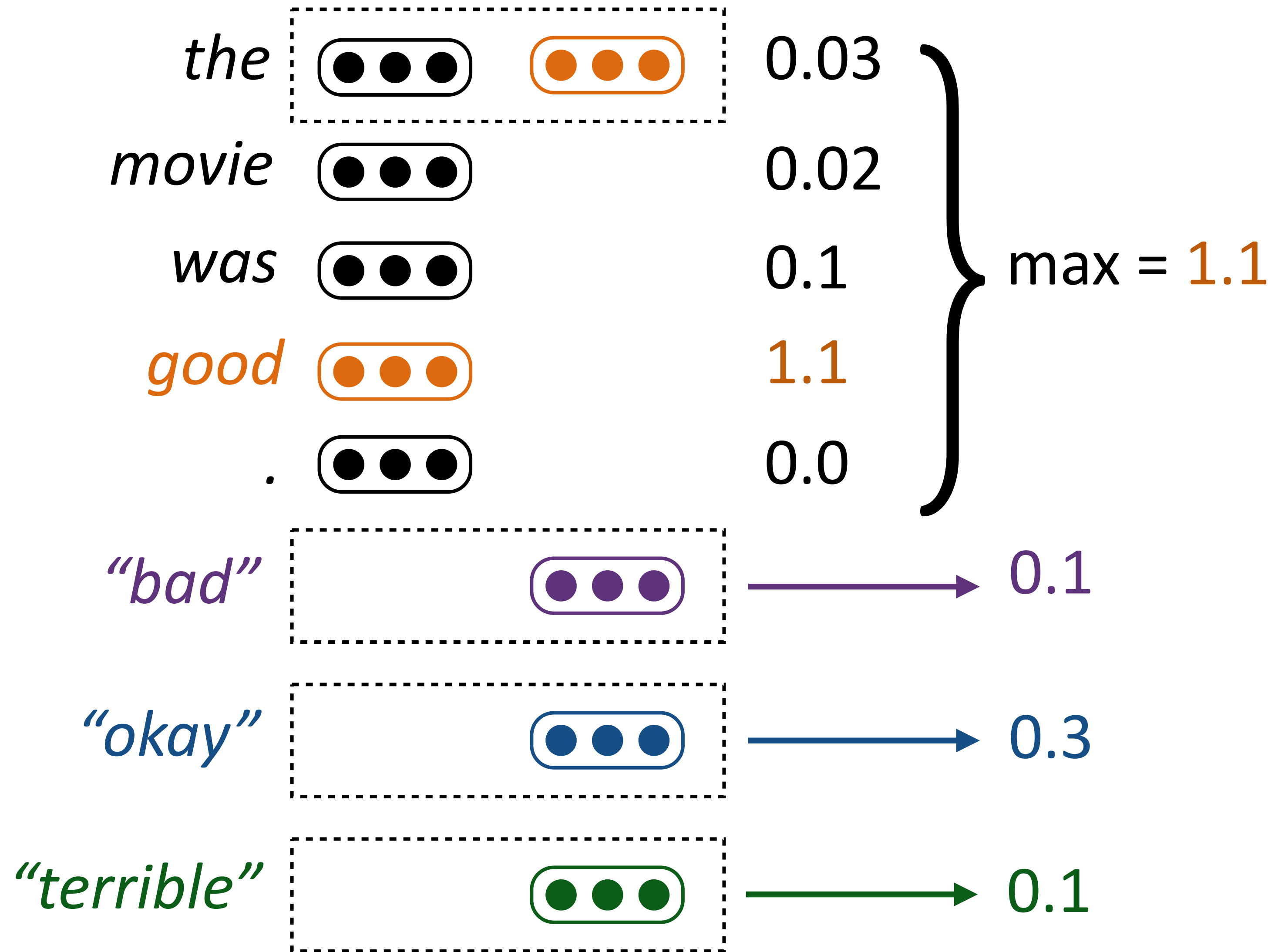
Understanding CNNs for Sentiment



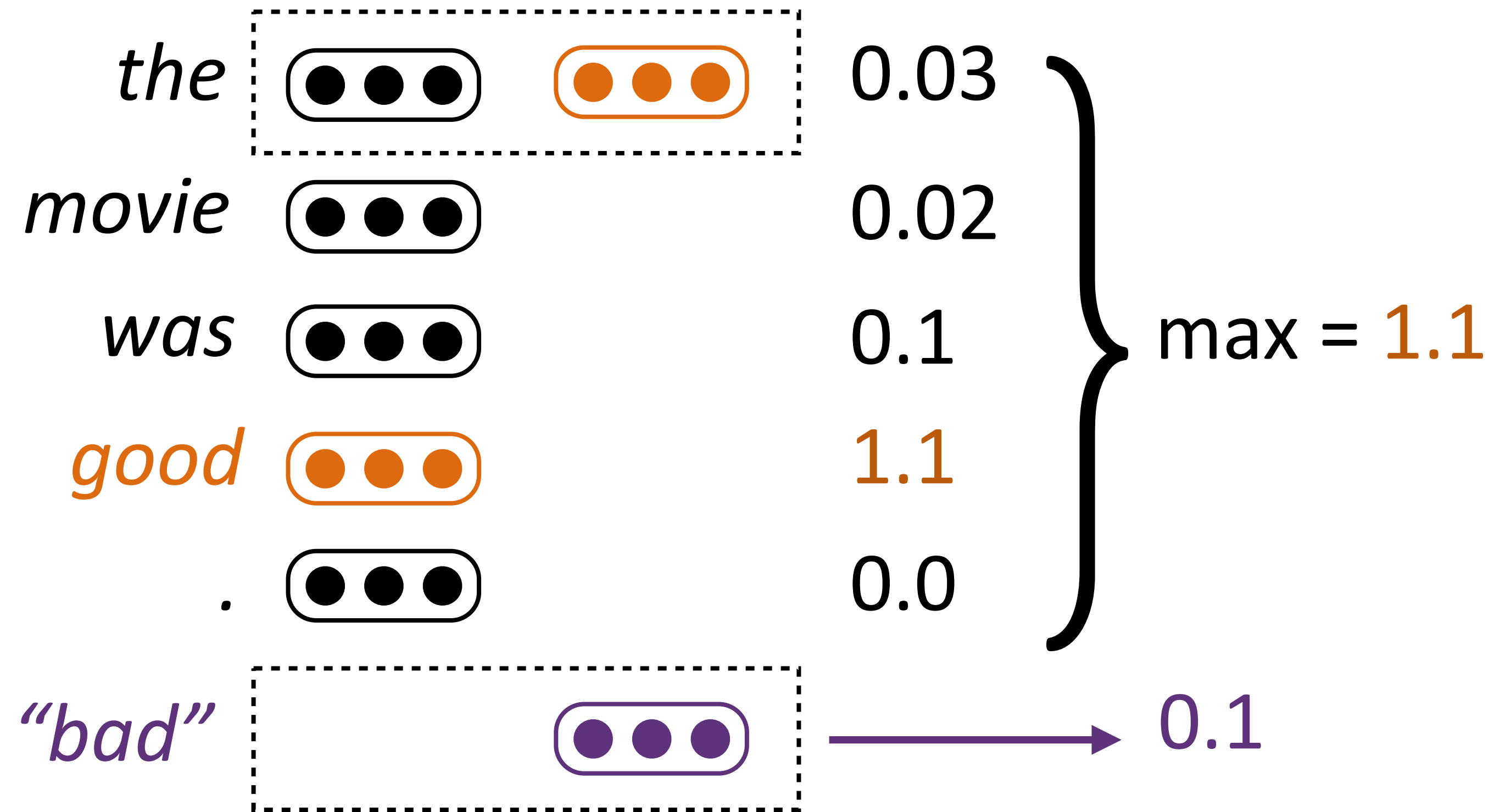
Understanding CNNs for Sentiment



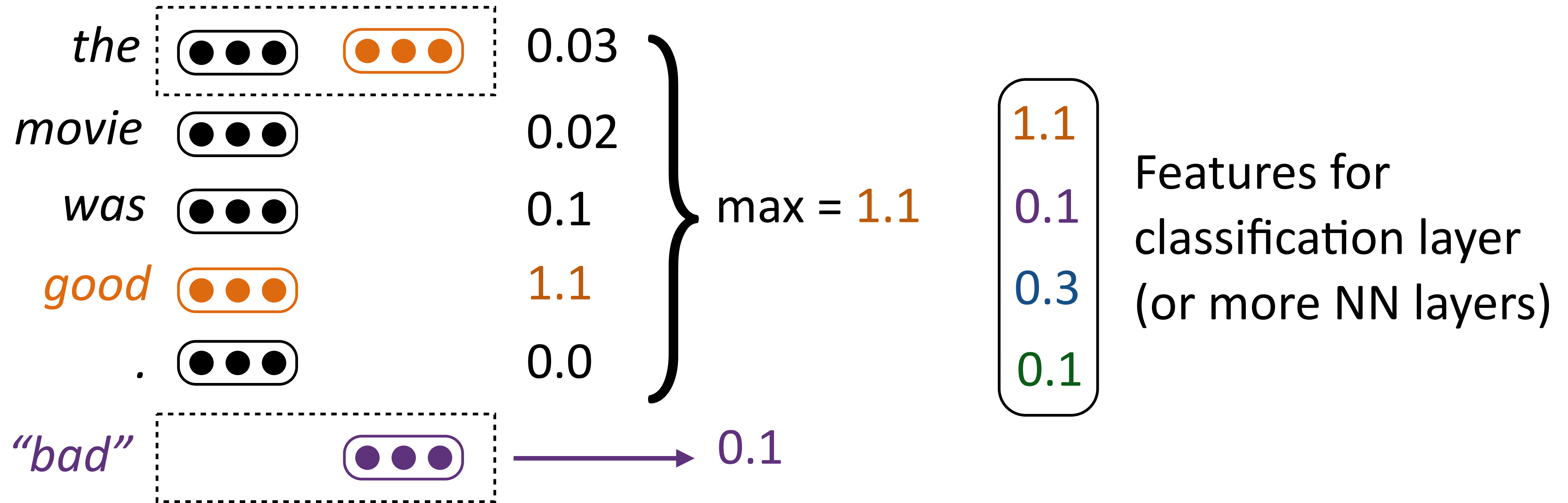
Understanding CNNs for Sentiment



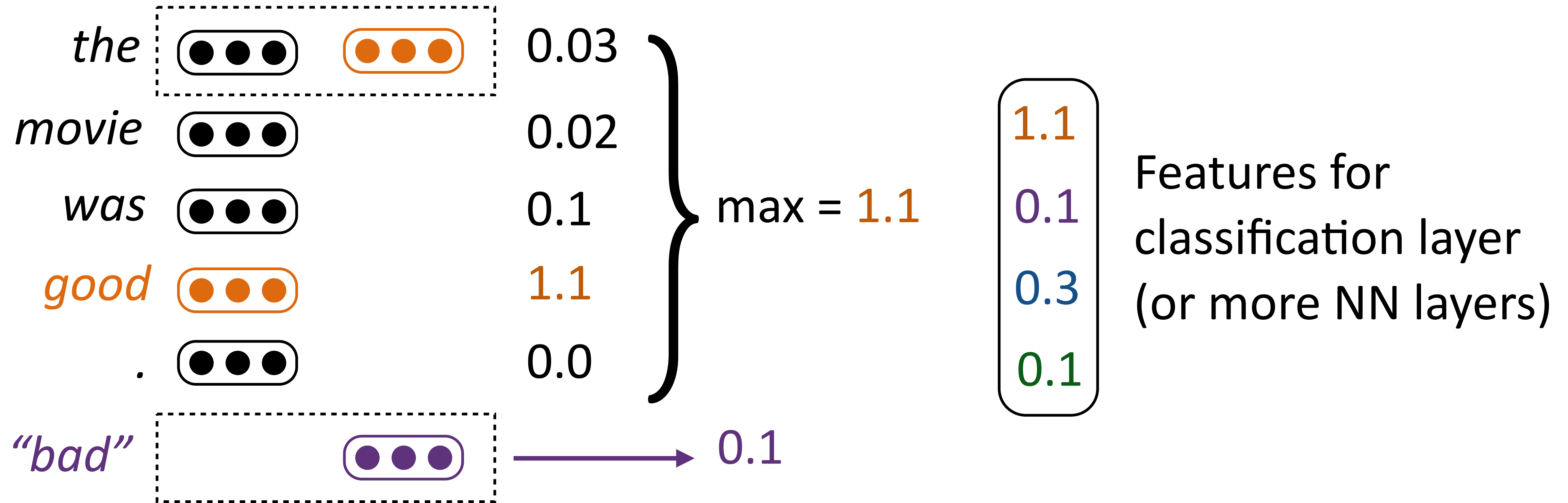
Understanding CNNs for Sentiment



Understanding CNNs for Sentiment

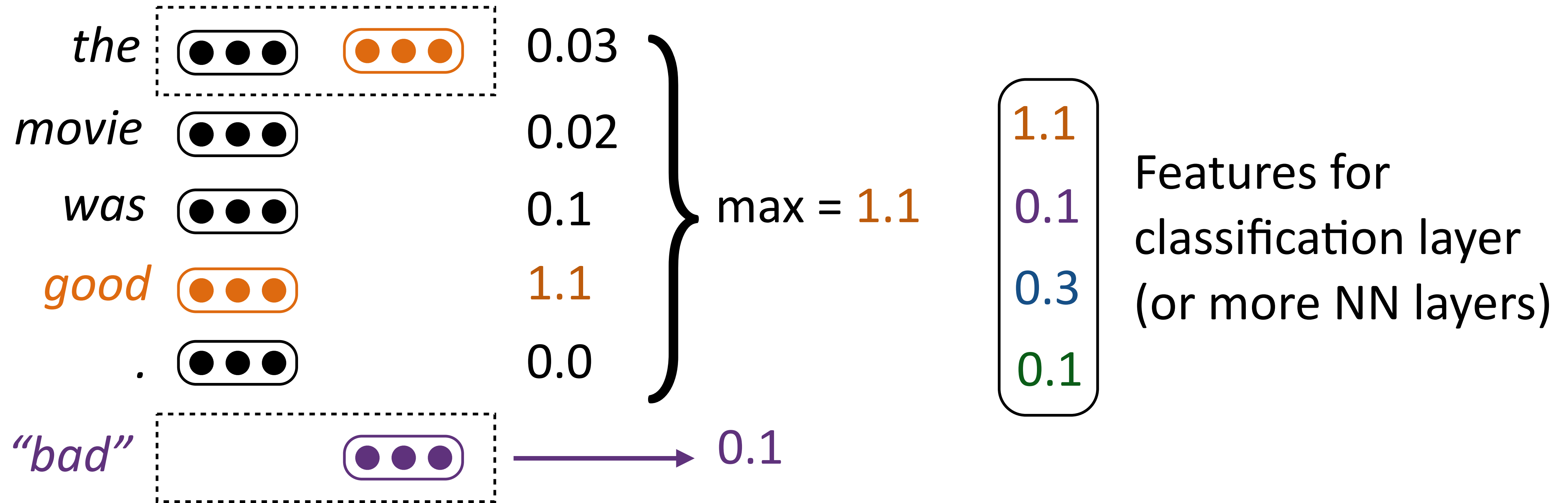


Understanding CNNs for Sentiment



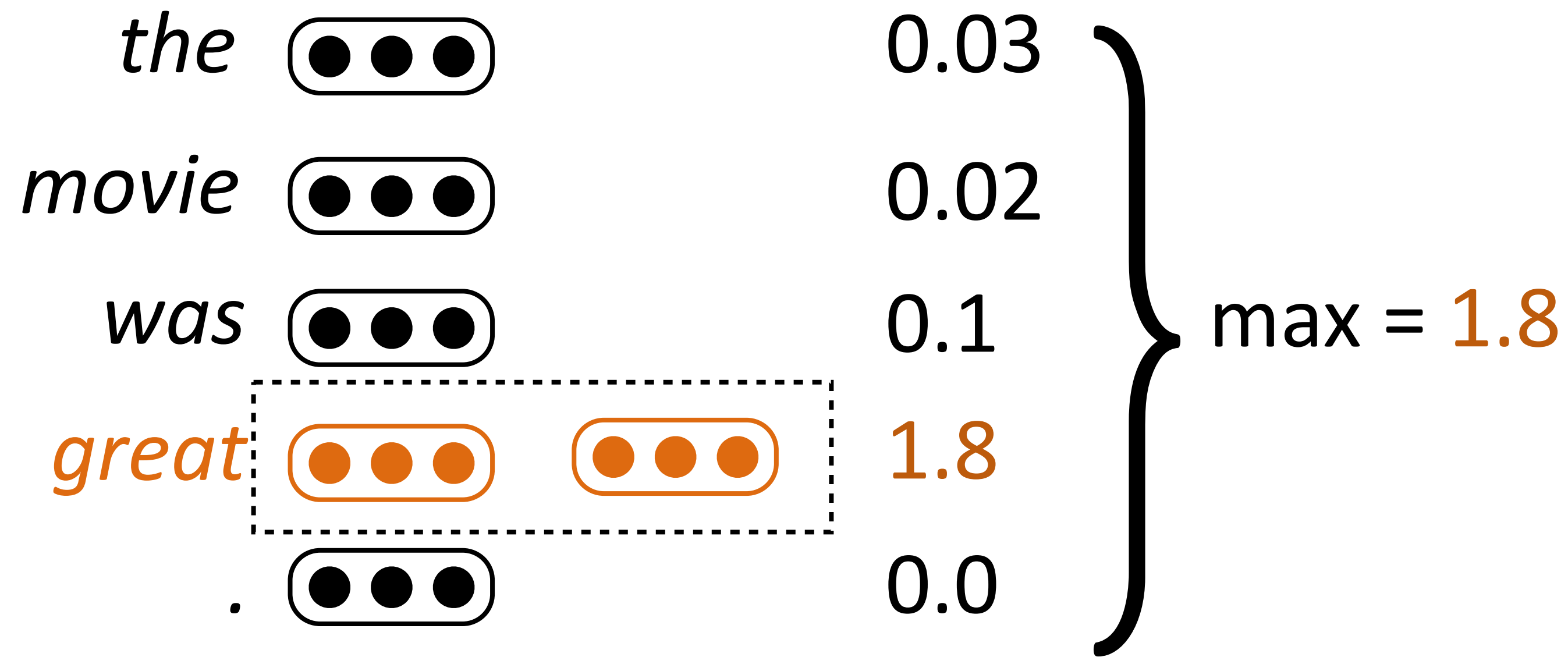
- ▶ Takes variable-length input and turns it into fixed-length output

Understanding CNNs for Sentiment



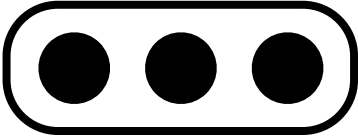
- ▶ Takes variable-length input and turns it into fixed-length output
- ▶ Filters are initialized randomly and then learned

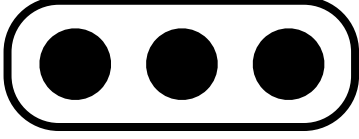
Understanding CNNs for Sentiment



- ▶ Word vectors for similar words are similar, so convolutional filters will have similar outputs

Understanding CNNs for Sentiment

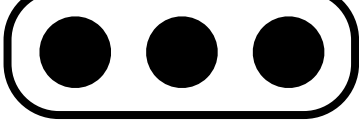
the 

movie 

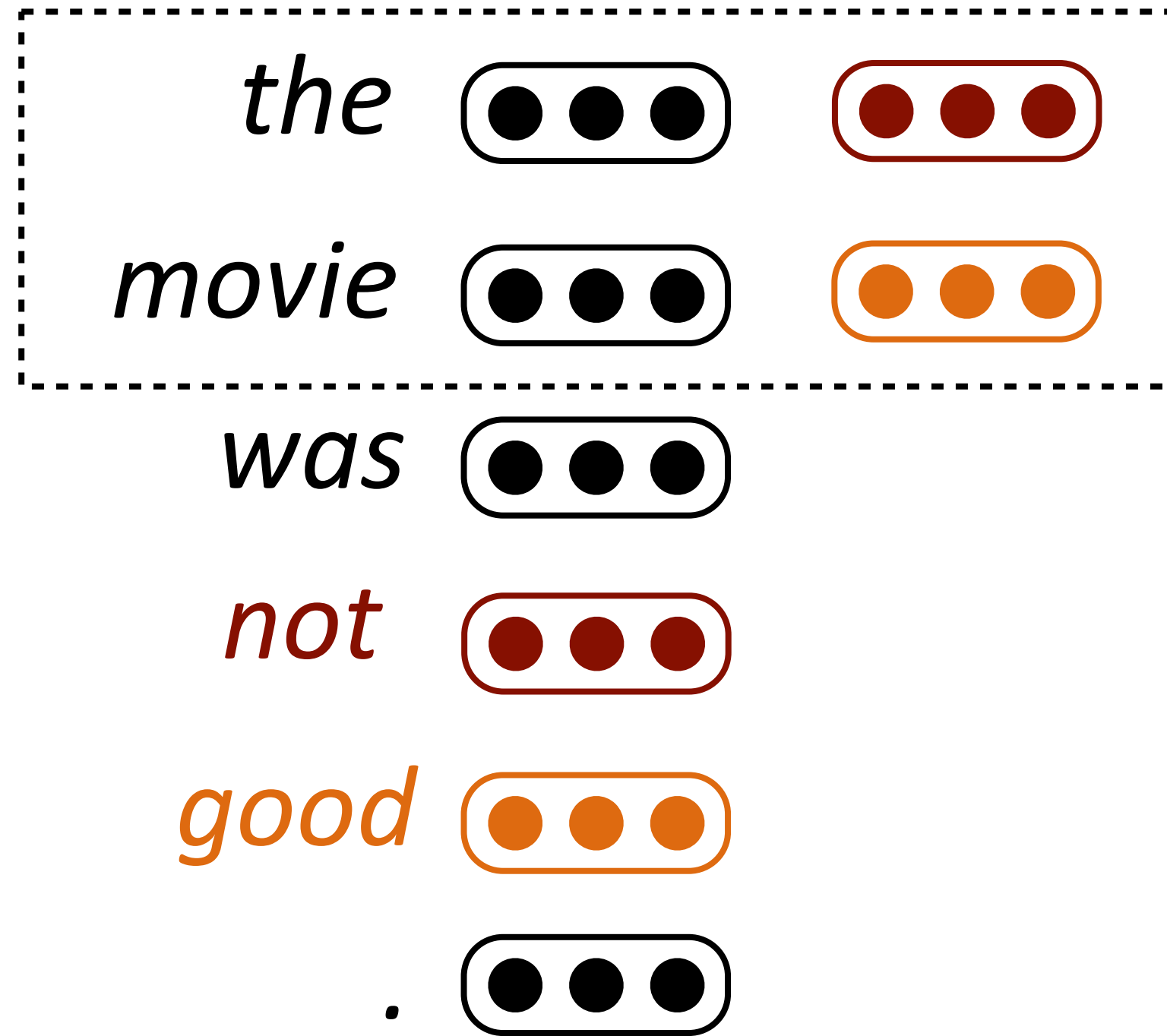
was 

not 

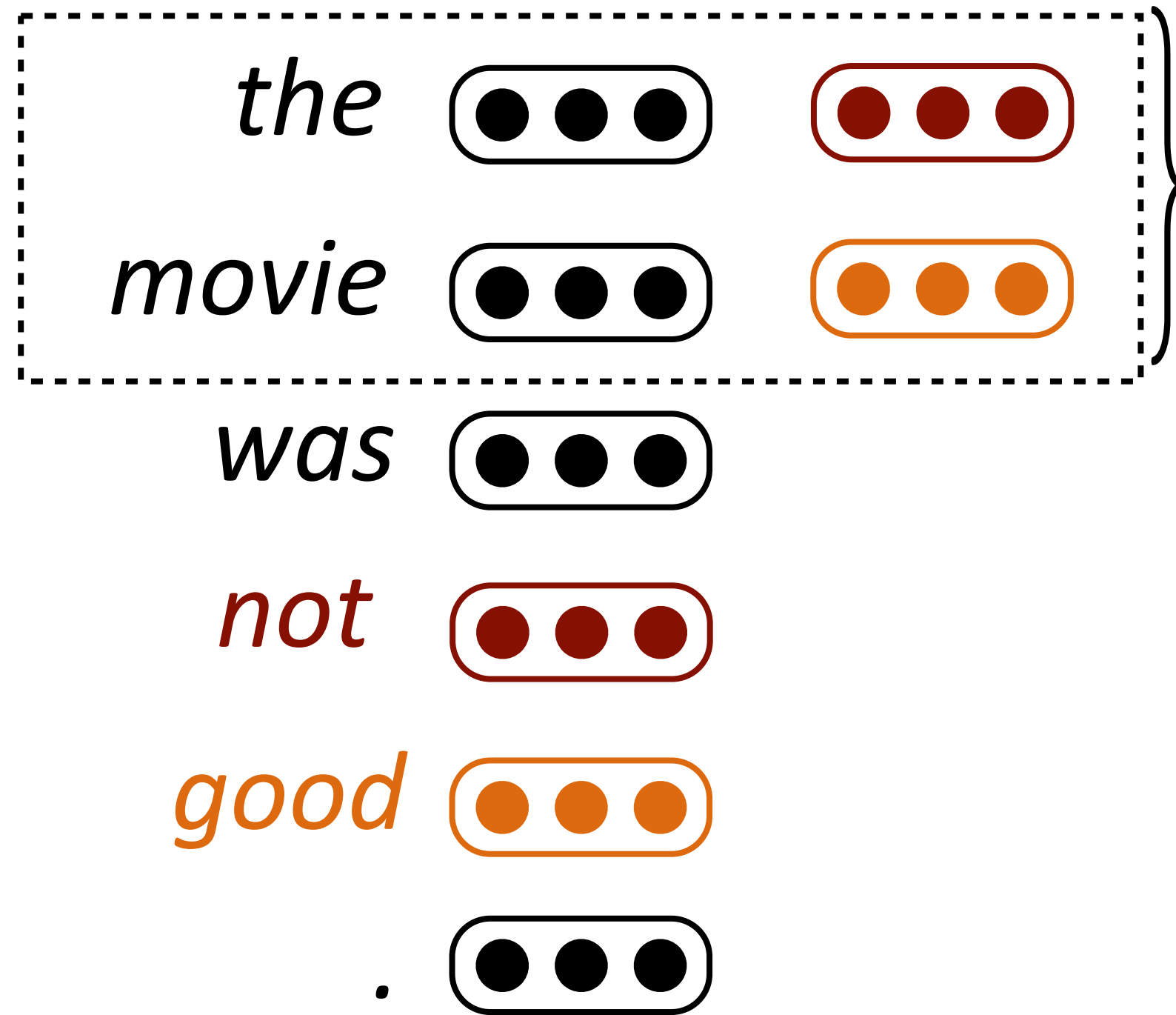
good 

. 

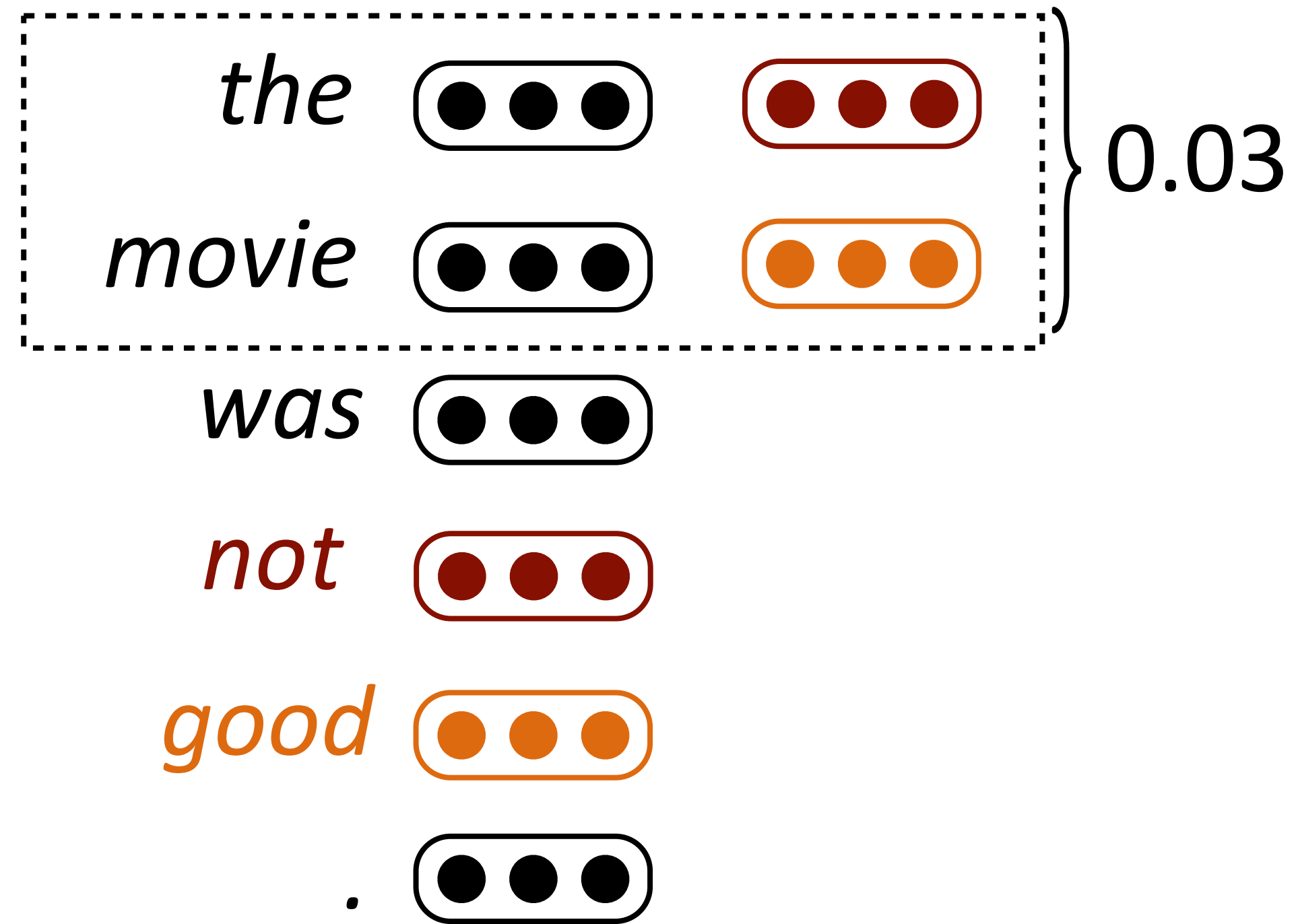
Understanding CNNs for Sentiment



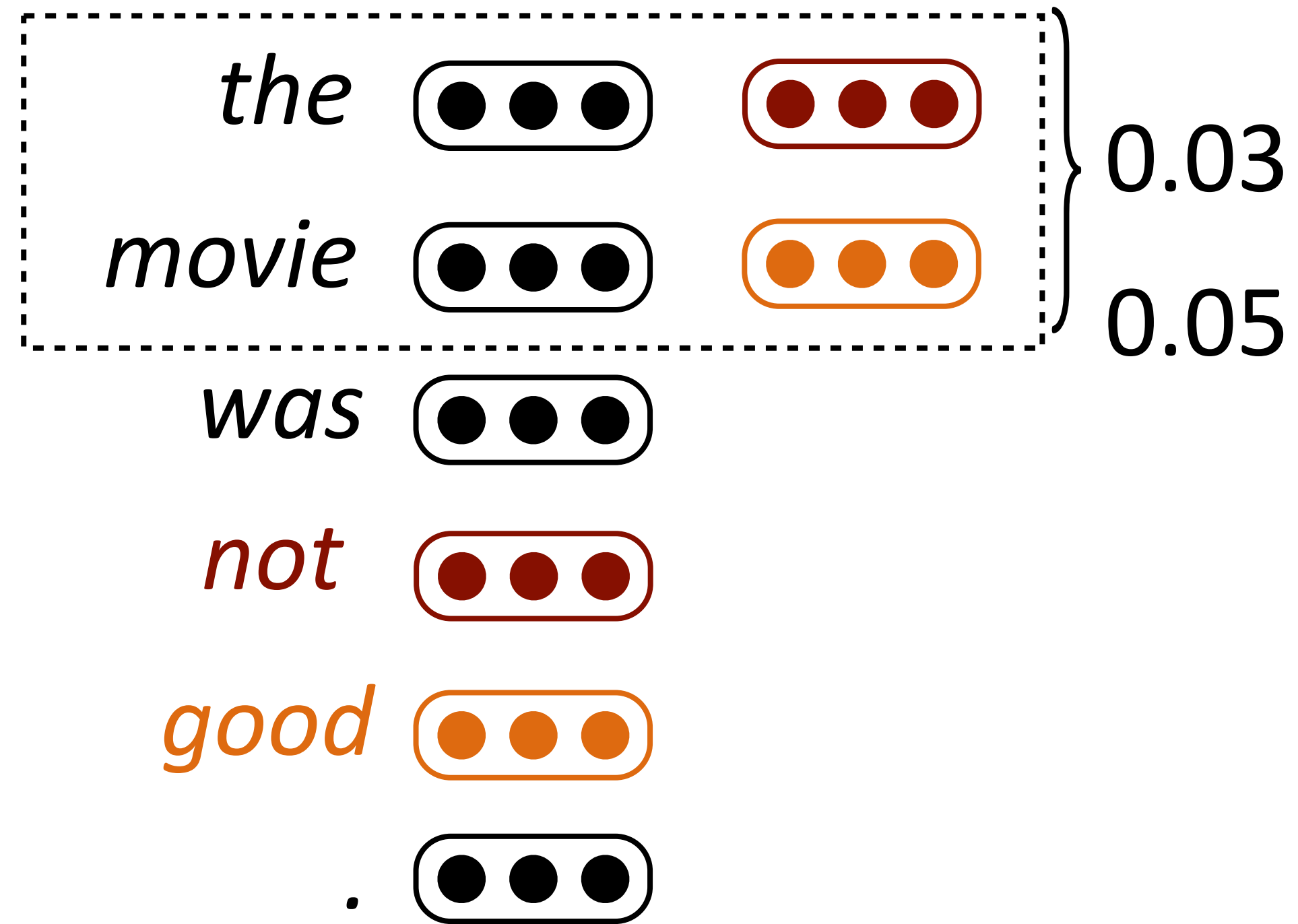
Understanding CNNs for Sentiment



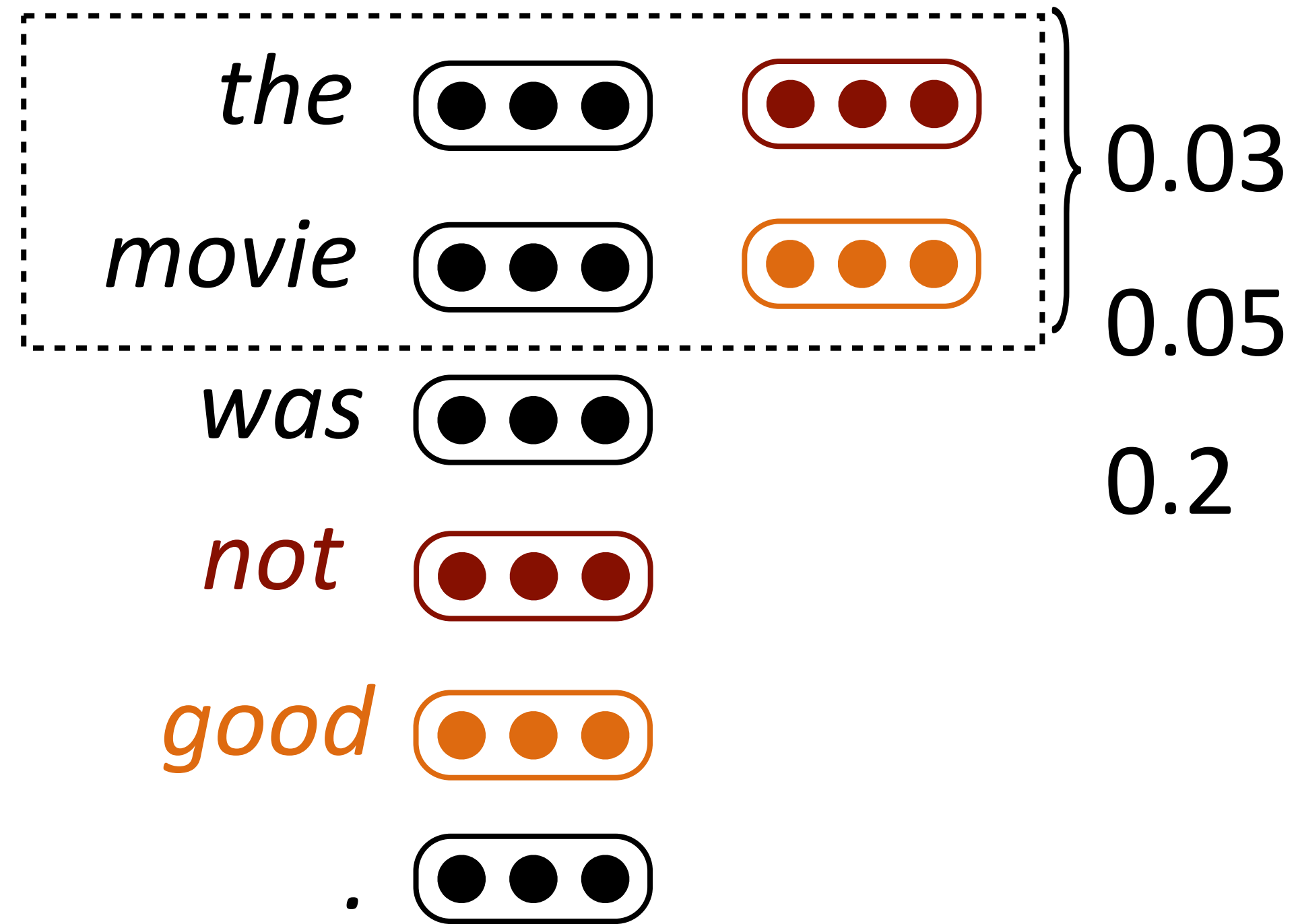
Understanding CNNs for Sentiment



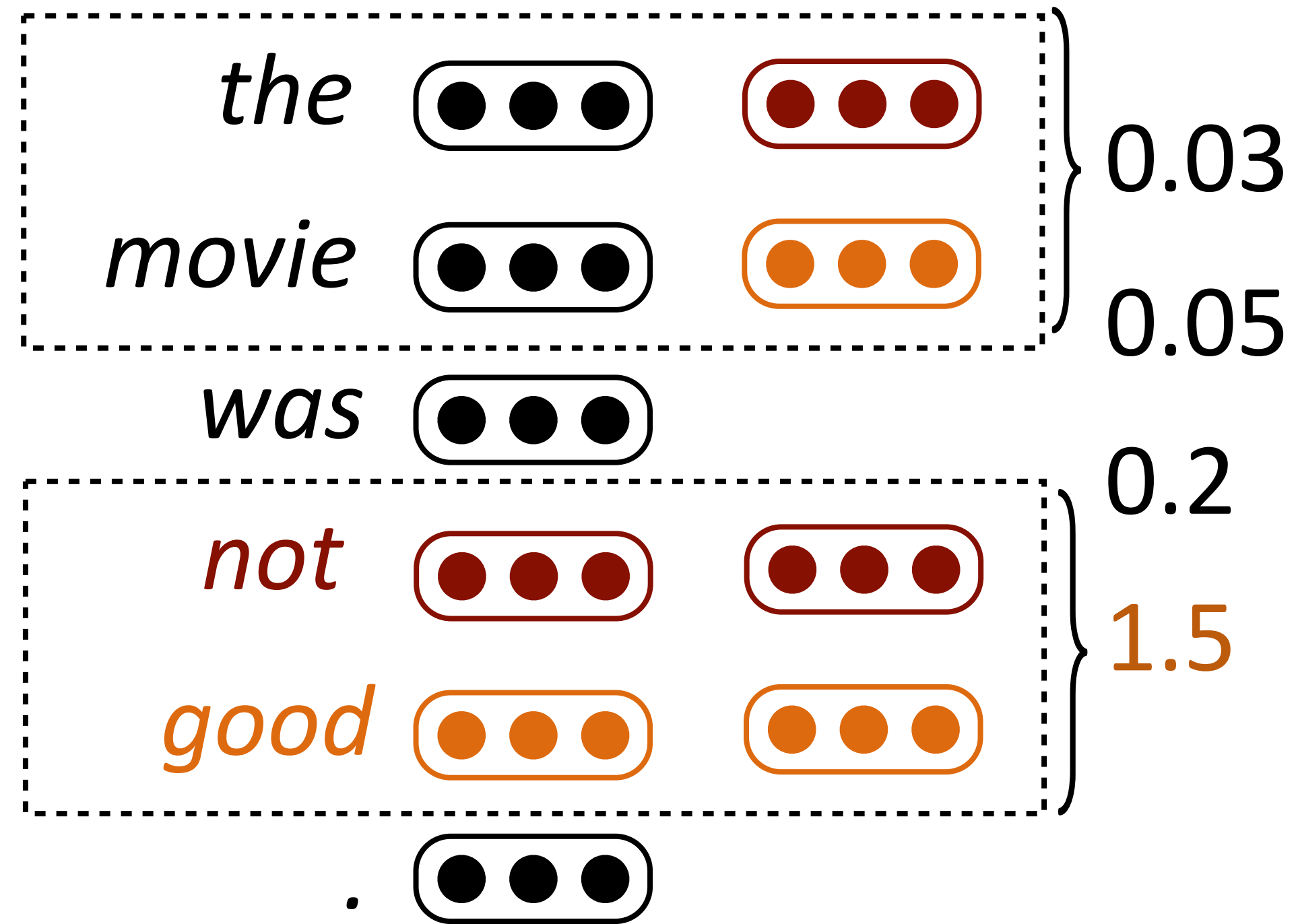
Understanding CNNs for Sentiment



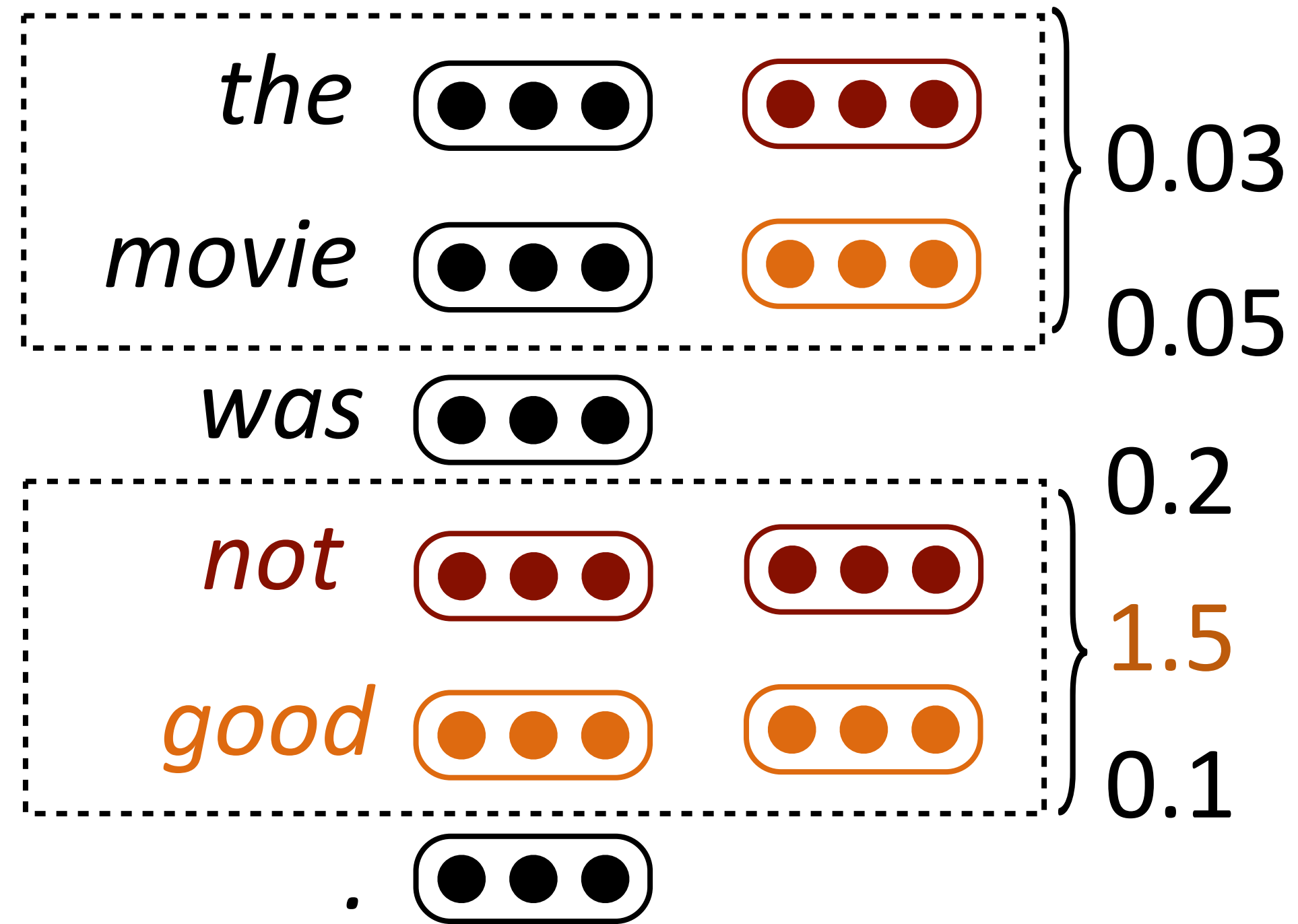
Understanding CNNs for Sentiment



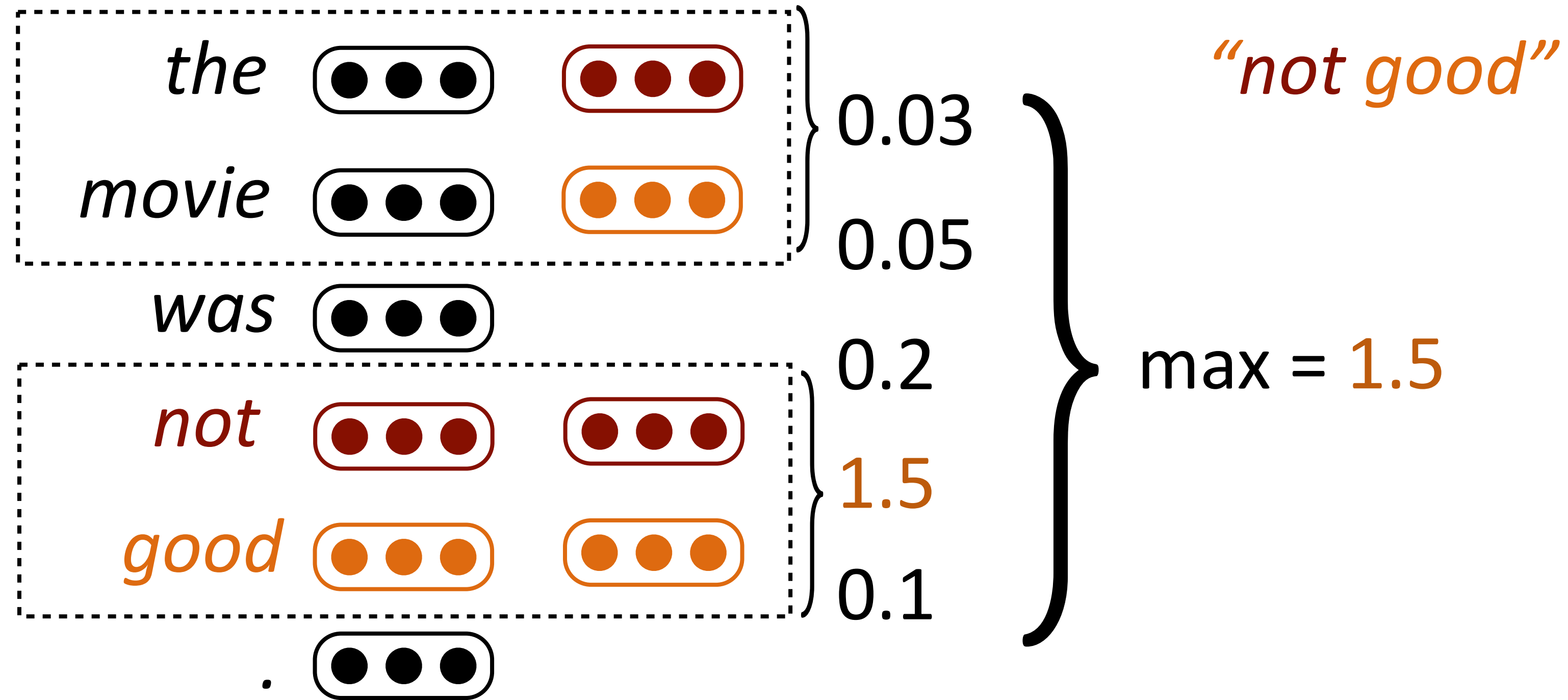
Understanding CNNs for Sentiment



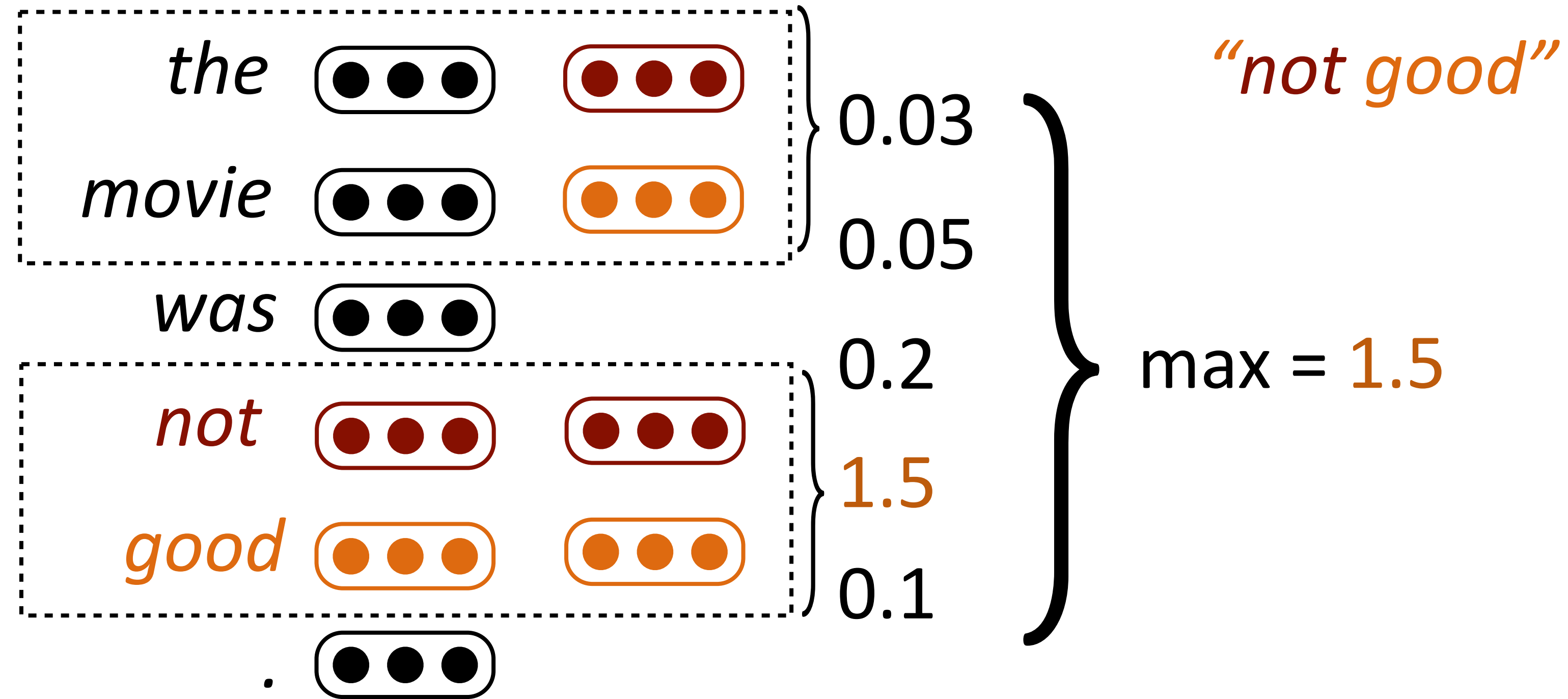
Understanding CNNs for Sentiment



Understanding CNNs for Sentiment

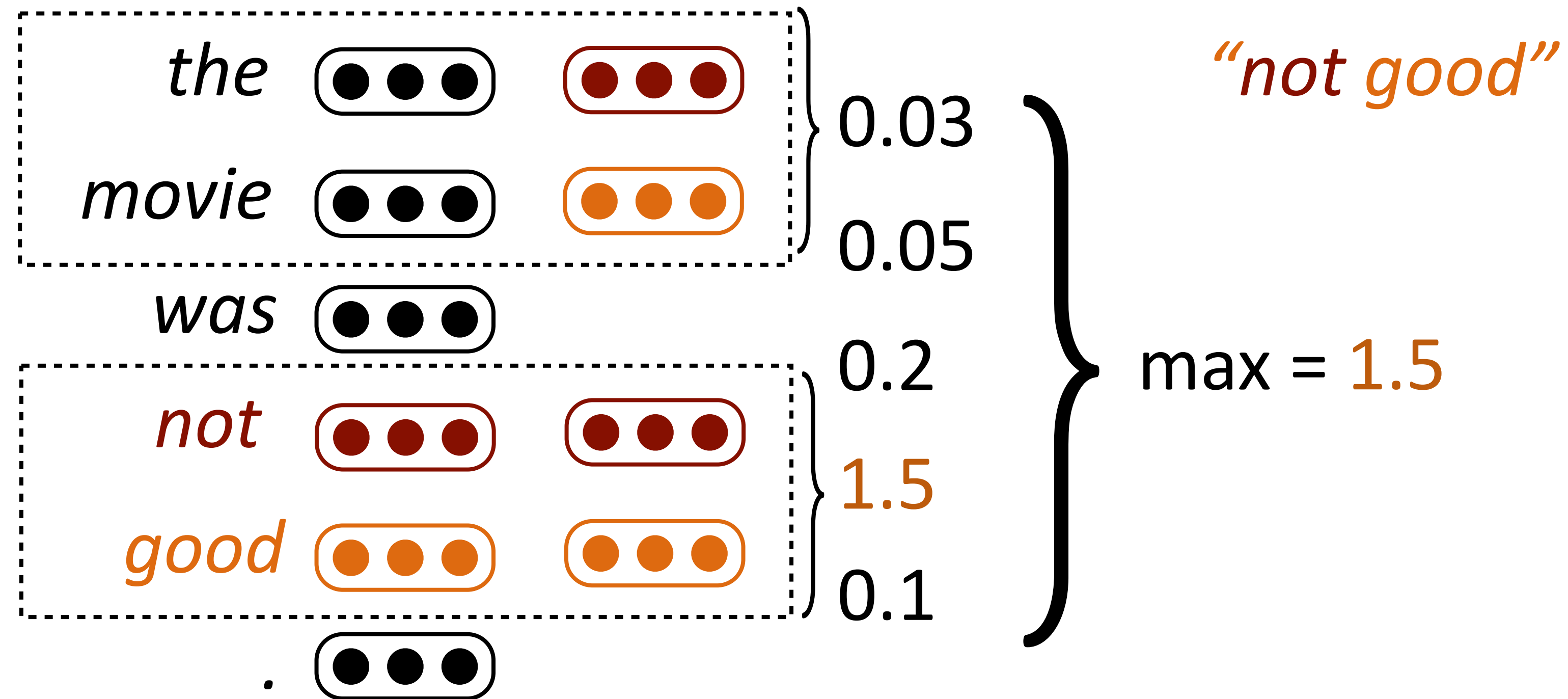


Understanding CNNs for Sentiment



- ▶ Analogous to bigram features in bag-of-words models

Understanding CNNs for Sentiment



- ▶ Analogous to bigram features in bag-of-words models
- ▶ Indicator feature of text containing bigram \leftrightarrow max pooling of a filter that matches that bigram

What can CNNs learn?

the movie was not good

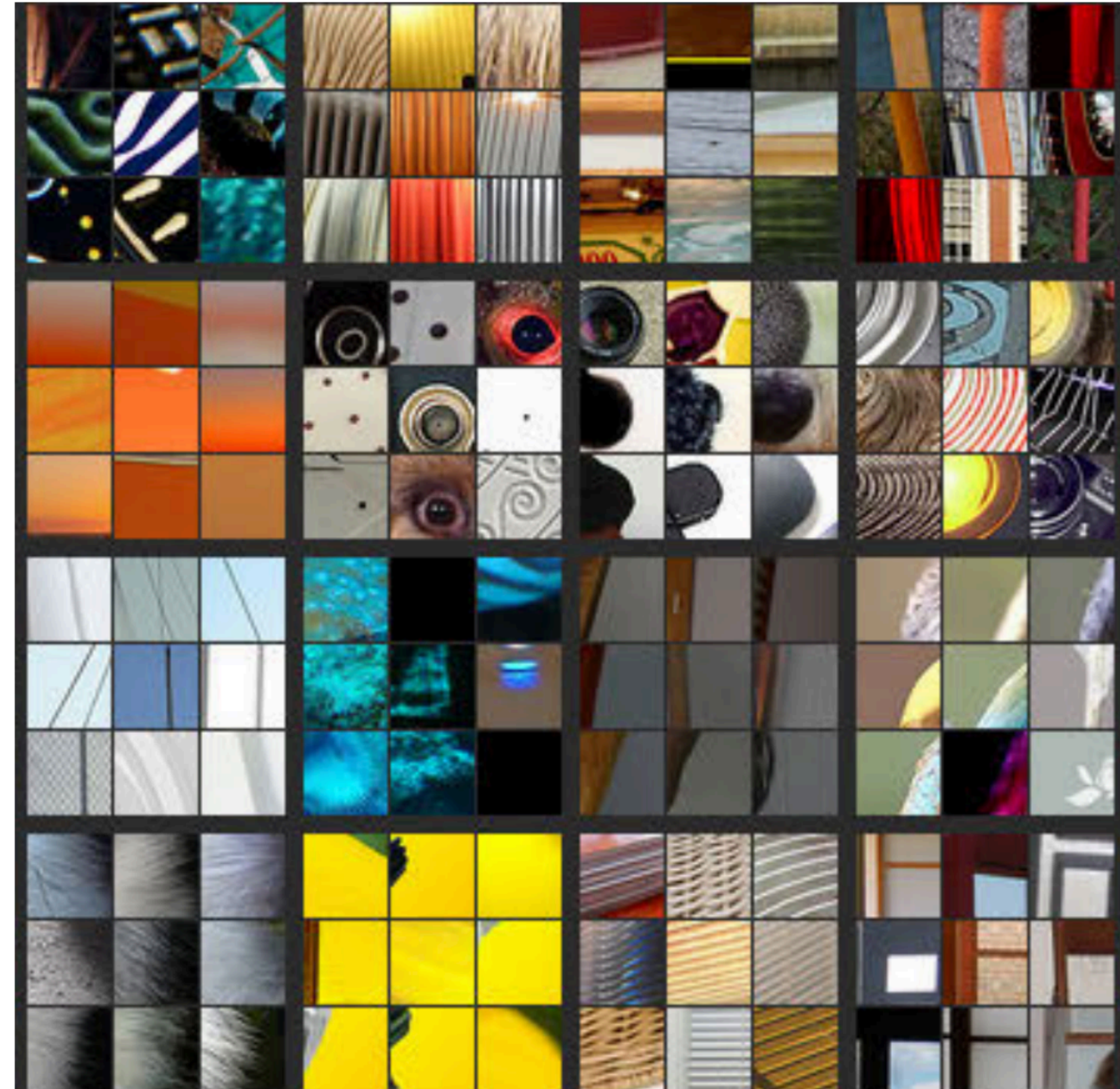
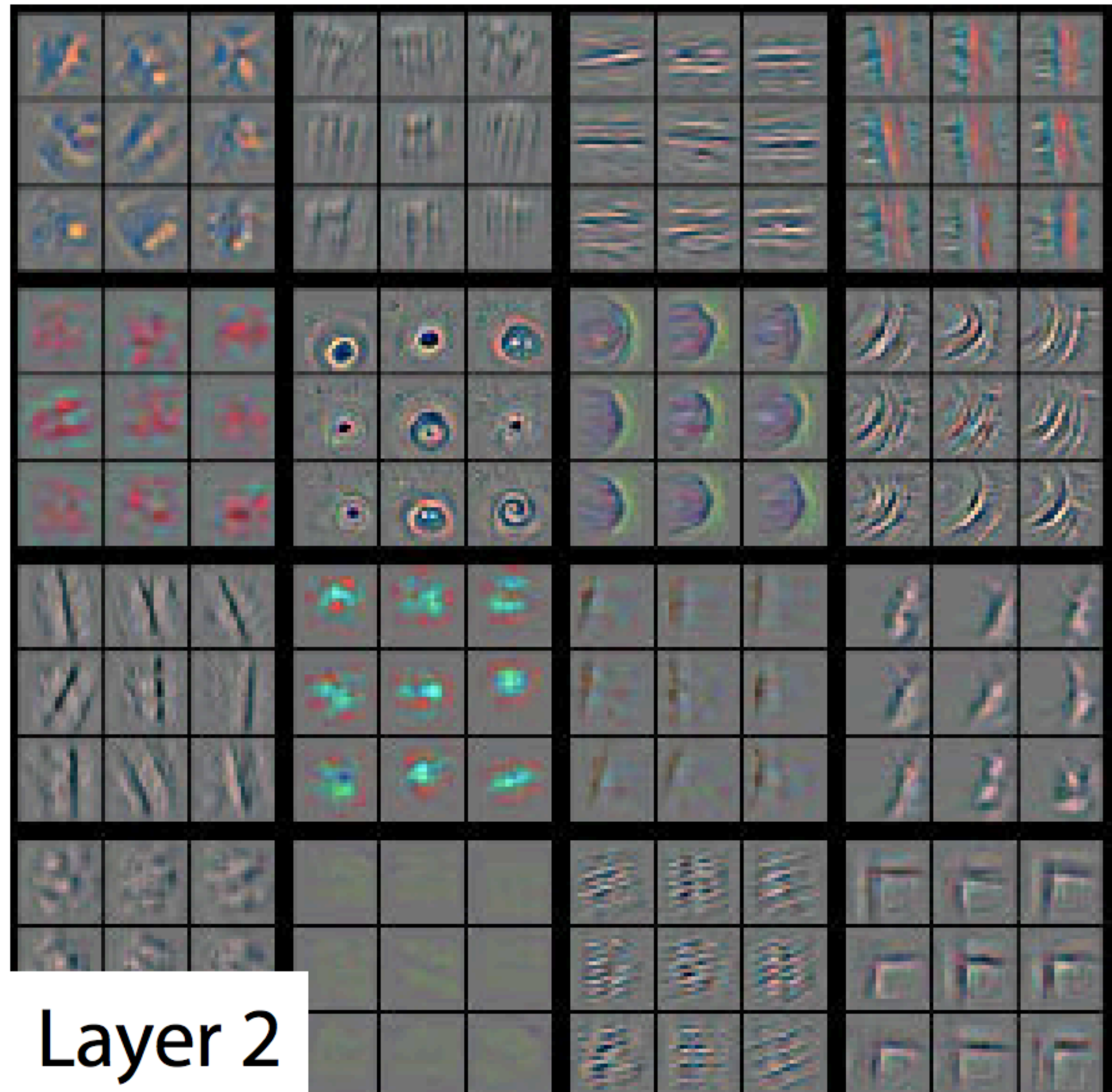
the movie was not really all that good

the cinematography was good, the music great, but the movie was bad

I entered the theater in the bloom of youth and left as an old man

Deep Convolutional Networks

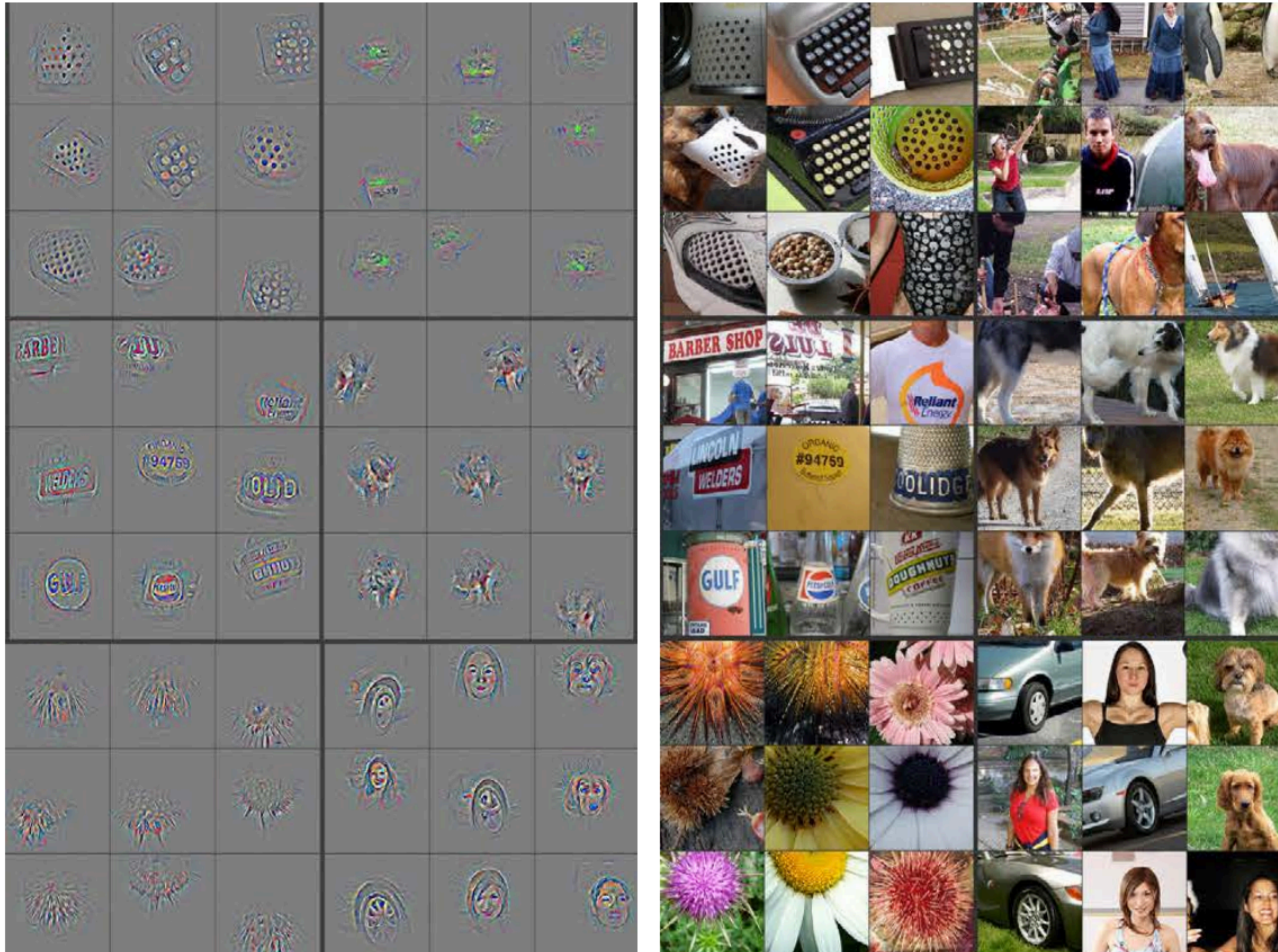
- ▶ Low-level filters: extract low-level features from the data



Zeiler and Fergus (2014)

Deep Convolutional Networks

- ▶ High-level filters: match larger and more “semantic patterns”



Zeiler and Fergus (2014)

CNNs: Implementation

- ▶ Input is $\text{batch_size} \times n \times k$ matrix, filters are $c \times m \times k$ matrix (c filters)

CNNs: Implementation

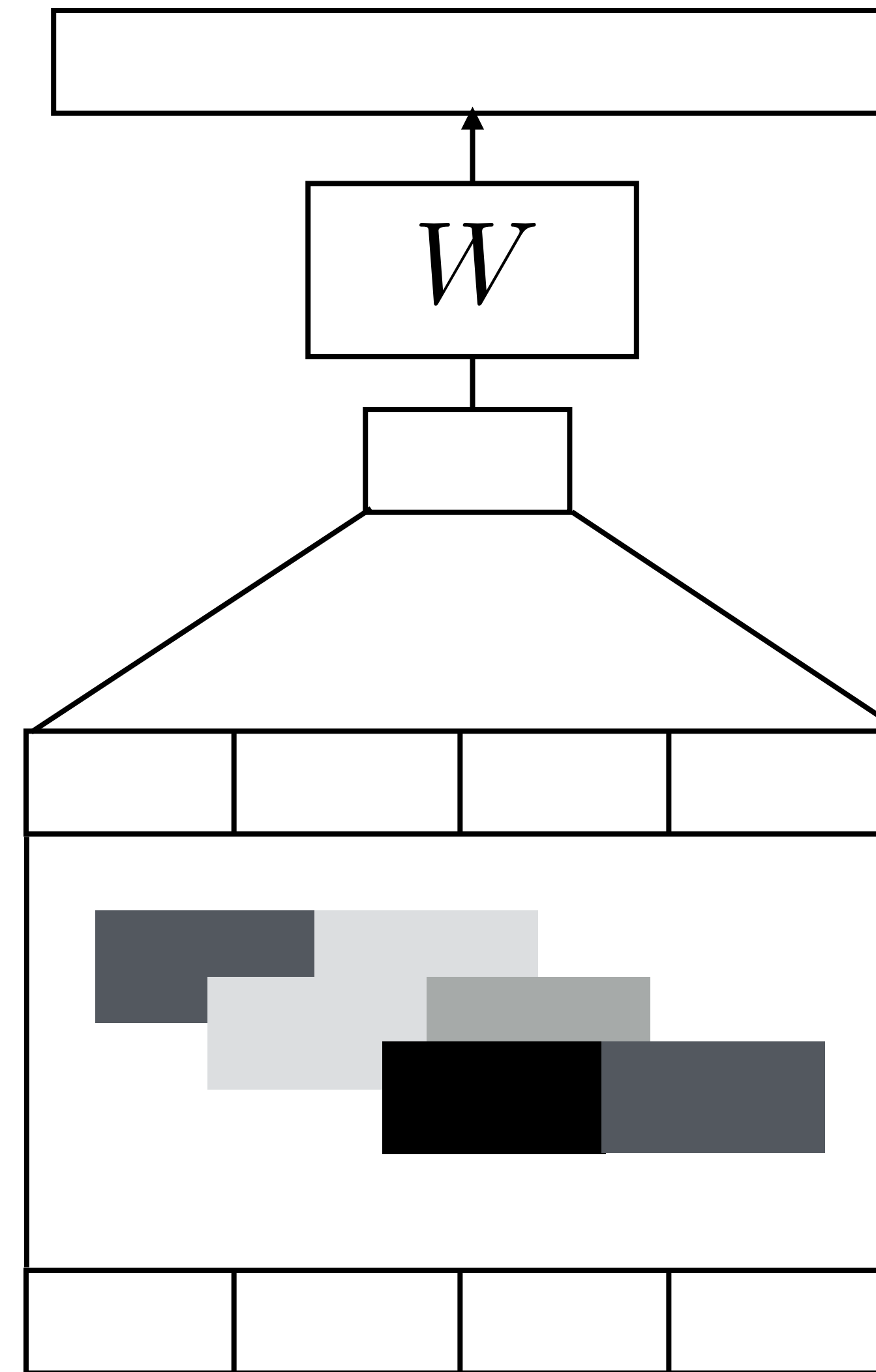
- ▶ Input is $\text{batch_size} \times n \times k$ matrix, filters are $c \times m \times k$ matrix (c filters)
- ▶ Typically use filters with m ranging from 1 to 5 or so (multiple filter widths in a single convnet)

CNNs: Implementation

- ▶ Input is $\text{batch_size} \times n \times k$ matrix, filters are $c \times m \times k$ matrix (c filters)
- ▶ Typically use filters with m ranging from 1 to 5 or so (multiple filter widths in a single convnet)
- ▶ All computation graph libraries support efficient convolution operations

CNNs for Sentence Classification

- ▶ Question classification, sentiment, etc.
- ▶ Conv+pool, then use feedforward layers to classify
- ▶ Can use multiple types of input vectors (fixed initializer and learned)



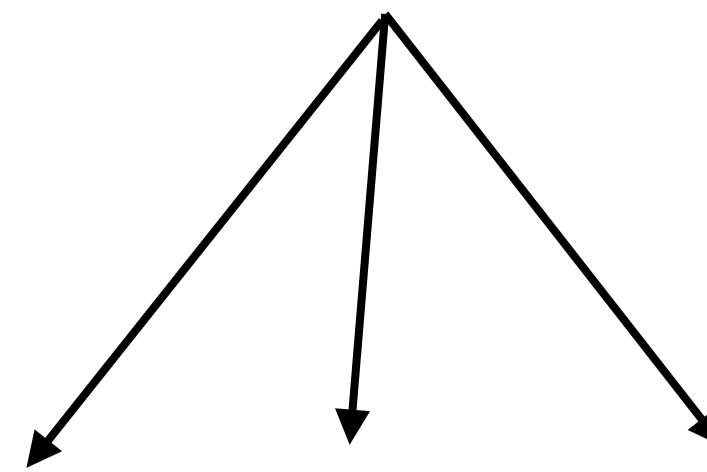
the movie was good

Sentence Classification

Model	MR	SST-1	SST-2	Subj	TREC	CR	MPQA
CNN-multichannel	81.1	47.4	88.1	93.2	92.2	85.0	89.4
NBSVM (Wang and Manning, 2012)	79.4	—	—	93.2	—	81.8	86.3

Sentence Classification

movie review
sentiment

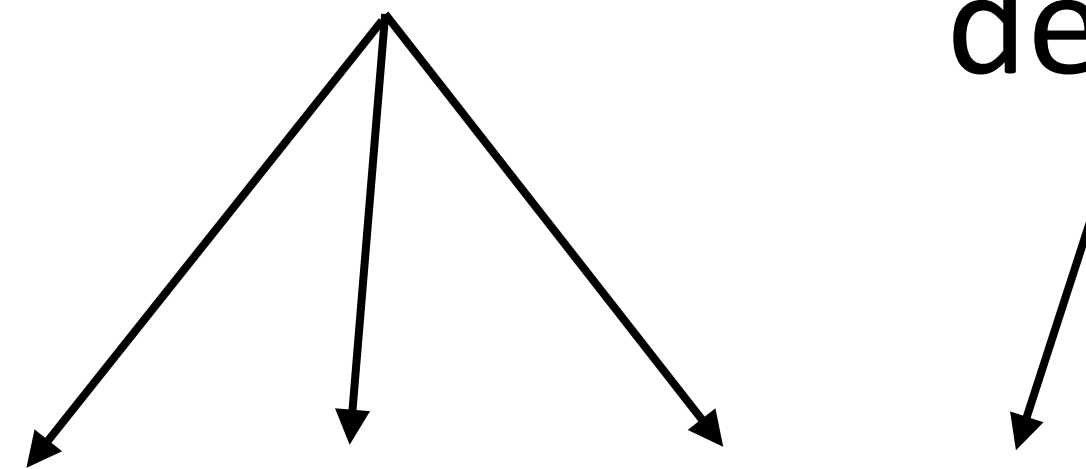


Model	MR	SST-1	SST-2	Subj	TREC	CR	MPQA
CNN-multichannel	81.1	47.4	88.1	93.2	92.2	85.0	89.4
NBSVM (Wang and Manning, 2012)	79.4	—	—	93.2	—	81.8	86.3

Sentence Classification

movie review
sentiment

subjectivity/objectivity
detection

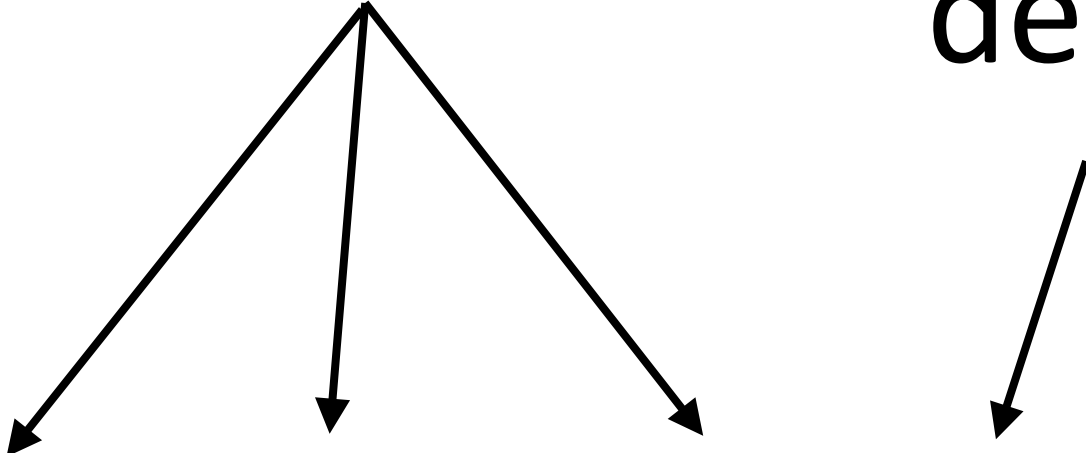


Model	MR	SST-1	SST-2	Subj	TREC	CR	MPQA
CNN-multichannel	81.1	47.4	88.1	93.2	92.2	85.0	89.4
NBSVM (Wang and Manning, 2012)	79.4	—	—	93.2	—	81.8	86.3

Sentence Classification

movie review
sentiment

subjectivity/objectivity
detection



Model	MR	SST-1	SST-2	Subj	TREC	CR	MPQA
CNN-multichannel	81.1	47.4	88.1	93.2	92.2	85.0	89.4
NBSVM (Wang and Manning, 2012)	79.4	—	—	93.2	—	81.8	86.3

question type
classification

Sentence Classification

movie review
sentiment

subjectivity/objectivity
detection

product
reviews

Model	MR	SST-1	SST-2	Subj	TREC	CR	MPQA
CNN-multichannel	81.1	47.4	88.1	93.2	92.2	85.0	89.4
NBSVM (Wang and Manning, 2012)	79.4	—	—	93.2	—	81.8	86.3

question type
classification

Sentence Classification

movie review sentiment subjectivity/objectivity detection product reviews

Model	MR	SST-1	SST-2	Subj	TREC	CR	MPQA
CNN-multichannel	81.1	47.4	88.1	93.2	92.2	85.0	89.4
NBSVM (Wang and Manning, 2012)	79.4	—	—	93.2	—	81.8	86.3

question type classification

- ▶ Also effective at document-level text classification

Neural CRF Basics

NER Revisited

B-PER I-PER O O O B-LOC O O O B-ORG O O

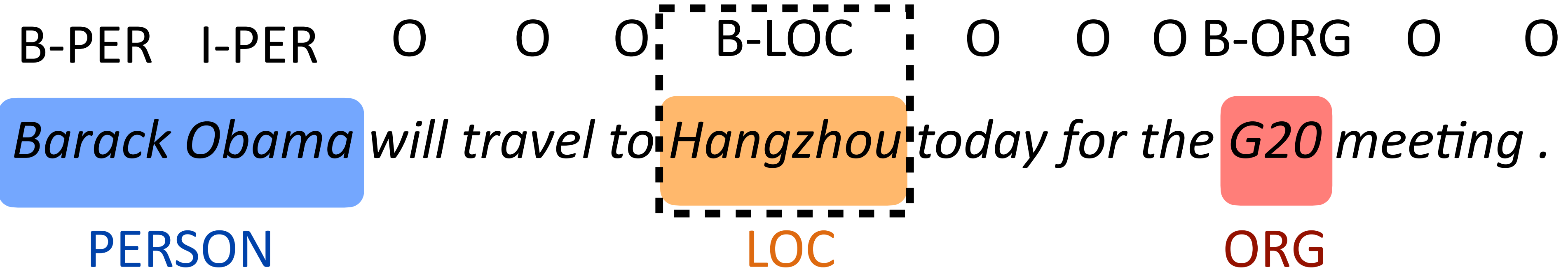
Barack Obama will travel to Hangzhou today for the G20 meeting .

PERSON

LOC

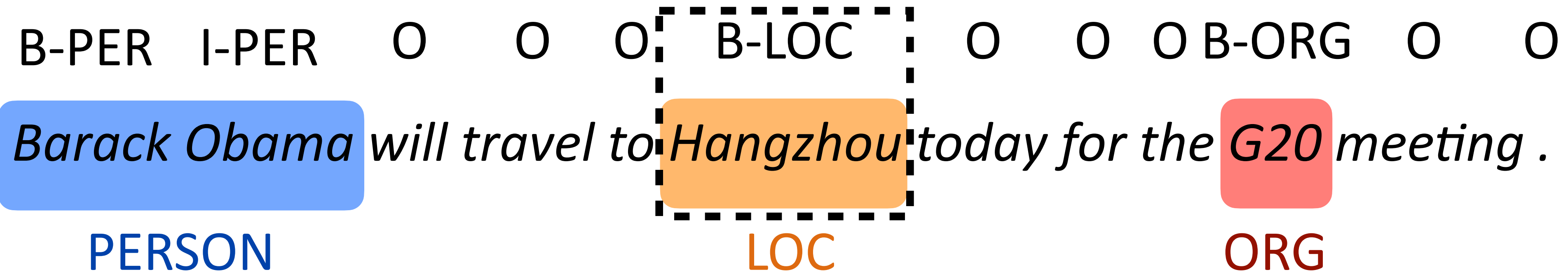
ORG

NER Revisited



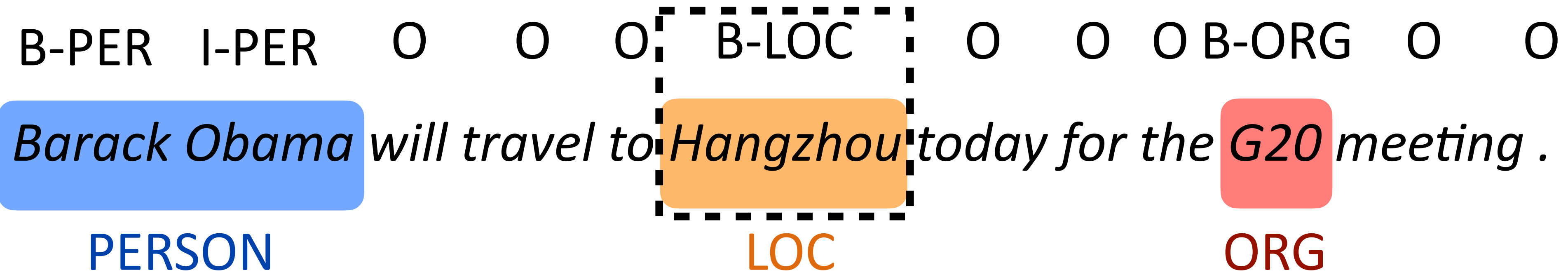
- ▶ Features in CRFs: $I[\text{tag}=\text{B-LOC} \ \& \ \text{curr_word}=\text{Hangzhou}]$,
 $I[\text{tag}=\text{B-LOC} \ \& \ \text{prev_word}=\text{to}]$, $I[\text{tag}=\text{B-LOC} \ \& \ \text{curr_prefix}=\text{Han}]$

NER Revisited



- ▶ Features in CRFs: $I[\text{tag}=\text{B-LOC} \ \& \ \text{curr_word}=\text{Hangzhou}]$,
 $I[\text{tag}=\text{B-LOC} \ \& \ \text{prev_word}=\text{to}]$, $I[\text{tag}=\text{B-LOC} \ \& \ \text{curr_prefix}=\text{Han}]$
- ▶ Linear model over features

NER Revisited



- ▶ Features in CRFs: $I[\text{tag}=\text{B-LOC} \ \& \ \text{curr_word}=\text{Hangzhou}]$, $I[\text{tag}=\text{B-LOC} \ \& \ \text{prev_word}=\text{to}]$, $I[\text{tag}=\text{B-LOC} \ \& \ \text{curr_prefix}=\text{Han}]$
- ▶ Linear model over features
- ▶ Downsides:

NER Revisited



- ▶ Features in CRFs: $I[\text{tag}=\text{B-LOC} \ \& \ \text{curr_word}=\text{Hangzhou}]$, $I[\text{tag}=\text{B-LOC} \ \& \ \text{prev_word}=\text{to}]$, $I[\text{tag}=\text{B-LOC} \ \& \ \text{curr_prefix}=\text{Han}]$
- ▶ Linear model over features
- ▶ Downsides:
 - ▶ Lexical features mean that words need to be seen in the training data

NER Revisited



- ▶ Features in CRFs: $I[\text{tag}=\text{B-LOC} \ \& \ \text{curr_word}=\text{Hangzhou}]$, $I[\text{tag}=\text{B-LOC} \ \& \ \text{prev_word}=\text{to}]$, $I[\text{tag}=\text{B-LOC} \ \& \ \text{curr_prefix}=\text{Han}]$
- ▶ Linear model over features
- ▶ Downsides:
 - ▶ Lexical features mean that words need to be seen in the training data
 - ▶ Linear model can't capture feature conjunctions as effectively (doesn't work well to look at more than 2 words with a single feature)

LSTMs for NER

B-PER I-PER O O O B-LOC O O O B-ORG O O

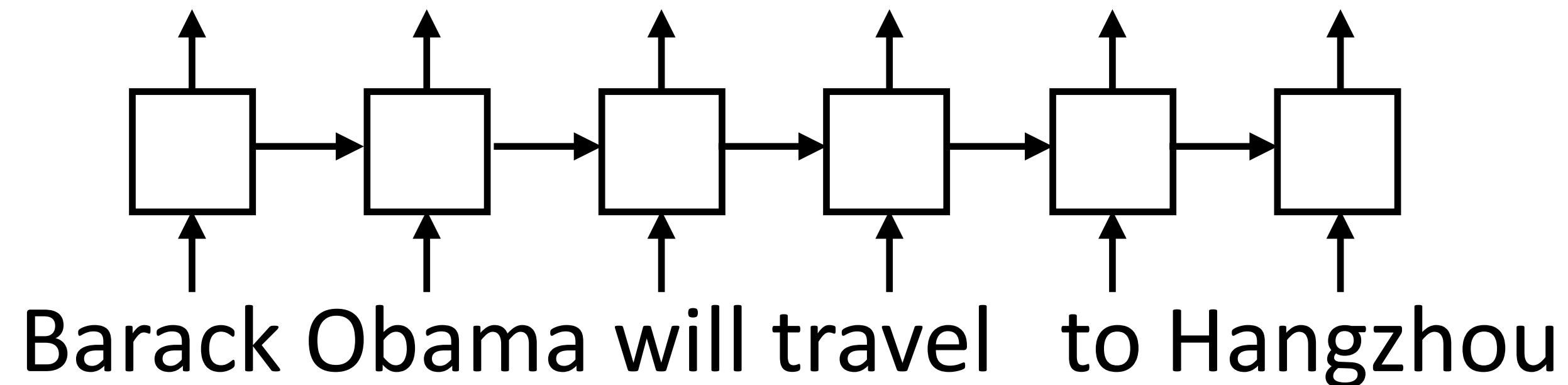
Barack Obama will travel to Hangzhou today for the G20 meeting .

PERSON

LOC

ORG

B-PER I-PER O O O B-LOC



- ▶ Transducer (LM-like model)
- ▶ What are the strengths and weaknesses of this model compared to CRFs?

LSTMs for NER

B-PER I-PER O O O B-LOC O O O B-ORG O O

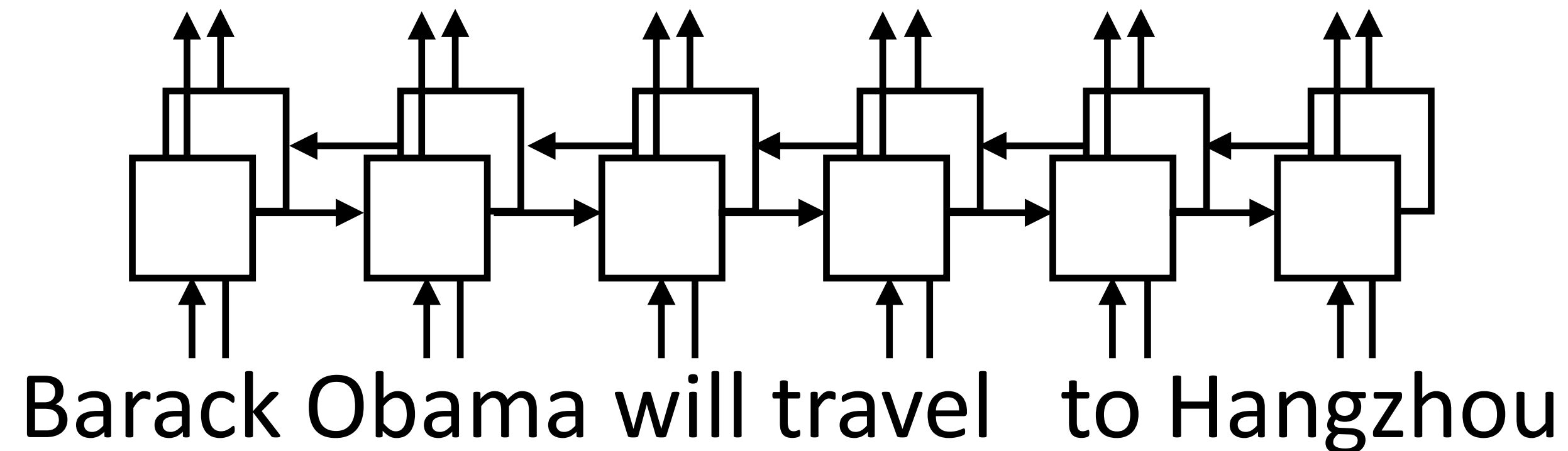
Barack Obama will travel to Hangzhou today for the G20 meeting .

PERSON

LOC

ORG

B-PER I-PER O O O B-LOC



- ▶ Bidirectional transducer model
- ▶ What are the strengths and weaknesses of this model compared to CRFs?

Neural CRFs

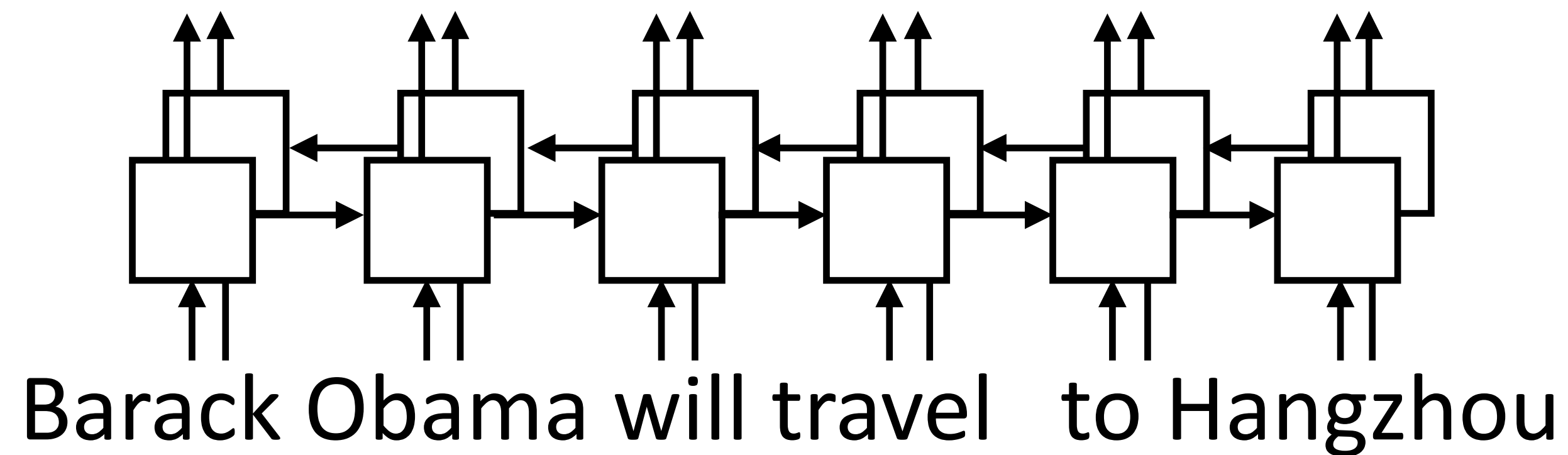
B-PER I-PER O O O B-LOC O O O B-ORG O O

Barack Obama will travel to Hangzhou today for the G20 meeting .

PERSON

LOC

ORG



Neural CRFs

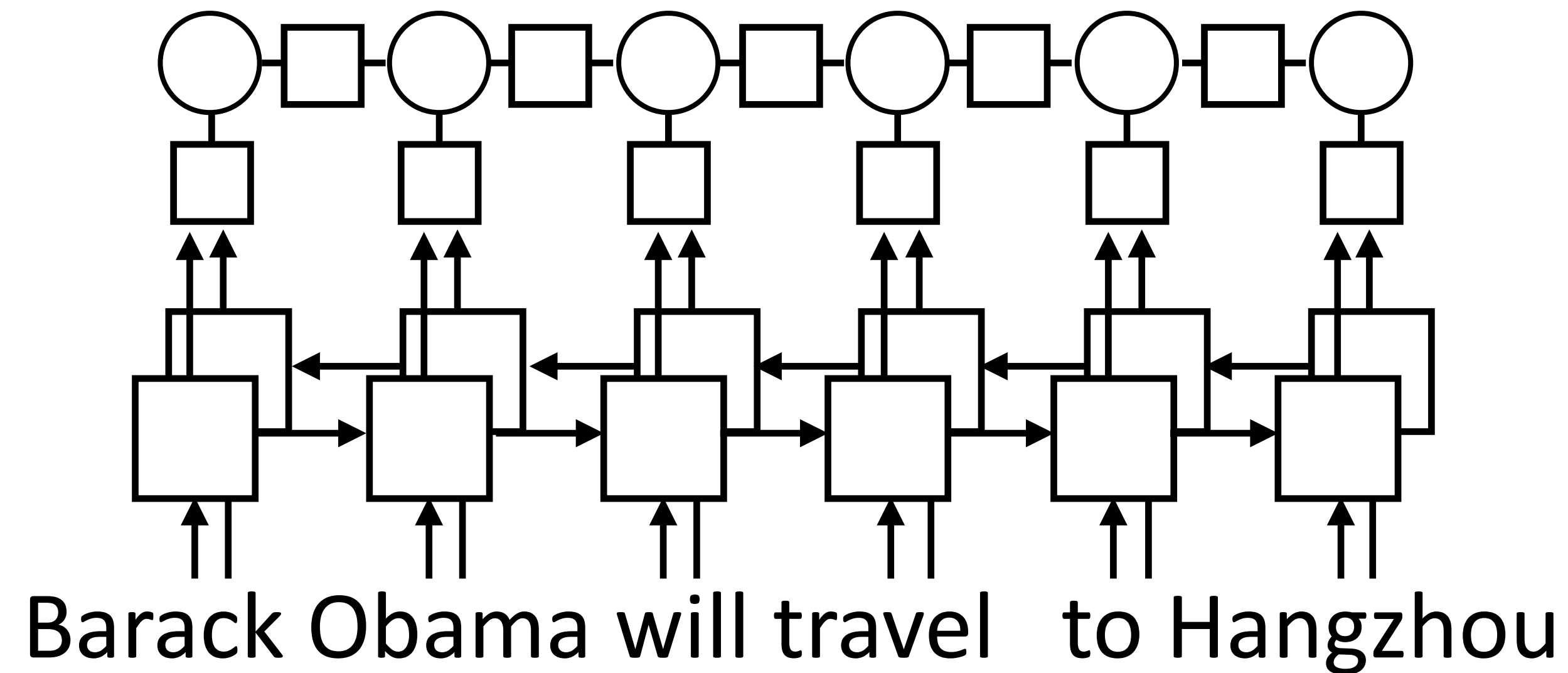
B-PER I-PER O O O B-LOC O O O B-ORG O O

Barack Obama will travel to Hangzhou today for the G20 meeting .

PERSON

LOC

ORG



Neural CRFs

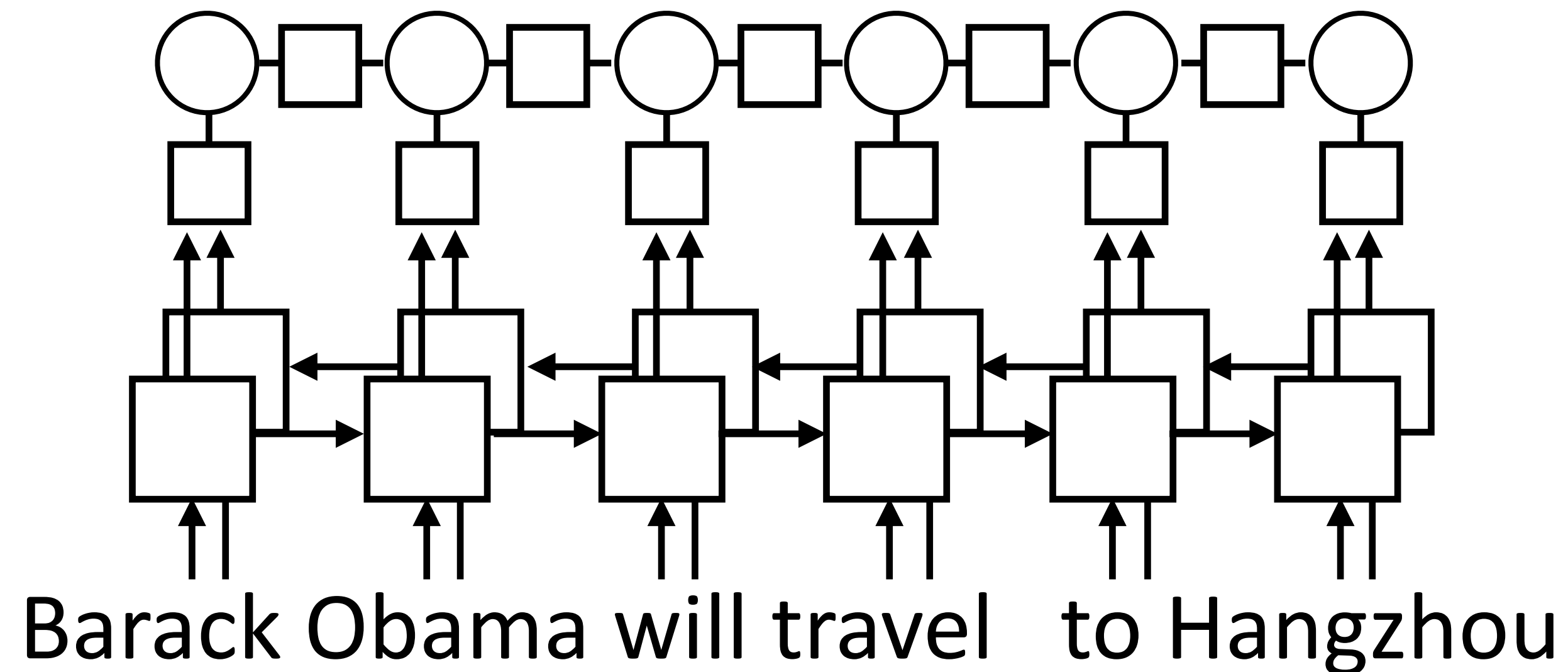
B-PER I-PER O O O B-LOC O O O B-ORG O O

Barack Obama will travel to Hangzhou today for the G20 meeting .

PERSON

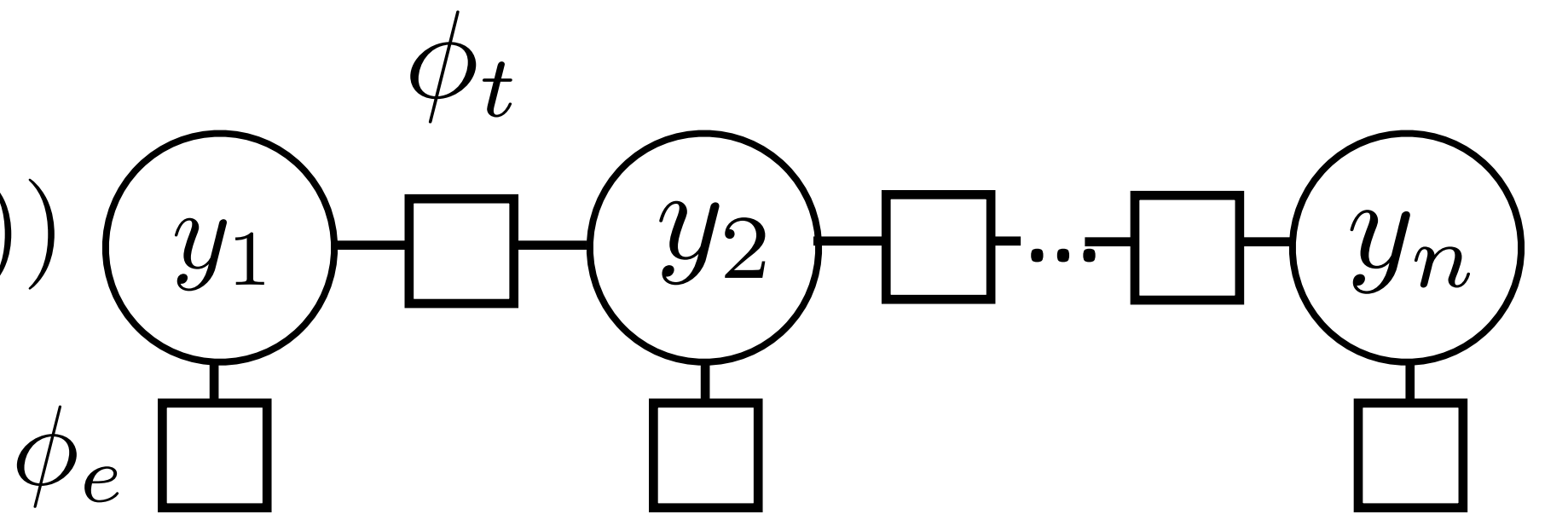
LOC

ORG



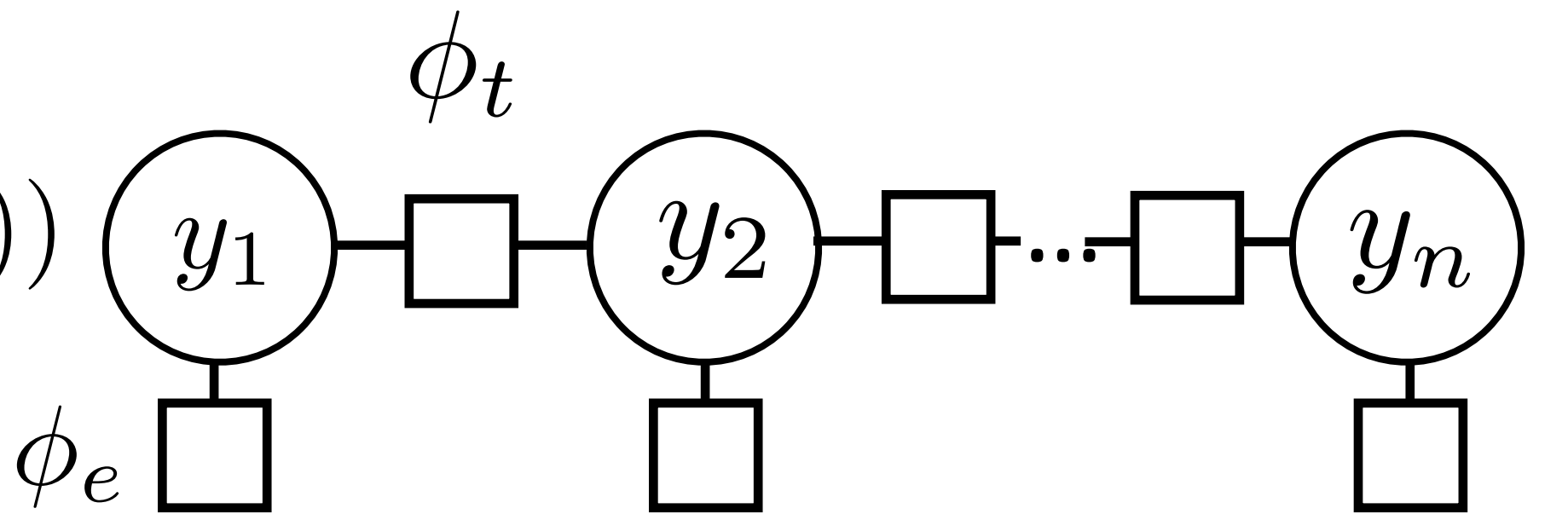
- ▶ Neural CRFs: bidirectional LSTMs (or some NN) compute emission potentials, capture structural constraints in transition potentials

Neural CRFs

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \prod_{i=2}^n \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^n \exp(\phi_e(y_i, i, \mathbf{x}))$$


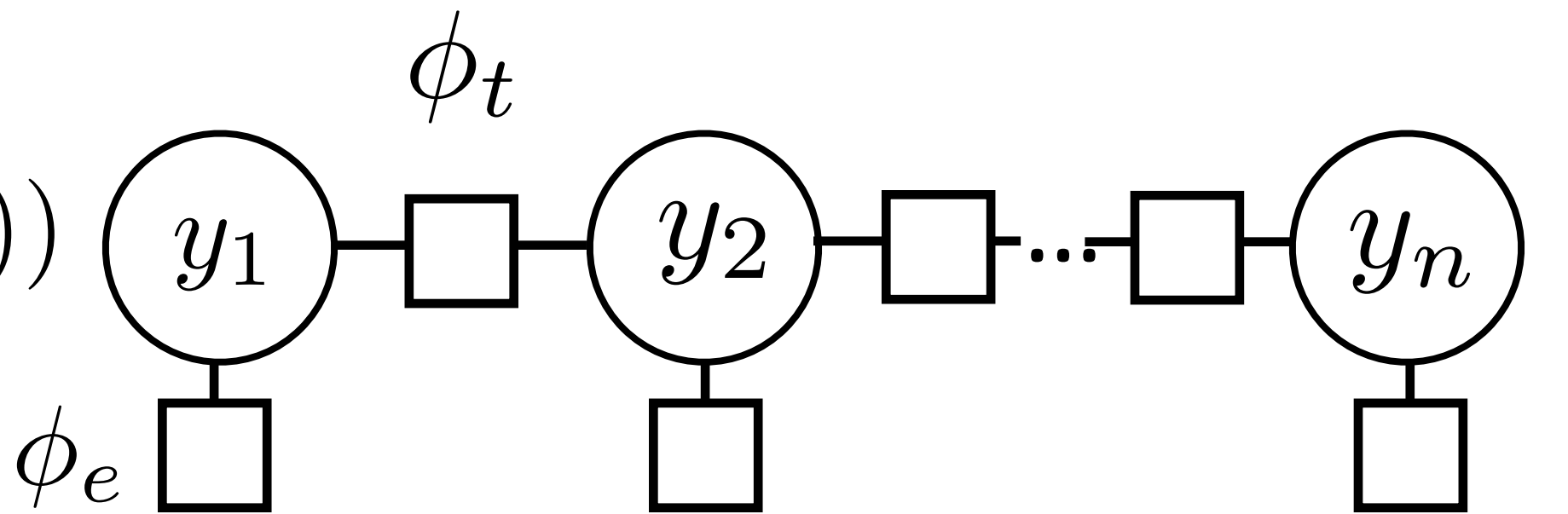
- ▶ Conventional: $\phi_e(y_i, i, \mathbf{x}) = w^\top f_e(y_i, i, \mathbf{x})$

Neural CRFs

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \prod_{i=2}^n \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^n \exp(\phi_e(y_i, i, \mathbf{x}))$$


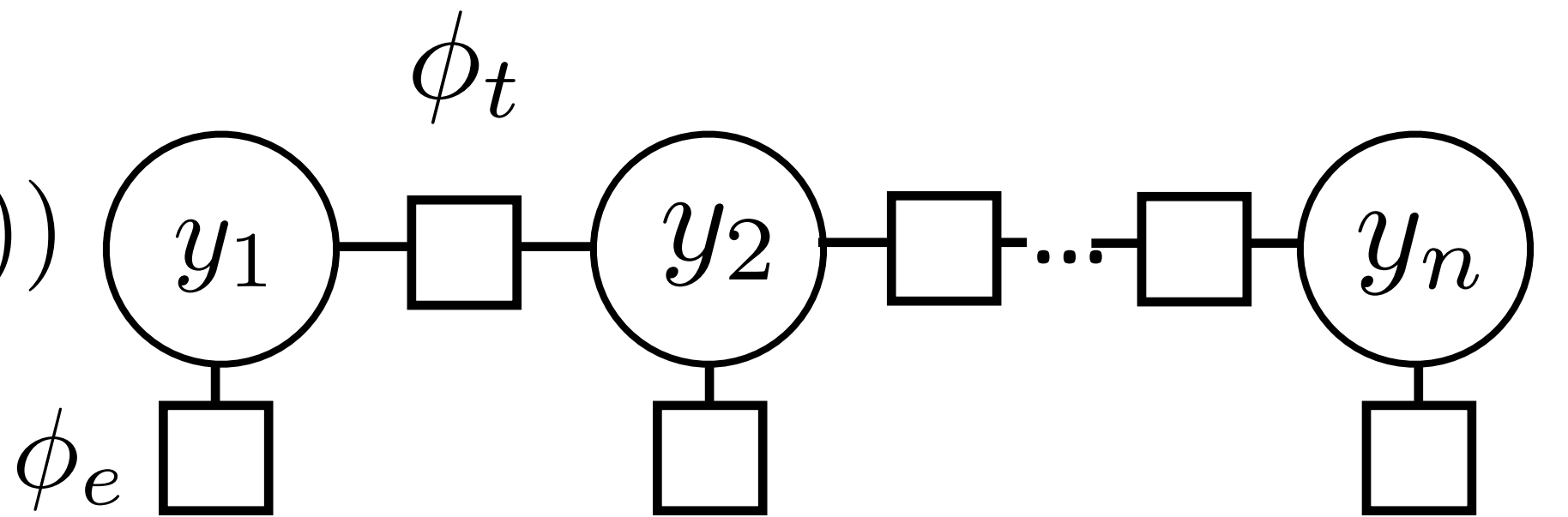
- ▶ Conventional: $\phi_e(y_i, i, \mathbf{x}) = w^\top f_e(y_i, i, \mathbf{x})$
- ▶ Neural: $\phi_e(y_i, i, \mathbf{x}) = W_{y_i}^\top f(i, \mathbf{x})$ W is a num_tags x len(f) matrix

Neural CRFs

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \prod_{i=2}^n \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^n \exp(\phi_e(y_i, i, \mathbf{x}))$$


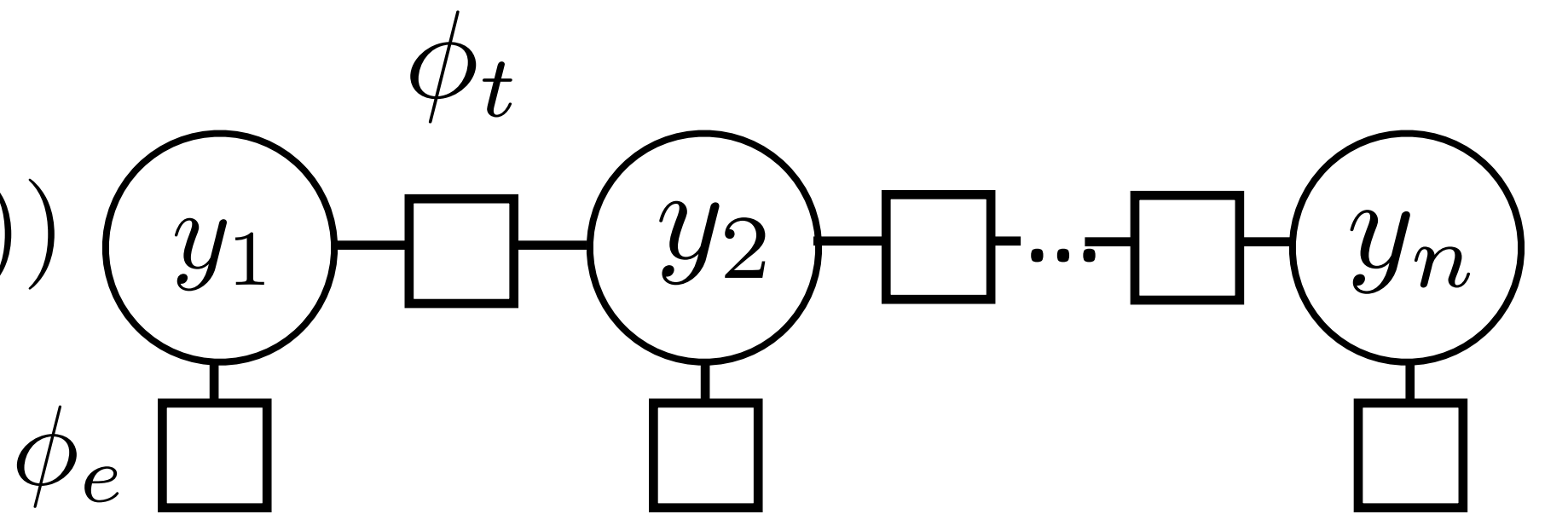
- ▶ Conventional: $\phi_e(y_i, i, \mathbf{x}) = w^\top f_e(y_i, i, \mathbf{x})$
- ▶ Neural: $\phi_e(y_i, i, \mathbf{x}) = W_{y_i}^\top f(i, \mathbf{x})$ W is a `num_tags x len(f)` matrix
- ▶ $f(i, \mathbf{x})$ could be the output of a feedforward neural network looking at the words around position i , or the i th output of an LSTM, ...

Neural CRFs

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \prod_{i=2}^n \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^n \exp(\phi_e(y_i, i, \mathbf{x}))$$


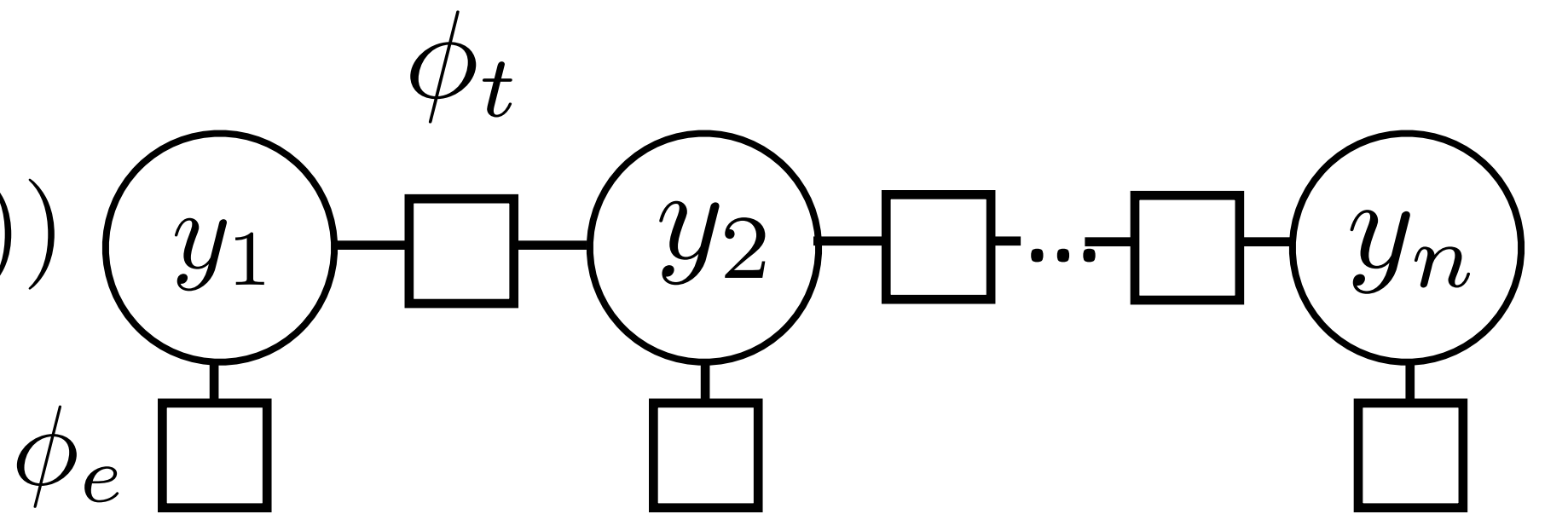
- ▶ Conventional: $\phi_e(y_i, i, \mathbf{x}) = w^\top f_e(y_i, i, \mathbf{x})$
- ▶ Neural: $\phi_e(y_i, i, \mathbf{x}) = W_{y_i}^\top f(i, \mathbf{x})$ W is a `num_tags x len(f)` matrix
- ▶ $f(i, \mathbf{x})$ could be the output of a feedforward neural network looking at the words around position i , or the i th output of an LSTM, ...
- ▶ Neural network computes unnormalized potentials that are consumed and “normalized” by a structured model

Neural CRFs

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \prod_{i=2}^n \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^n \exp(\phi_e(y_i, i, \mathbf{x}))$$


- ▶ Conventional: $\phi_e(y_i, i, \mathbf{x}) = w^\top f_e(y_i, i, \mathbf{x})$
- ▶ Neural: $\phi_e(y_i, i, \mathbf{x}) = W_{y_i}^\top f(i, \mathbf{x})$ W is a `num_tags x len(f)` matrix
- ▶ $f(i, \mathbf{x})$ could be the output of a feedforward neural network looking at the words around position i , or the i th output of an LSTM, ...
- ▶ Neural network computes unnormalized potentials that are consumed and “normalized” by a structured model
- ▶ Inference: compute f , use Viterbi

Computing Gradients

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \prod_{i=2}^n \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^n \exp(\phi_e(y_i, i, \mathbf{x}))$$


The diagram illustrates a sequence model. It consists of a horizontal chain of nodes. The nodes are circles labeled y_1, y_2, \dots, y_n and squares. The circles represent hidden states, and the squares represent input and output nodes. The first hidden state y_1 is connected to an input square labeled ϕ_e . The second hidden state y_2 is connected to an output square. The transition between hidden states y_{i-1} and y_i is labeled ϕ_t . The input and output nodes are also connected to the hidden states.

- ▶ Conventional: $\phi_e(y_i, i, \mathbf{x}) = w^\top f_e(y_i, i, \mathbf{x})$
- ▶ Neural: $\phi_e(y_i, i, \mathbf{x}) = W_{y_i}^\top f(i, \mathbf{x})$

Computing Gradients

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \prod_{i=2}^n \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^n \exp(\phi_e(y_i, i, \mathbf{x}))$$

▶ Conventional: $\phi_e(y_i, i, \mathbf{x}) = w^\top f_e(y_i, i, \mathbf{x})$

▶ Neural: $\phi_e(y_i, i, \mathbf{x}) = W_{y_i}^\top f(i, \mathbf{x})$

$$\frac{\partial \mathcal{L}}{\partial \phi_{e,i}} = -P(y_i = s|\mathbf{x}) + I[s \text{ is gold}] \quad \text{“error signal”, compute with F-B}$$

Computing Gradients

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \prod_{i=2}^n \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^n \exp(\phi_e(y_i, i, \mathbf{x}))$$

▶ Conventional: $\phi_e(y_i, i, \mathbf{x}) = w^\top f_e(y_i, i, \mathbf{x})$

▶ Neural: $\phi_e(y_i, i, \mathbf{x}) = W_{y_i}^\top f(i, \mathbf{x})$

$$\frac{\partial \mathcal{L}}{\partial \phi_{e,i}} = -P(y_i = s|\mathbf{x}) + I[s \text{ is gold}] \quad \text{“error signal”, compute with F-B}$$

▶ For linear model: $\frac{\partial \phi_{e,i}}{w_i} = f_{e,i}(y_i, i, \mathbf{x})$

Computing Gradients

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \prod_{i=2}^n \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^n \exp(\phi_e(y_i, i, \mathbf{x}))$$

► Conventional: $\phi_e(y_i, i, \mathbf{x}) = w^\top f_e(y_i, i, \mathbf{x})$

► Neural: $\phi_e(y_i, i, \mathbf{x}) = W_{y_i}^\top f(i, \mathbf{x})$

$$\frac{\partial \mathcal{L}}{\partial \phi_{e,i}} = -P(y_i = s|\mathbf{x}) + I[s \text{ is gold}]$$

“error signal”, compute with F-B

► For linear model: $\frac{\partial \phi_{e,i}}{w_i} = f_{e,i}(y_i, i, \mathbf{x})$

chain rule say to multiply together, gives our update

Computing Gradients

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \prod_{i=2}^n \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^n \exp(\phi_e(y_i, i, \mathbf{x}))$$

▶ Conventional: $\phi_e(y_i, i, \mathbf{x}) = w^\top f_e(y_i, i, \mathbf{x})$

▶ Neural: $\phi_e(y_i, i, \mathbf{x}) = W_{y_i}^\top f(i, \mathbf{x})$

$\frac{\partial \mathcal{L}}{\partial \phi_{e,i}} = -P(y_i = s|\mathbf{x}) + I[s \text{ is gold}]$ “error signal”, compute with F-B

▶ For linear model: $\frac{\partial \phi_{e,i}}{w_i} = f_{e,i}(y_i, i, \mathbf{x})$ chain rule say to multiply together, gives our update

▶ For neural model: compute gradient of phi w.r.t. parameters of neural net

Neural CRFs

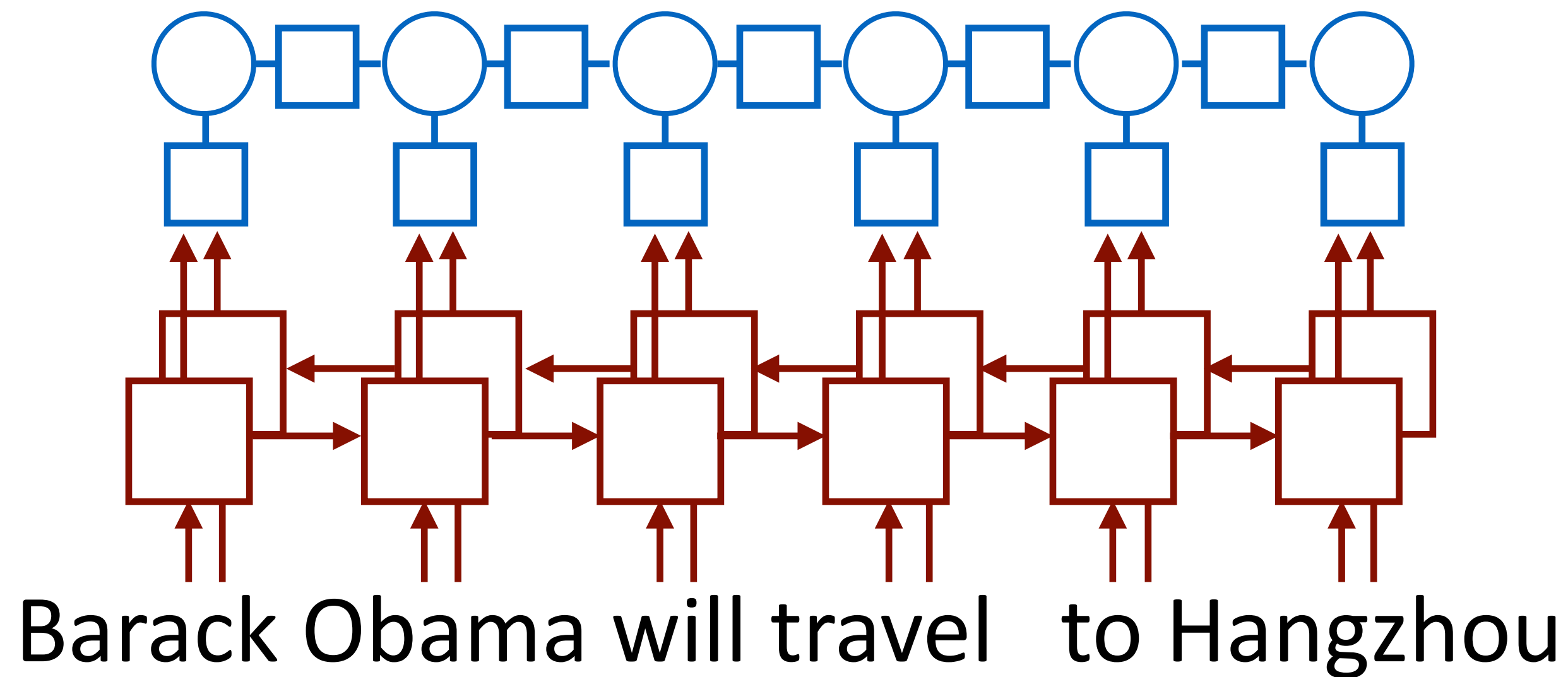
B-PER I-PER O O O B-LOC O O O B-ORG O O

Barack Obama will travel to Hangzhou today for the G20 meeting .

PERSON

LOC

ORG



Neural CRFs

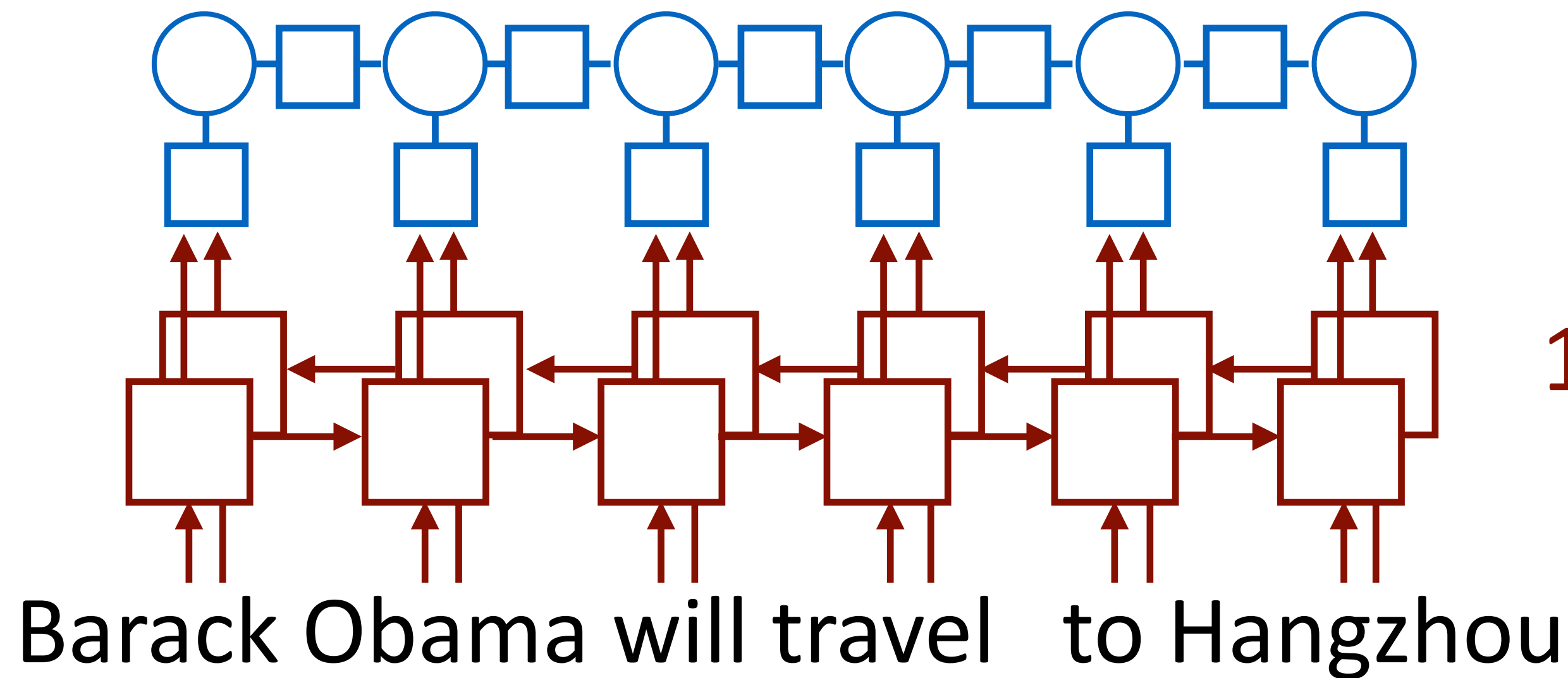
B-PER I-PER O O O B-LOC O O O B-ORG O O

Barack Obama will travel to Hangzhou today for the G20 meeting .

PERSON

LOC

ORG



1) Compute $f(\mathbf{x})$

Neural CRFs

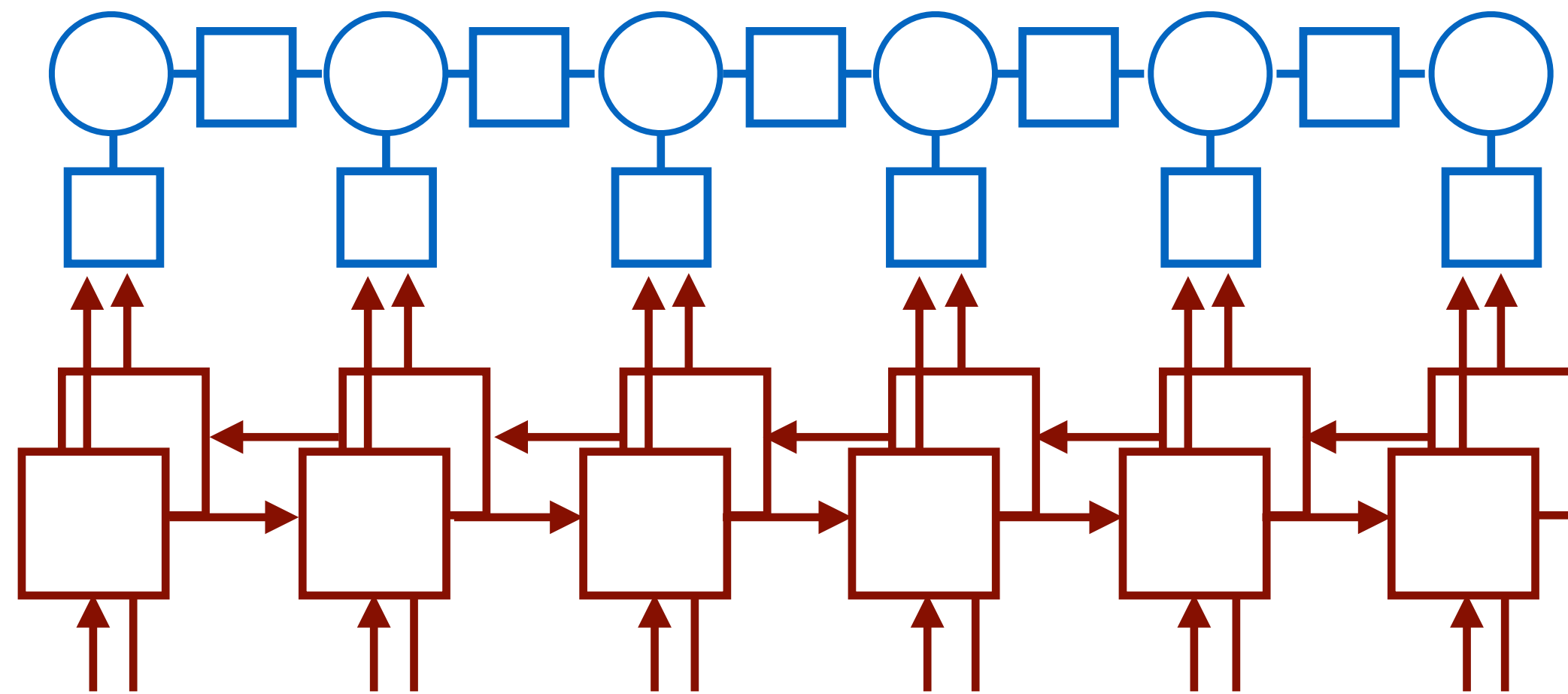
B-PER I-PER O O O B-LOC O O O B-ORG O O

Barack Obama will travel to Hangzhou today for the G20 meeting .

PERSON

LOC

ORG



Barack Obama will travel to Hangzhou

2) Run forward-backward

1) Compute $f(\mathbf{x})$

Neural CRFs

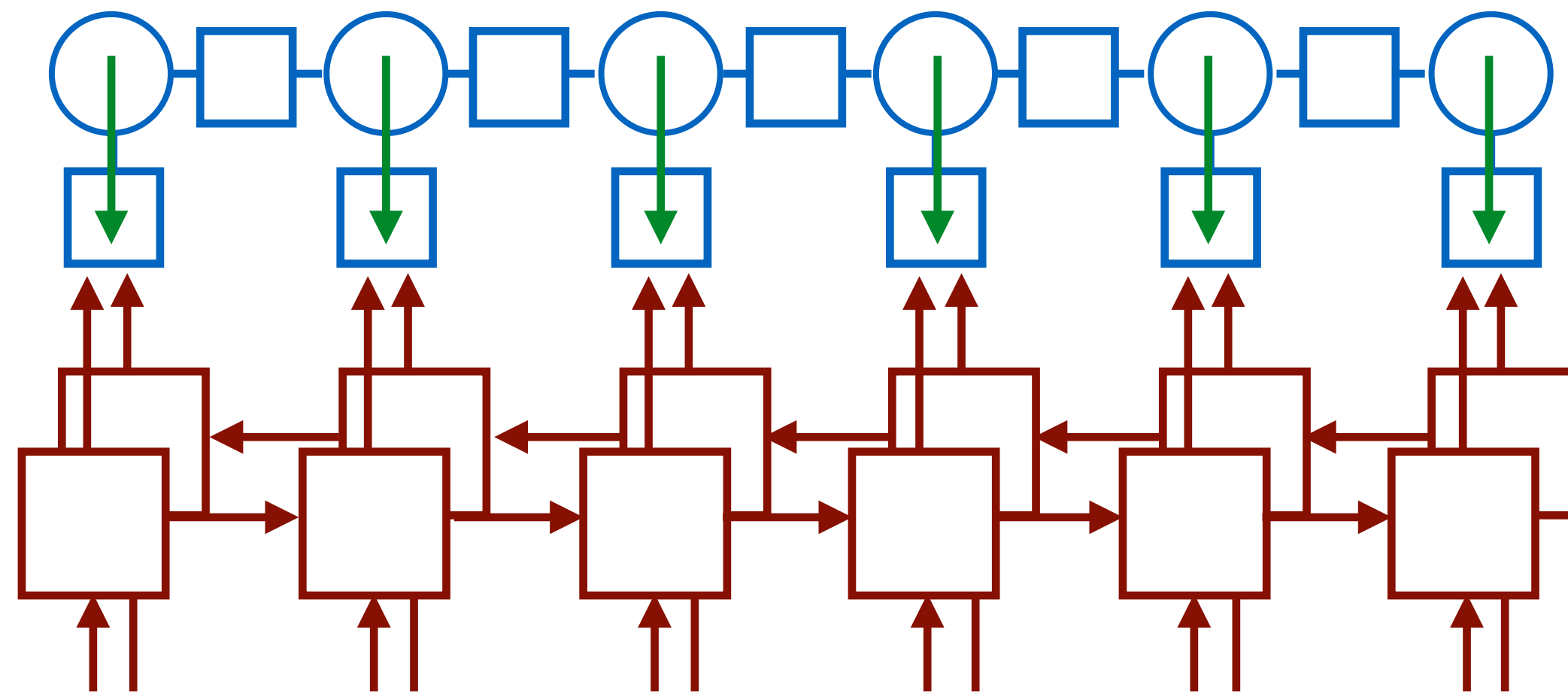
B-PER I-PER O O O B-LOC O O O B-ORG O O

Barack Obama will travel to Hangzhou today for the G20 meeting .

PERSON

LOC

ORG



Barack Obama will travel to Hangzhou

2) Run forward-backward

3) Compute error signal

1) Compute $f(\mathbf{x})$

Neural CRFs

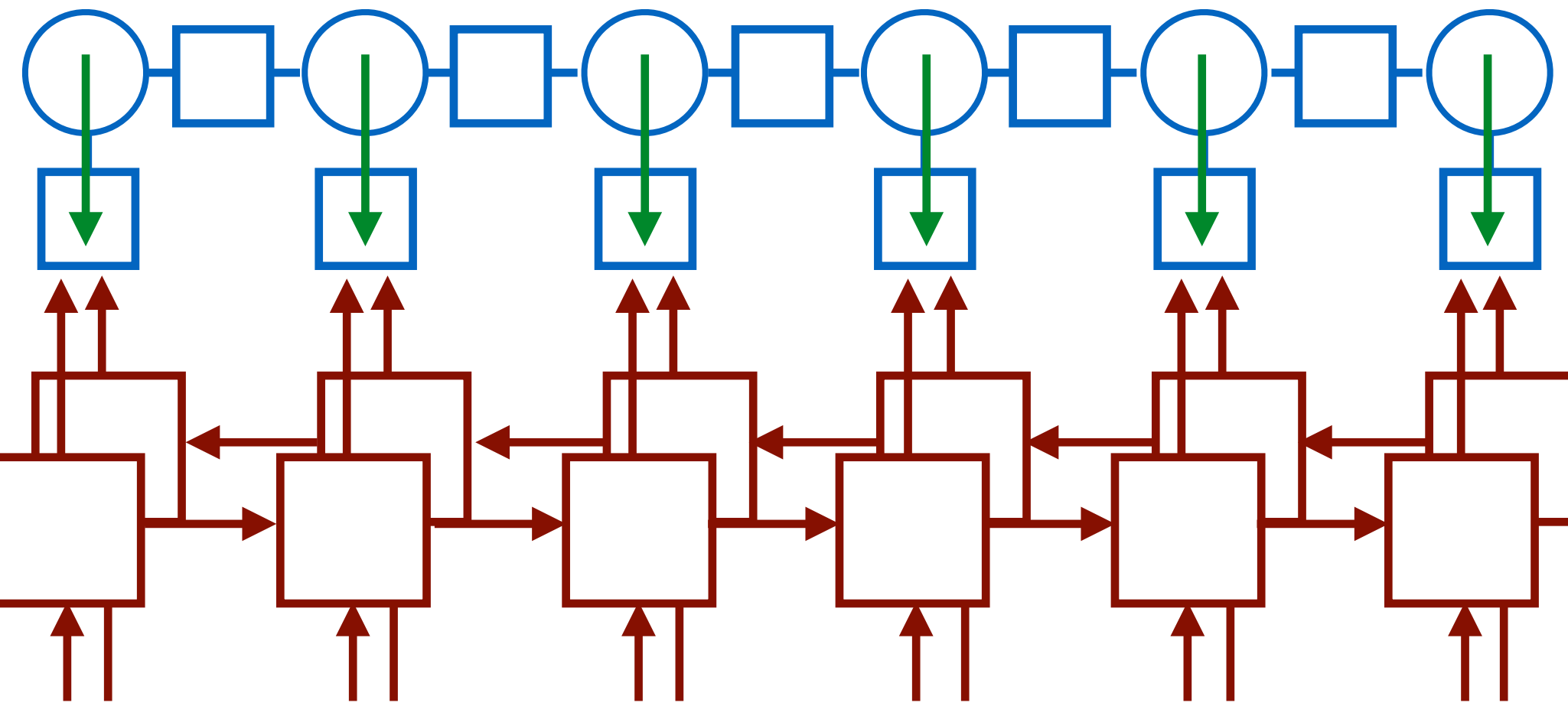
B-PER I-PER O O O B-LOC O O O B-ORG O O

Barack Obama will travel to Hangzhou today for the G20 meeting .

PERSON

LOC

ORG



Barack Obama will travel to Hangzhou

2) Run forward-backward

3) Compute error signal

1) Compute $f(\mathbf{x})$

4) Backprop (no knowledge of sequential structure required)

FFNN Neural CRF for NER

B-PER I-PER O O O B-LOC O O O B-ORG O O

Barack Obama will travel to Hangzhou today for the G20 meeting .

PERSON

LOC

ORG

FFNN Neural CRF for NER

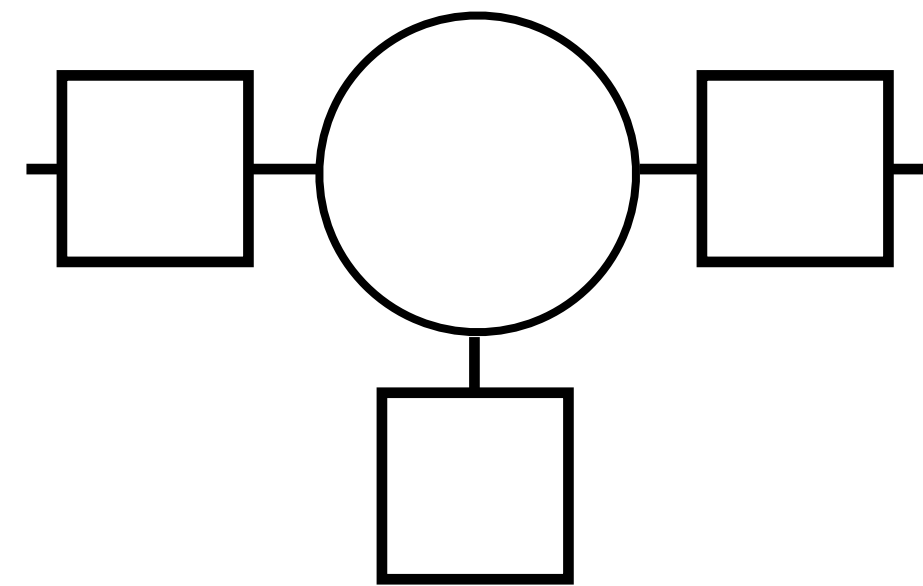
B-PER I-PER O O O B-LOC O O O B-ORG O O

Barack Obama will travel to Hangzhou today for the G20 meeting .

PERSON

LOC

ORG



to Hangzhou today

FFNN Neural CRF for NER

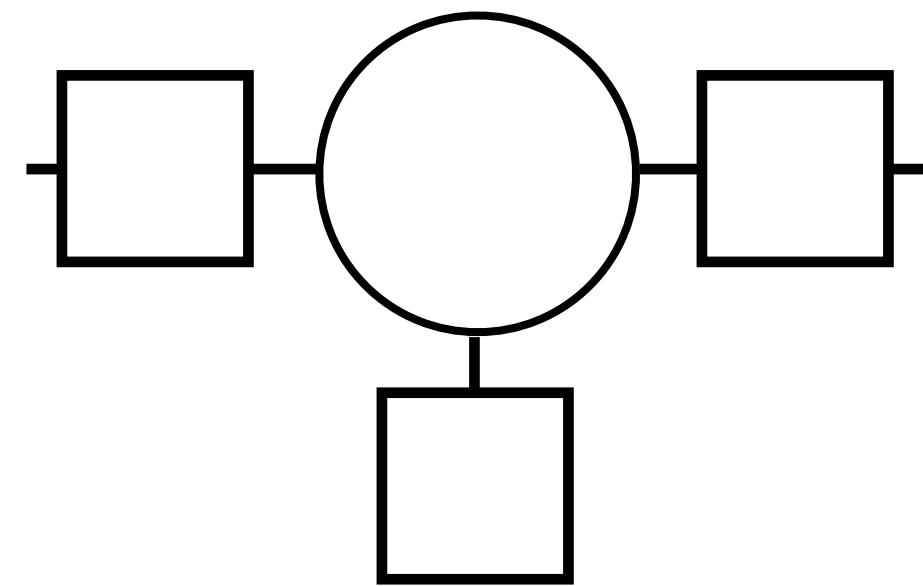
B-PER I-PER O O O B-LOC O O O B-ORG O O

Barack Obama will travel to Hangzhou today for the G20 meeting .

PERSON

LOC

ORG



$$f(\mathbf{x}, i) = [\text{emb}(\mathbf{x}_{i-1}), \text{emb}(\mathbf{x}_i), \text{emb}(\mathbf{x}_{i+1})]$$



previous word curr word next word

to Hangzhou today

FFNN Neural CRF for NER

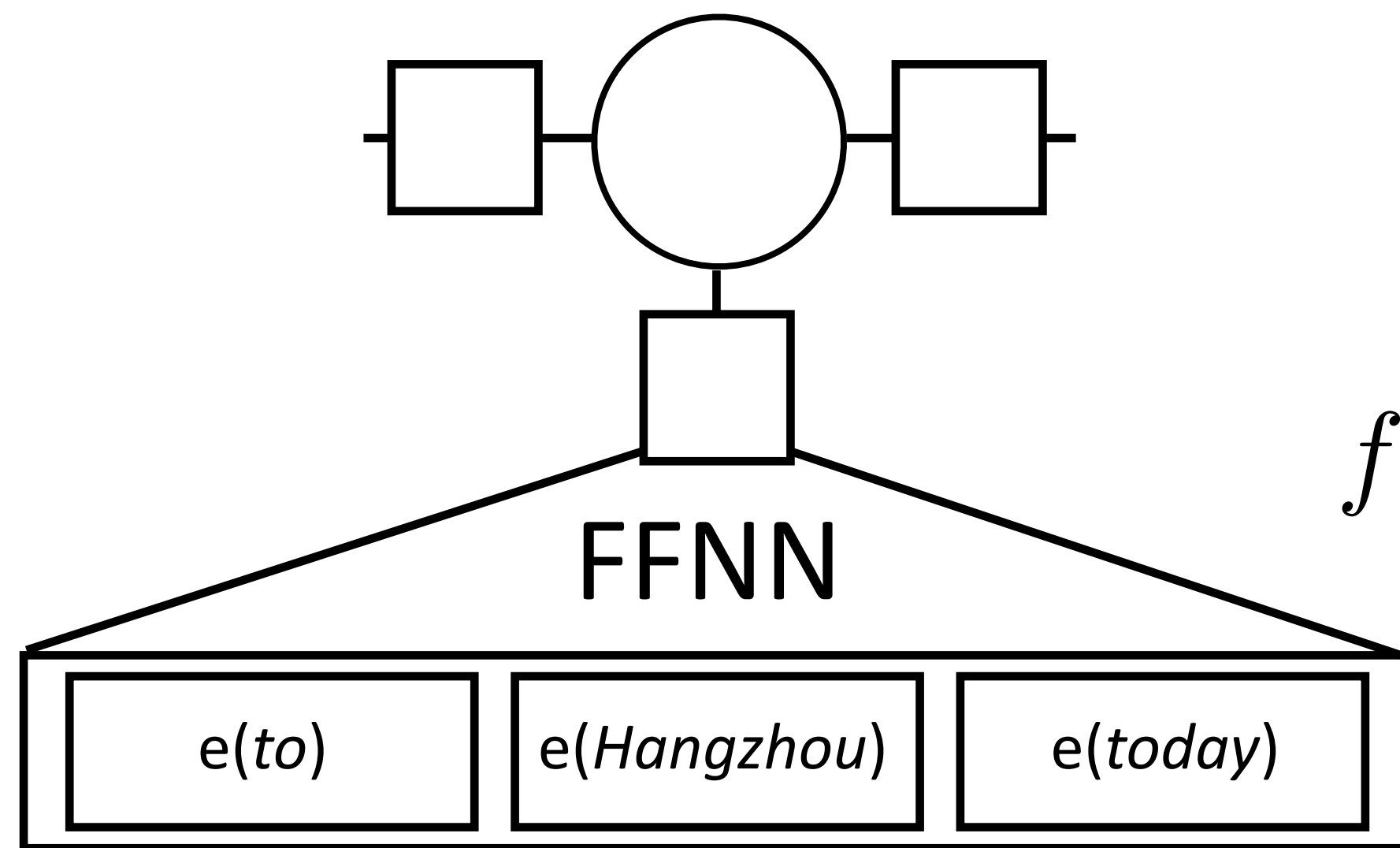
B-PER I-PER O O O B-LOC O O O B-ORG O O

Barack Obama will travel to Hangzhou today for the G20 meeting .

PERSON

LOC

ORG



$$f(\mathbf{x}, i) = [\text{emb}(\mathbf{x}_{i-1}), \text{emb}(\mathbf{x}_i), \text{emb}(\mathbf{x}_{i+1})]$$

previous word curr word next word

to Hangzhou today

FFNN Neural CRF for NER

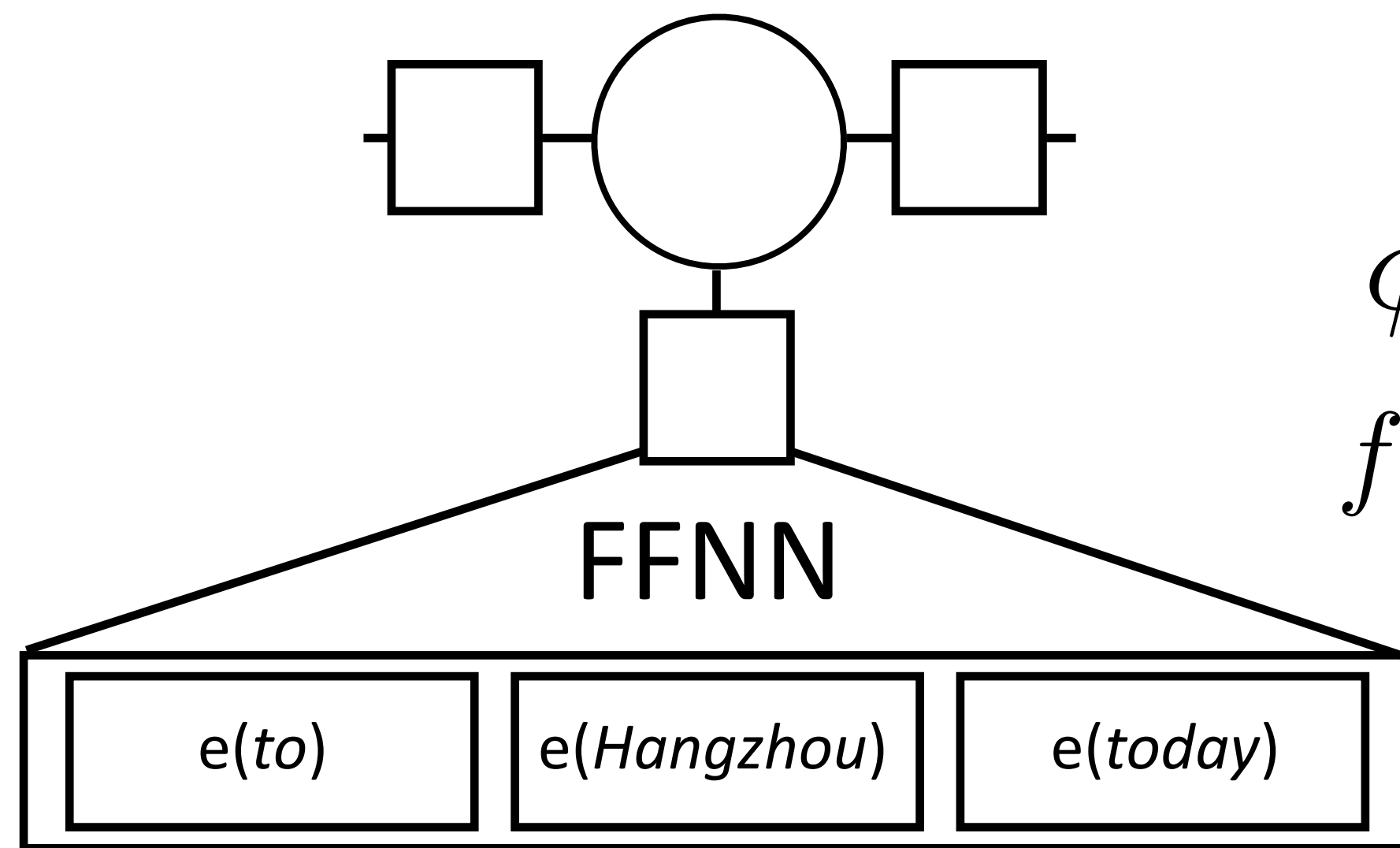
B-PER I-PER O O O B-LOC O O O B-ORG O O

Barack Obama will travel to Hangzhou today for the G20 meeting .

PERSON

LOC

ORG



$$\phi_e = Wg(Vf(\mathbf{x}, i))$$

$$f(\mathbf{x}, i) = [\text{emb}(\mathbf{x}_{i-1}), \text{emb}(\mathbf{x}_i), \text{emb}(\mathbf{x}_{i+1})]$$

previous word curr word next word

to Hangzhou today

LSTM Neural CRFs

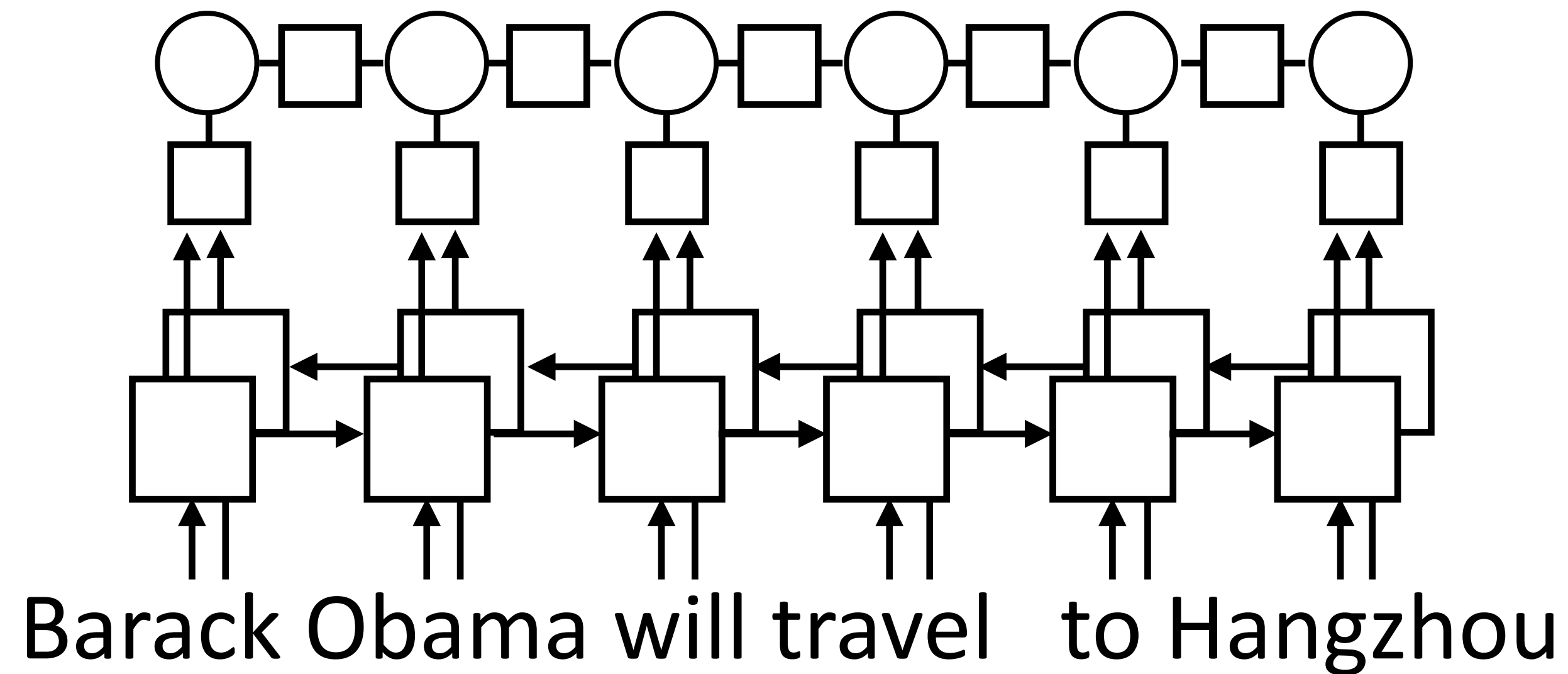
B-PER I-PER O O O B-LOC O O O B-ORG O O

Barack Obama will travel to Hangzhou today for the G20 meeting .

PERSON

LOC

ORG



- ▶ Bidirectional LSTMs compute emission (or transition) potentials

LSTMs for NER

B-PER I-PER O O O B-LOC O O O B-ORG O O

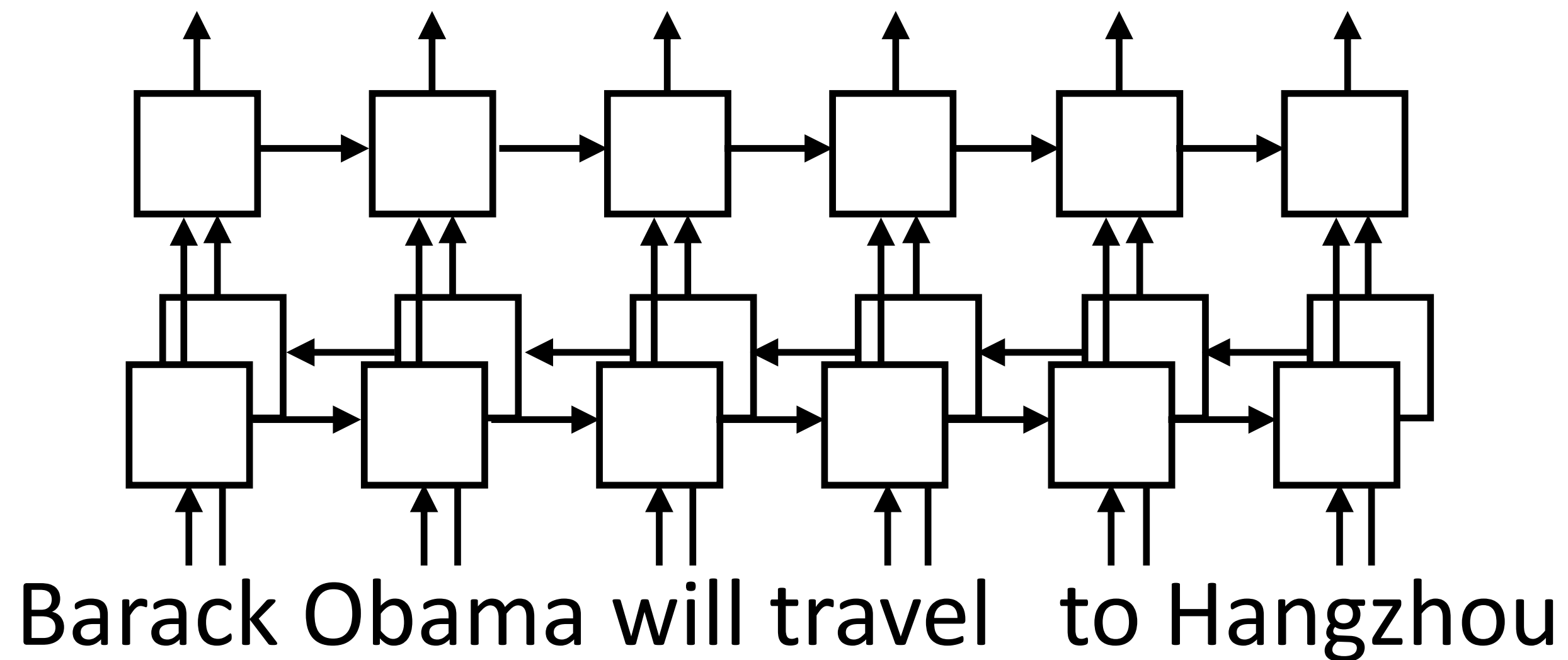
Barack Obama will travel to Hangzhou today for the G20 meeting .

PERSON

LOC

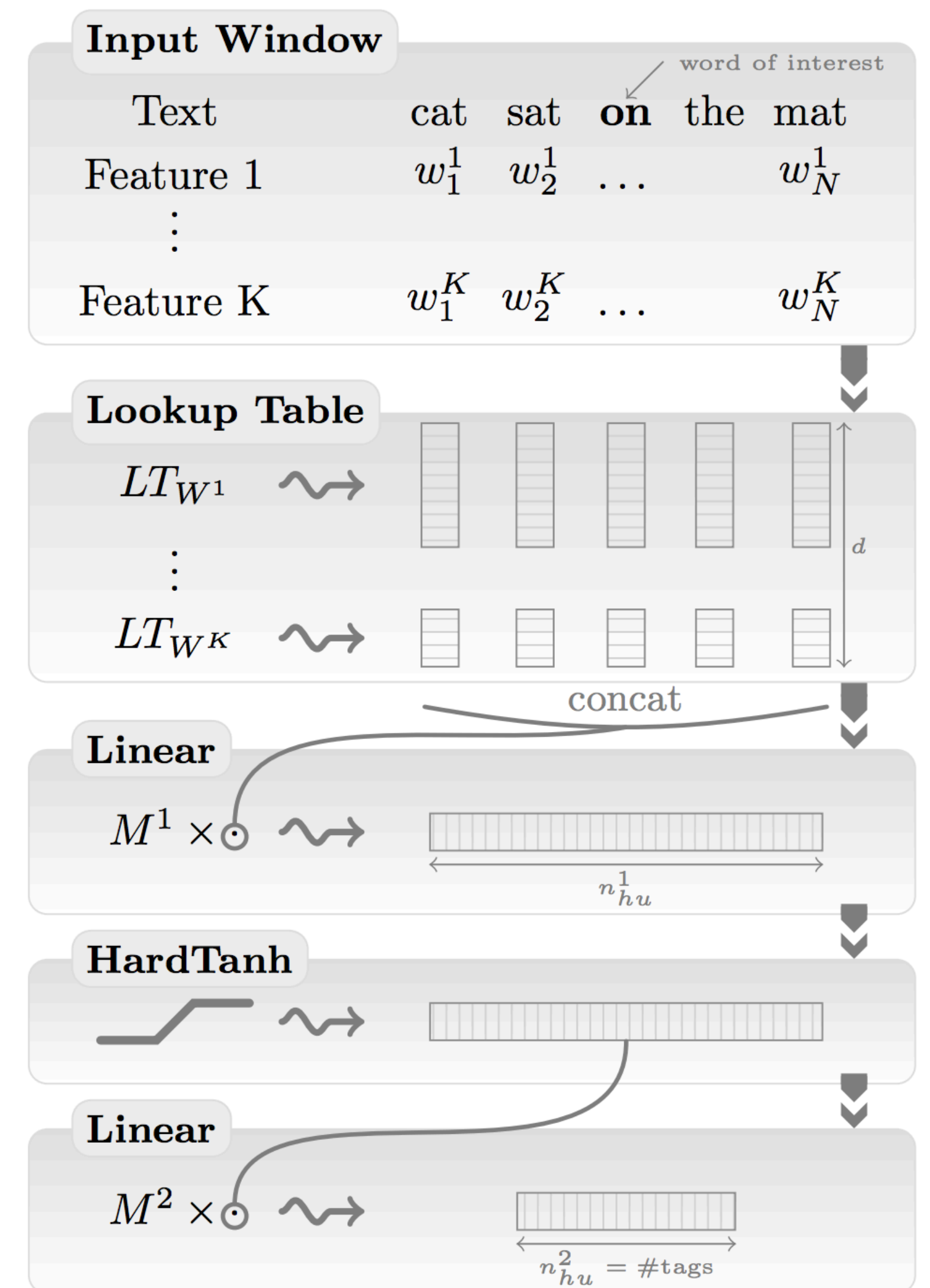
ORG

B-PER I-PER O O O B-LOC



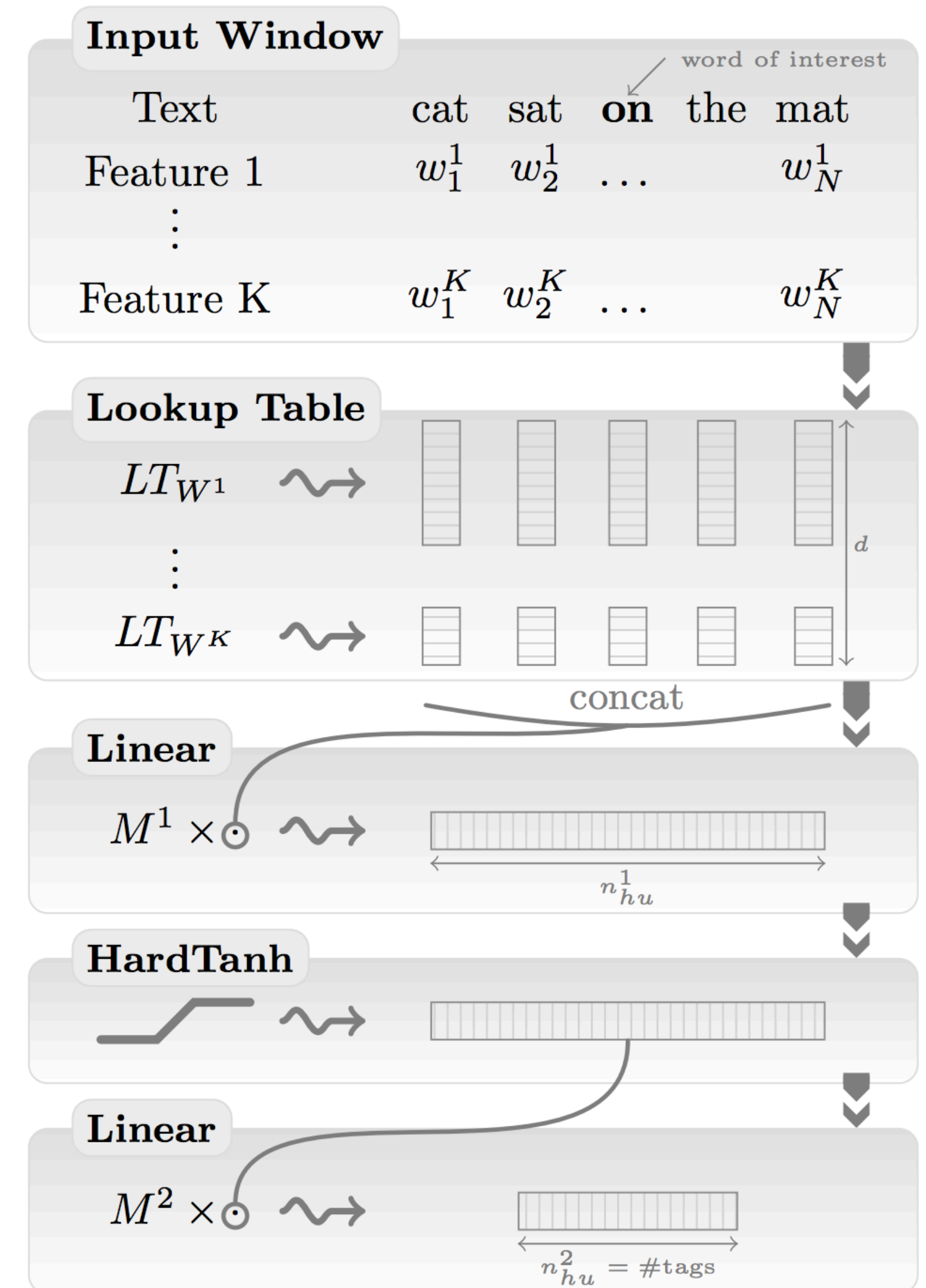
- ▶ How does this compare to neural CRF?

“NLP (Almost) From Scratch”



“NLP (Almost) From Scratch”

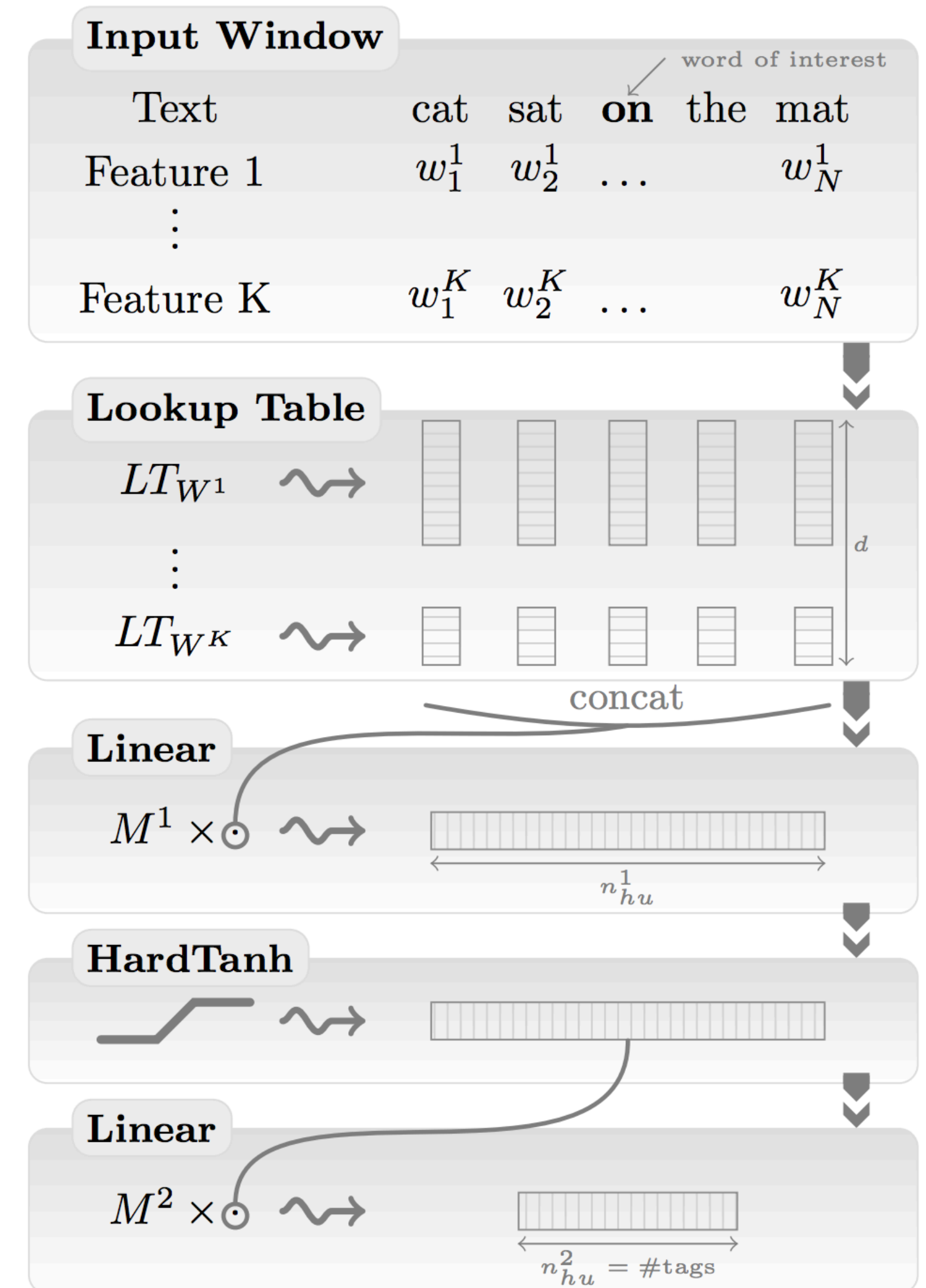
Approach	POS (PWA)	CHUNK (F1)	NER (F1)	SRL (F1)
Benchmark Systems	97.24	94.29	89.31	77.92
NN+WLL	96.31	89.13	79.53	55.40
NN+SLL	96.37	90.33	81.47	70.99
NN+WLL+LM1	97.05	91.91	85.68	58.18
NN+SLL+LM1	97.10	93.65	87.58	73.84
NN+WLL+LM2	97.14	92.04	86.96	58.34
NN+SLL+LM2	97.20	93.63	88.67	74.15



“NLP (Almost) From Scratch”

Approach	POS (PWA)	CHUNK (F1)	NER (F1)	SRL (F1)
Benchmark Systems	97.24	94.29	89.31	77.92
NN+WLL	96.31	89.13	79.53	55.40
NN+SLL	96.37	90.33	81.47	70.99
NN+WLL+LM1	97.05	91.91	85.68	58.18
NN+SLL+LM1	97.10	93.65	87.58	73.84
NN+WLL+LM2	97.14	92.04	86.96	58.34
NN+SLL+LM2	97.20	93.63	88.67	74.15

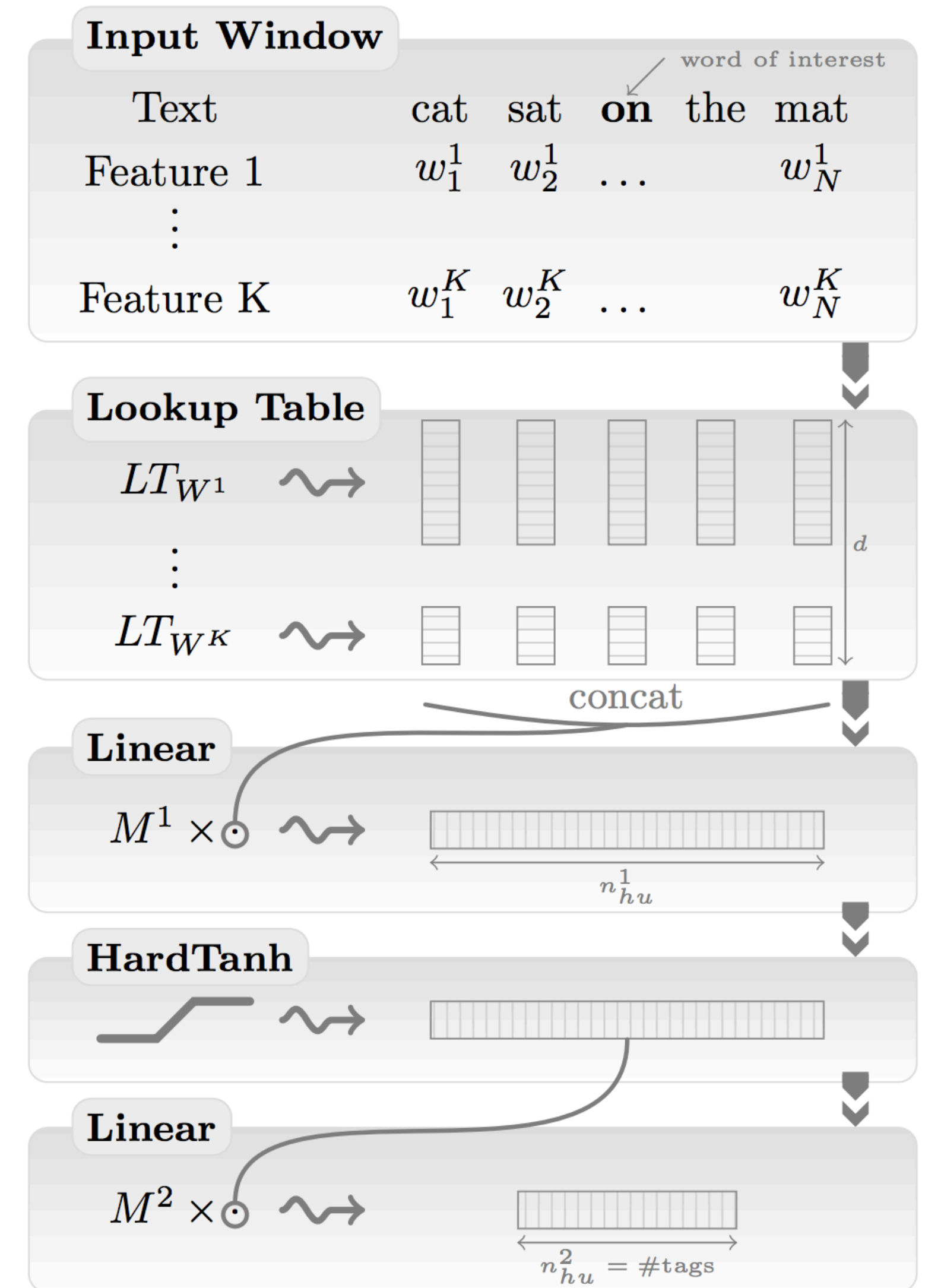
- ▶ WLL: independent classification; SLL: neural CRF



“NLP (Almost) From Scratch”

Approach	POS (PWA)	CHUNK (F1)	NER (F1)	SRL (F1)
Benchmark Systems	97.24	94.29	89.31	77.92
NN+WLL	96.31	89.13	79.53	55.40
NN+SLL	96.37	90.33	81.47	70.99
NN+WLL+LM1	97.05	91.91	85.68	58.18
NN+SLL+LM1	97.10	93.65	87.58	73.84
NN+WLL+LM2	97.14	92.04	86.96	58.34
NN+SLL+LM2	97.20	93.63	88.67	74.15

- ▶ WLL: independent classification; SLL: neural CRF
- ▶ LM2: word vectors learned from a precursor to word2vec/GloVe, trained for 2 weeks (!) on Wikipedia



CNN Neural CRFs

travel to Hangzhou today for

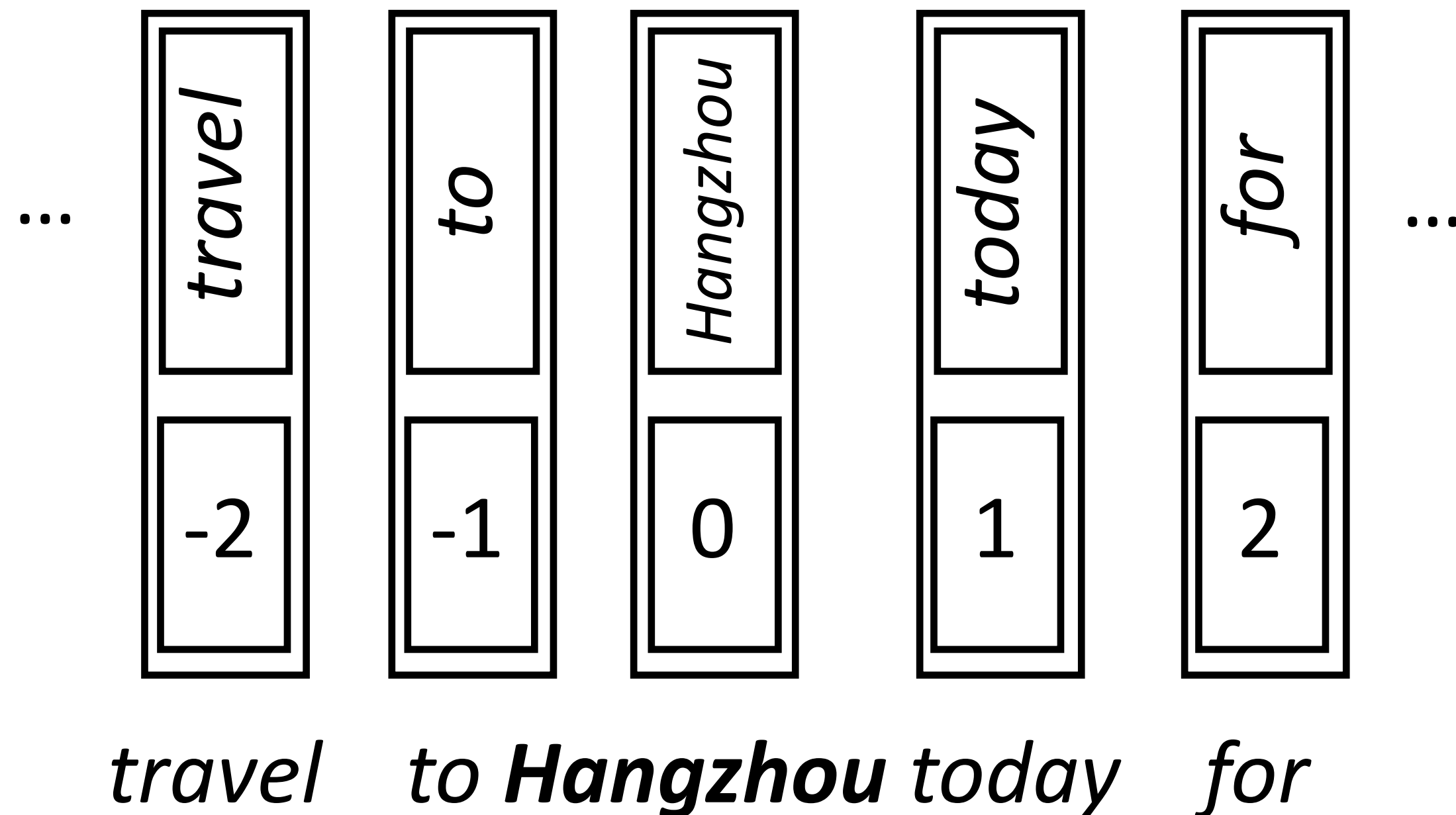
CNN Neural CRFs

- ▶ Append to each word vector an *embedding of the relative position of that word*

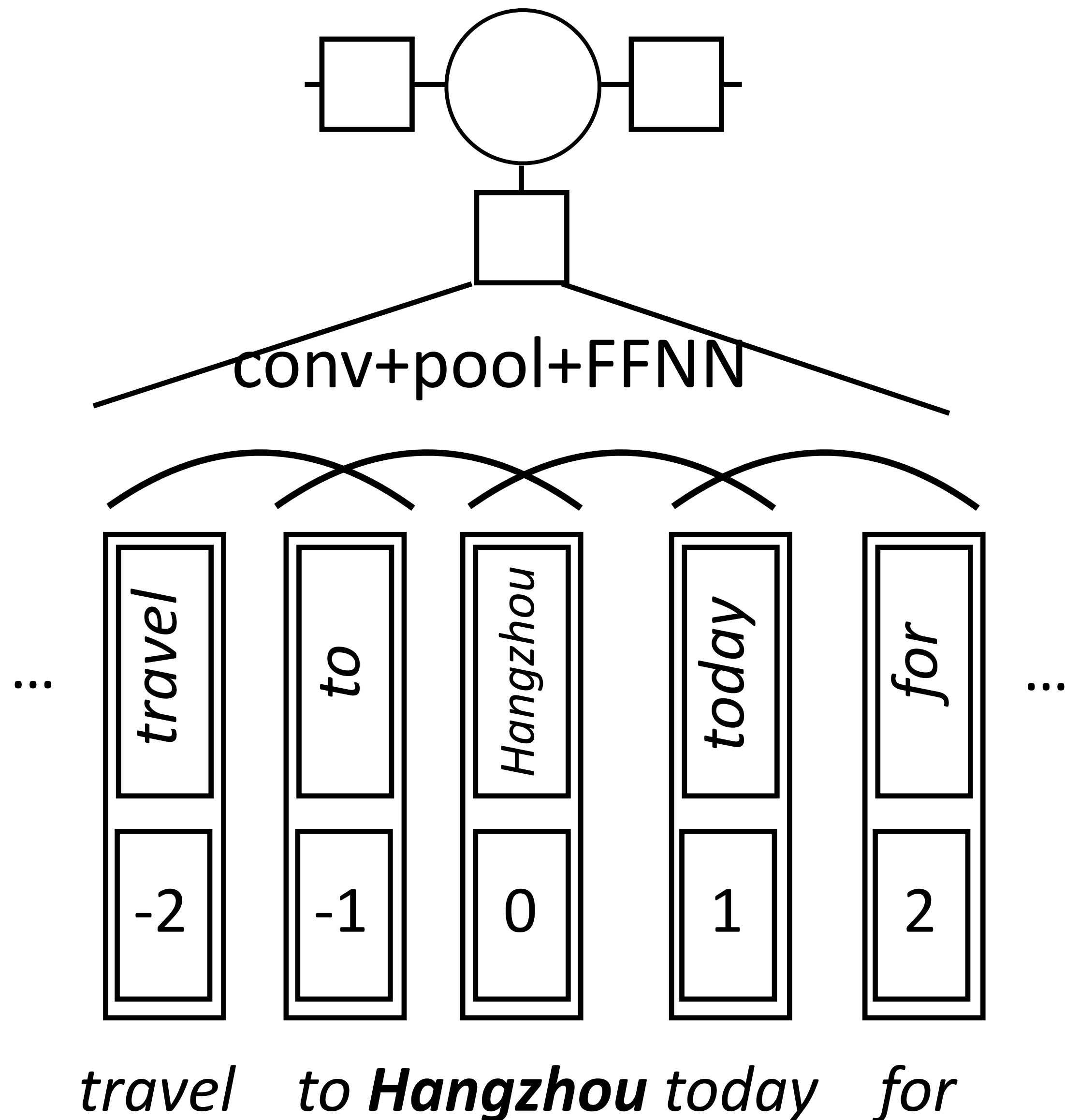
travel to Hangzhou today for

CNN Neural CRFs

- ▶ Append to each word vector an *embedding of the relative position of that word*

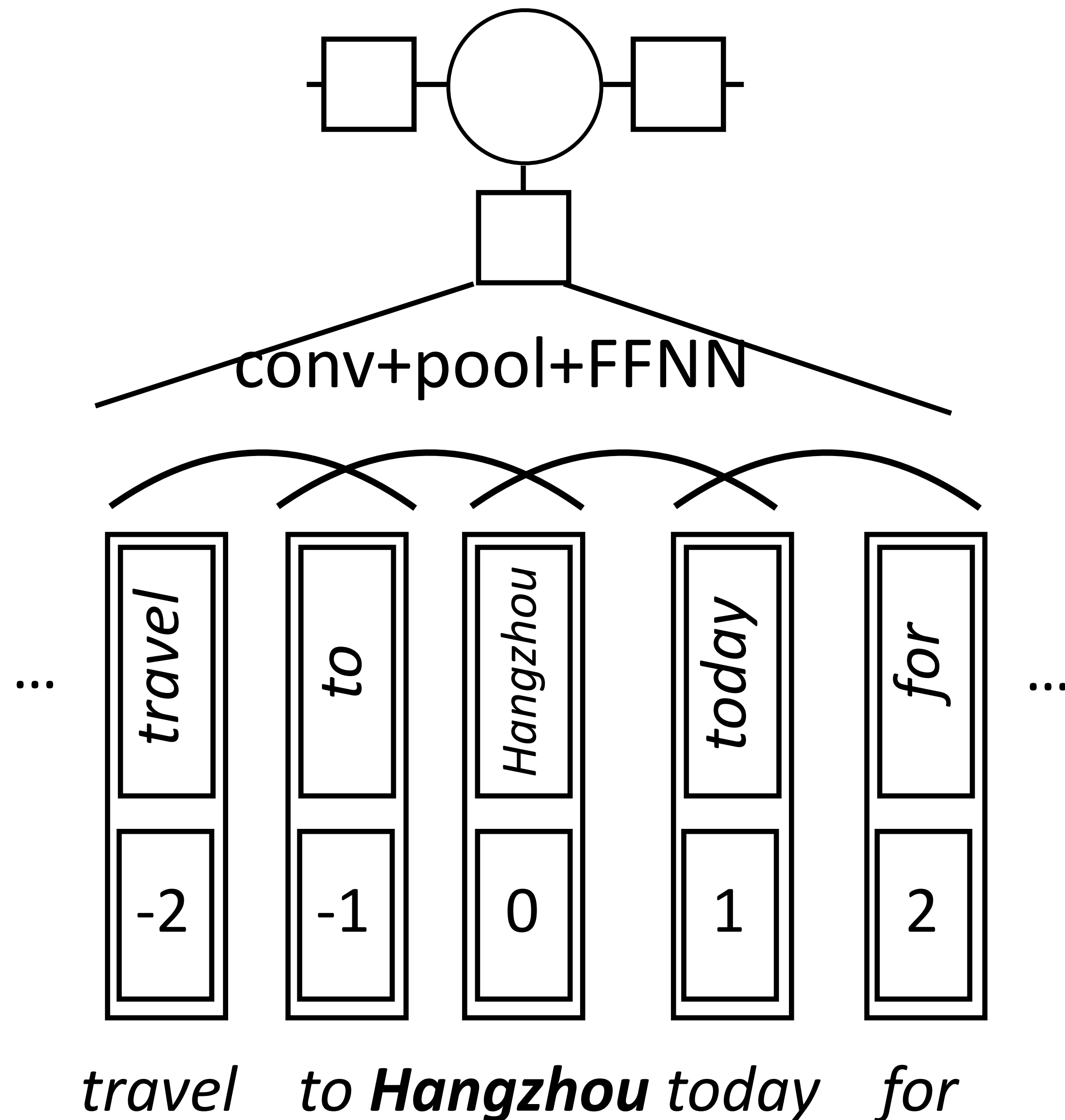


CNN Neural CRFs



- ▶ Append to each word vector an *embedding of the relative position* of that word

CNN Neural CRFs



- ▶ Append to each word vector an *embedding of the relative position* of that word
- ▶ Convolution over the sentence produces a position-dependent representation

CNN NCRFs vs. FFNN NCRFs

Approach	POS (PWA)	CHUNK (F1)	NER (F1)	SRL (F1)
Benchmark Systems	97.24	94.29	89.31	77.92
	<i>Window Approach</i>			
NN+SLL+LM2	97.20	93.63	88.67	–
	<i>Sentence Approach</i>			
NN+SLL+LM2	97.12	93.37	88.78	74.15

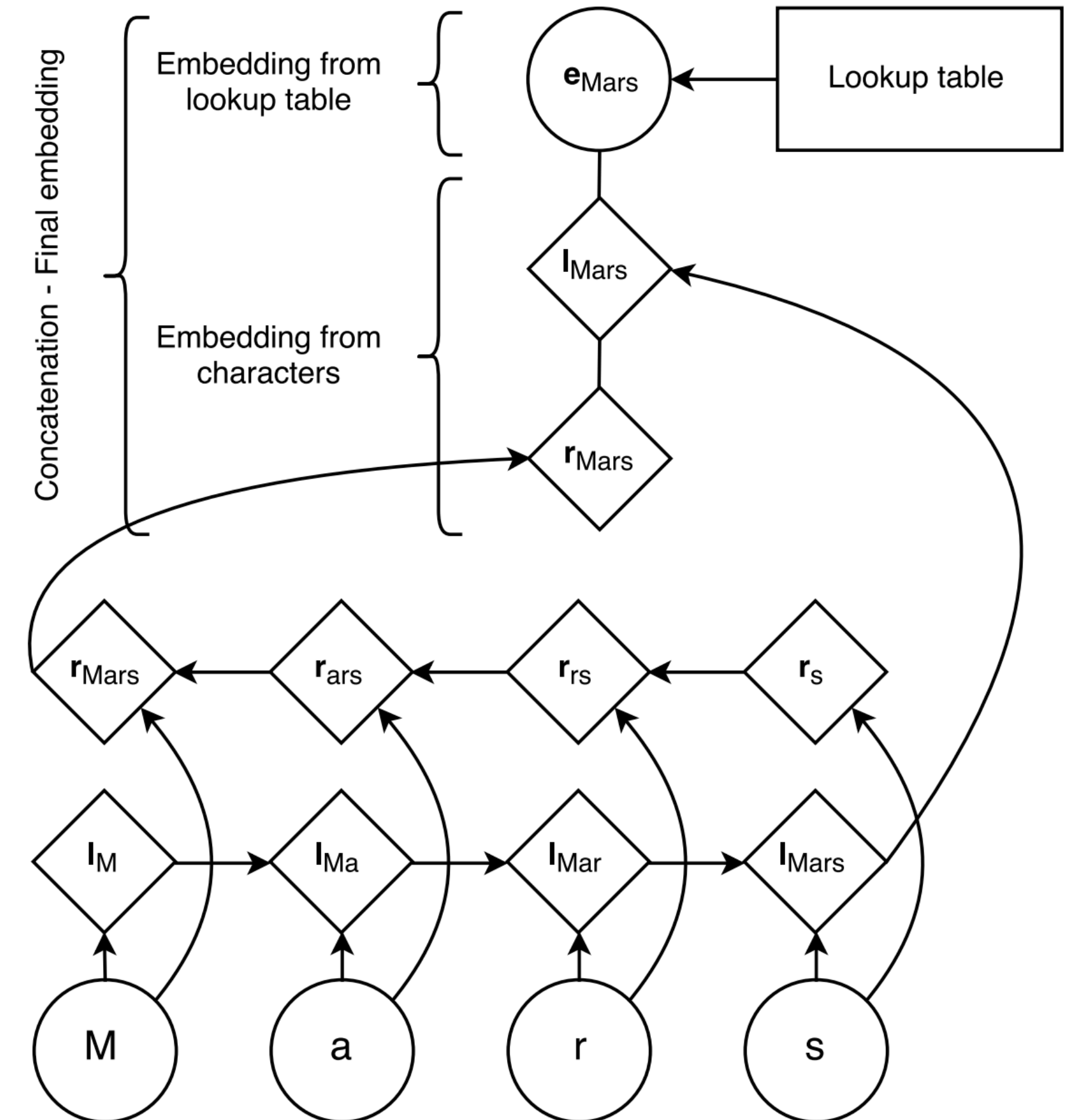
- ▶ Sentence approach (CNNs) is comparable to window approach (FFNNs) except for SRL where they claim it works much better

Neural CRFs with LSTMs

- ▶ Neural CRF using character LSTMs to compute word representations

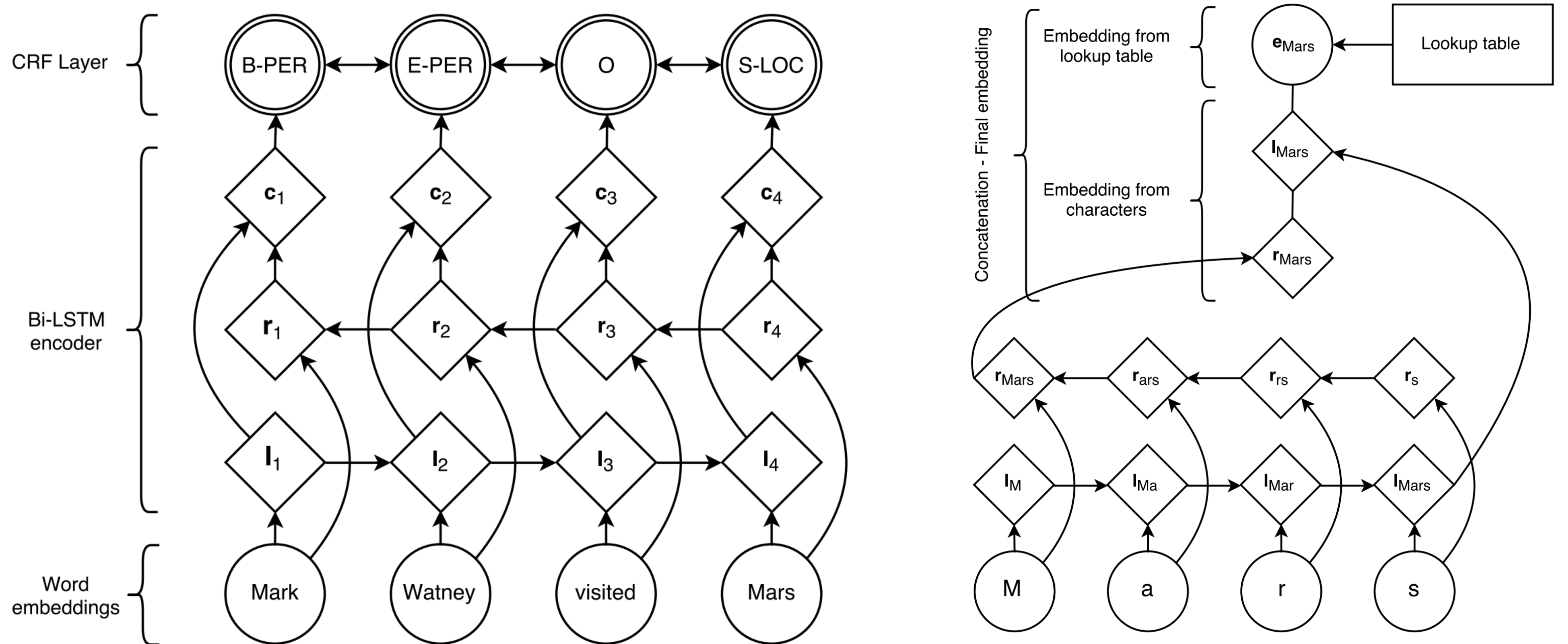
Neural CRFs with LSTMs

- ▶ Neural CRF using character LSTMs to compute word representations



Neural CRFs with LSTMs

- Neural CRF using character LSTMs to compute word representations



Chiu and Nichols (2015), Lample et al. (2016)

Neural CRFs with LSTMs

- ▶ Chiu+Nichols: character CNNs instead of LSTMs
- ▶ Lin/Passos/Luo: use external resources like Wikipedia
- ▶ LSTM-CRF captures the important aspects of NER: word context (LSTM), sub-word features (character LSTMs), outside knowledge (word embeddings)

Model	F ₁
Collobert et al. (2011)*	89.59
Lin and Wu (2009)	83.78
Lin and Wu (2009)*	90.90
Huang et al. (2015)*	90.10
Passos et al. (2014)	90.05
Passos et al. (2014)*	90.90
Luo et al. (2015)* + gaz	89.9
Luo et al. (2015)* + gaz + linking	91.2
Chiu and Nichols (2015)	90.69
Chiu and Nichols (2015)*	90.77
LSTM-CRF (no char)	90.20
LSTM-CRF	90.94

Takeaways

- ▶ CNNs are a flexible way of extracting features analogous to bag of n-grams, can also encode positional information
- ▶ All kinds of NNs can be integrated into CRFs for structured inference. Can be applied to NER, other tagging, parsing, ...