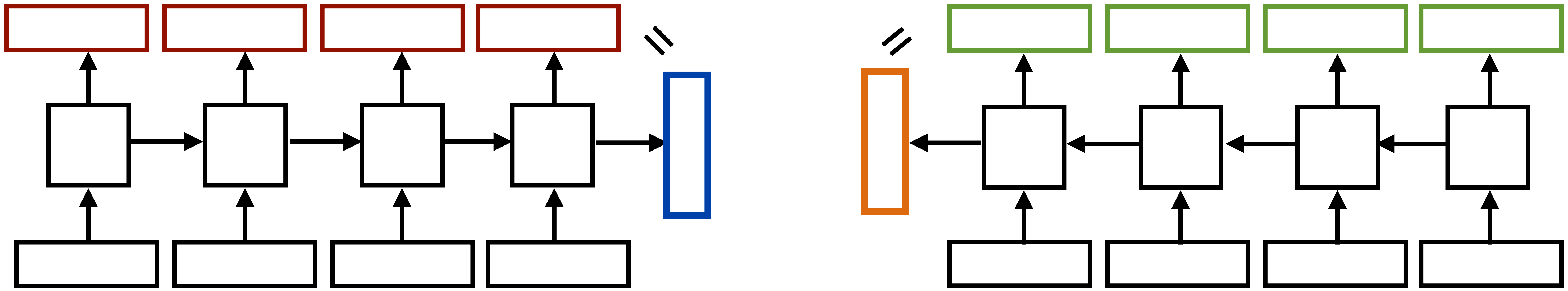
CNNs & Neural CRFs

Wei Xu

(many slides from Greg Durrett, Stanford 231n)

Recap — What do RNNs produce?



- RNN can be viewed as a transformation of a sequence of vectors into a sequence of context-dependent vectors

This Lecture

- Neural CRFs
- CNNs
- CNNs for Sentiment, Entity Linking

Administrivia

- Reading — Goldberg 9 (CNN); Eisenstein 3.4, 7.6

A Primer on Neural Network Models for Natural Language Processing

Yoav Goldberg
Draft as of October 5, 2015.

The most up-to-date version of this manuscript is available at `http://www.cs.biu.ac.il/~yogo/nnlp.pdf`. Major updates will be published on arxiv periodically.
I welcome any comments you may have regarding the content and presentation. If you spot a missing reference or have relevant work you'd like to see mentioned, do let me know.
`first.last@gmail`

Abstract

Over the past few years, neural networks have re-emerged as powerful machine-learning models, yielding state-of-the-art results in fields such as image recognition and speech processing. More recently, neural network models started to be applied also to textual natural language signals, again with very promising results. This tutorial surveys neural network models from the perspective of natural language processing research, in an attempt to bring natural-language researchers up to speed with the neural techniques. The tutorial covers input encoding for natural language tasks, feed-forward networks, convolutional networks, recurrent networks and recursive networks, as well as the computation graph abstraction for automatic gradient computation.

Neural CRF

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

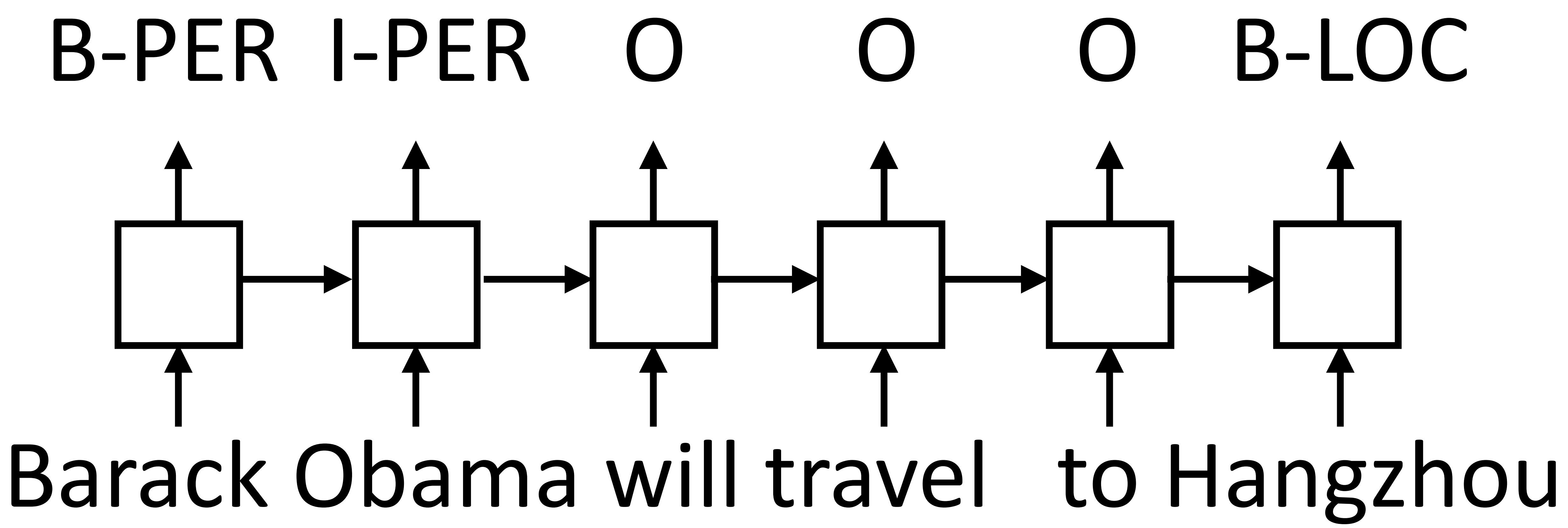
- Features in CRFs: I[tag=B-LOC & curr_word=*Hangzhou*], I[tag=B-LOC & prev_word=*to*], I[tag=B-LOC & curr_prefix=*Han*]
- Linear model over features
- Downsides:
  - Lexical features mean that words need to be seen in the training data
  - Linear model can't capture feature conjunctions as effectively (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



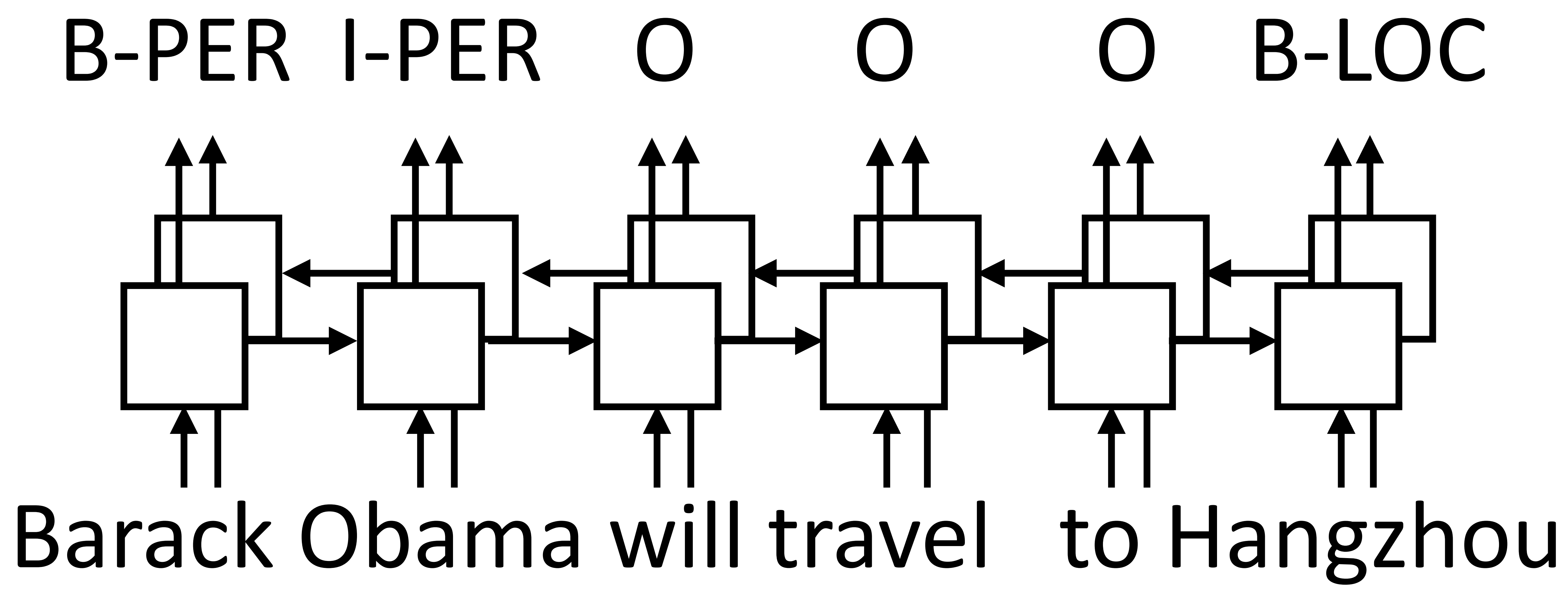
- Transducer (LM-like model)
- Q1: What are the strengths and weaknesses of this model compared to the linear CRFs?

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

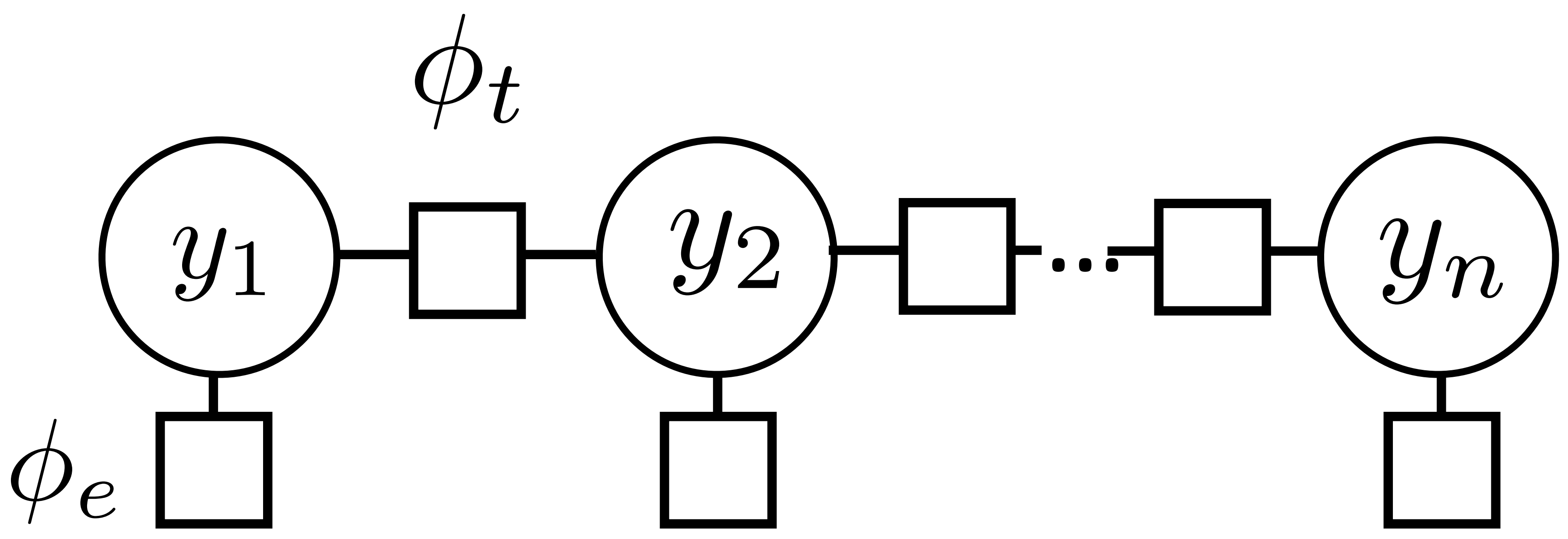


- Bidirectional transducer model
- Q2: What are the strengths and weaknesses of this model compared to the linear CRFs?

Recall: Sequential CRFs

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

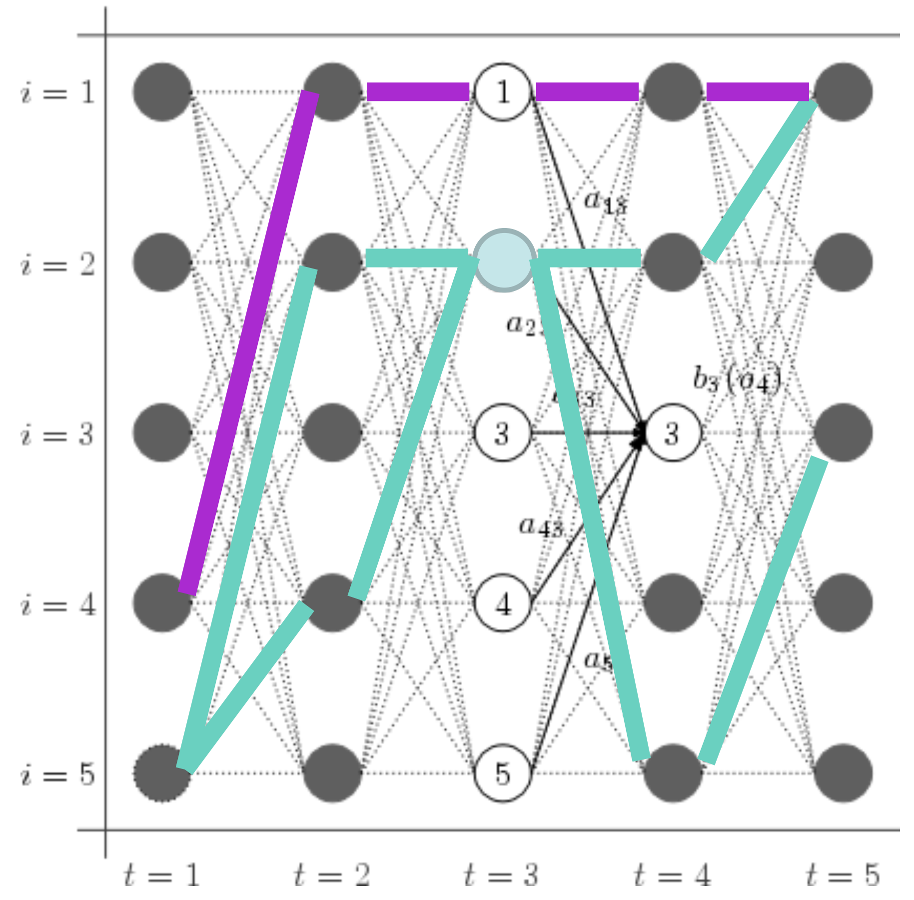
- Normalizing constant $$Z = \sum_{\mathbf{y}}\prod_{i=2}^{n}\exp(\phi_t(y_{i-1}, y_i))\prod_{i=1}^{n}\exp(\phi_e(y_i, i, \mathbf{x}))$$



Recall — Sequential CRFs

- Model: $$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z}\prod_{i=2}^{n}\exp(\phi_t(y_{i-1}, y_i))\prod_{i=1}^{n}\exp(\phi_e(y_i, i, \mathbf{x}))$$
- Inference: argmax P(**y**|**x**) from Viterbi
- Learning: run forward-backward to compute marginals

$$P(y_i = s|\mathbf{x}) = \sum_{y_1,\ldots,y_{i-1},y_{i+1},\ldots,y_n} P(\mathbf{y}|\mathbf{x})$$



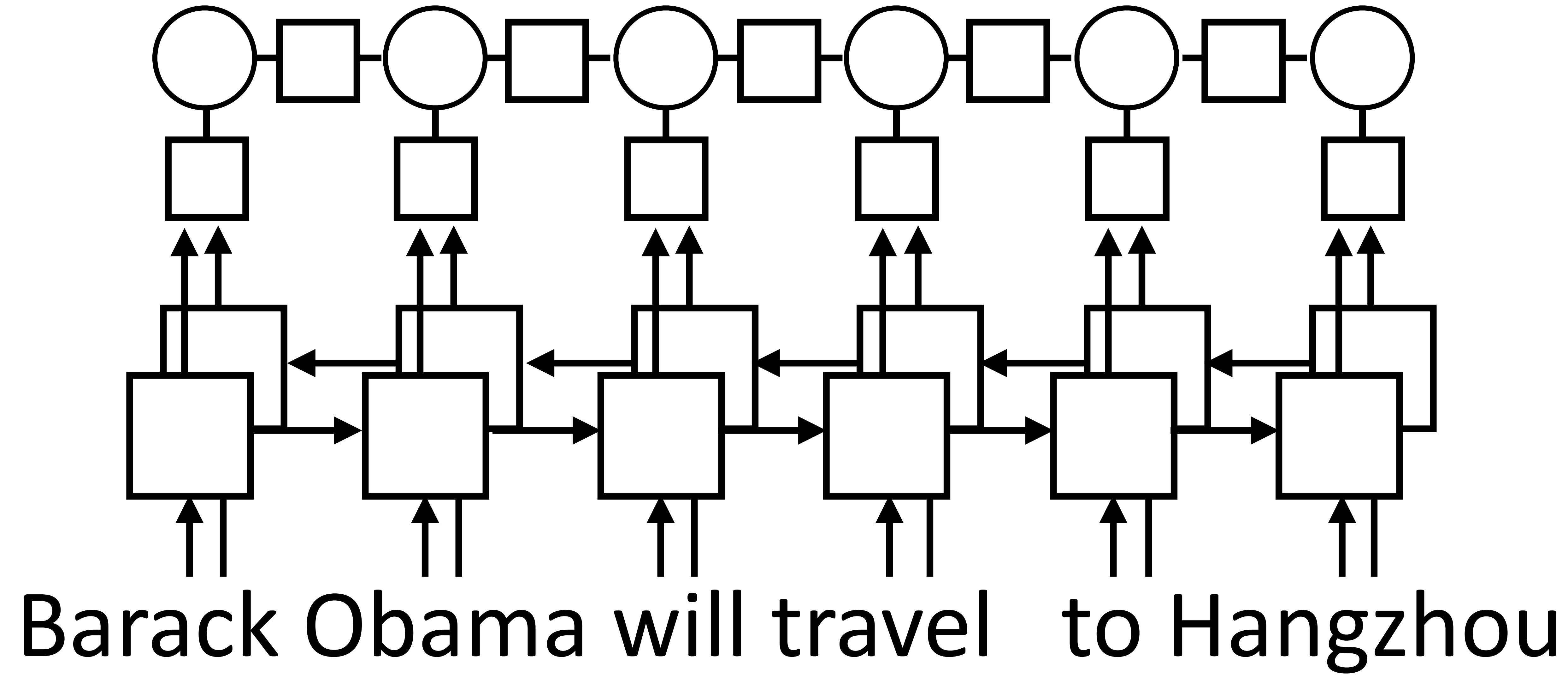
$P(y_i = s_1, y_{i+1} = s_2|\mathbf{x})$, then update gradient

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 .

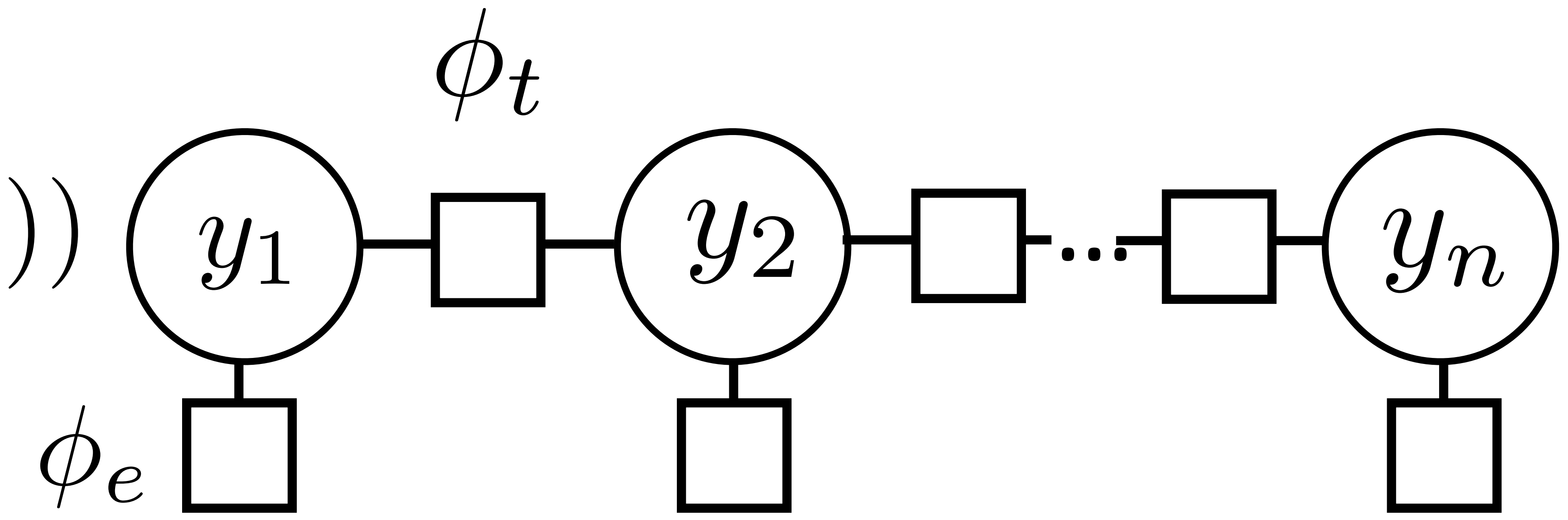
PERSON LOC ORG



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

Neural CRFs

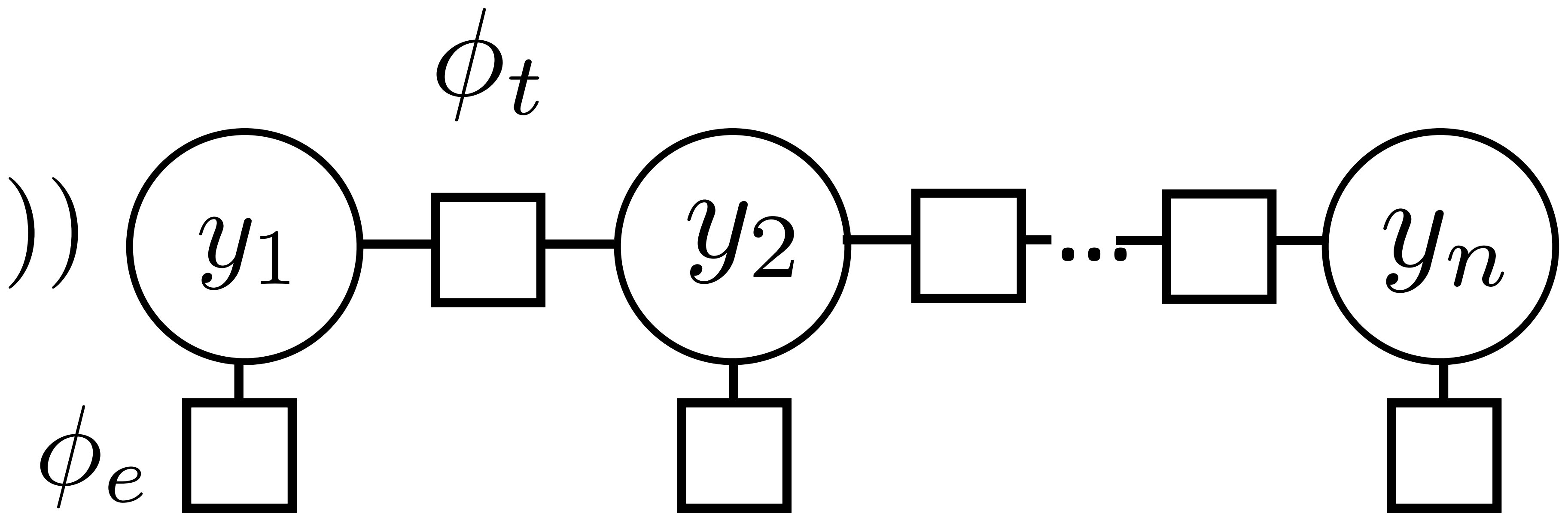
$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z}\prod_{i=2}^{n}\exp(\phi_t(y_{i-1}, y_i))\prod_{i=1}^{n}\exp(\phi_e(y_i, i, \mathbf{x}))$$



- Linear model: $\phi_e(y_i, i, \mathbf{x}) = w^\top f_e(y_i, i, \mathbf{x})$
- 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*, ***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

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



- 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

chain rule say to multiply together, gives our update

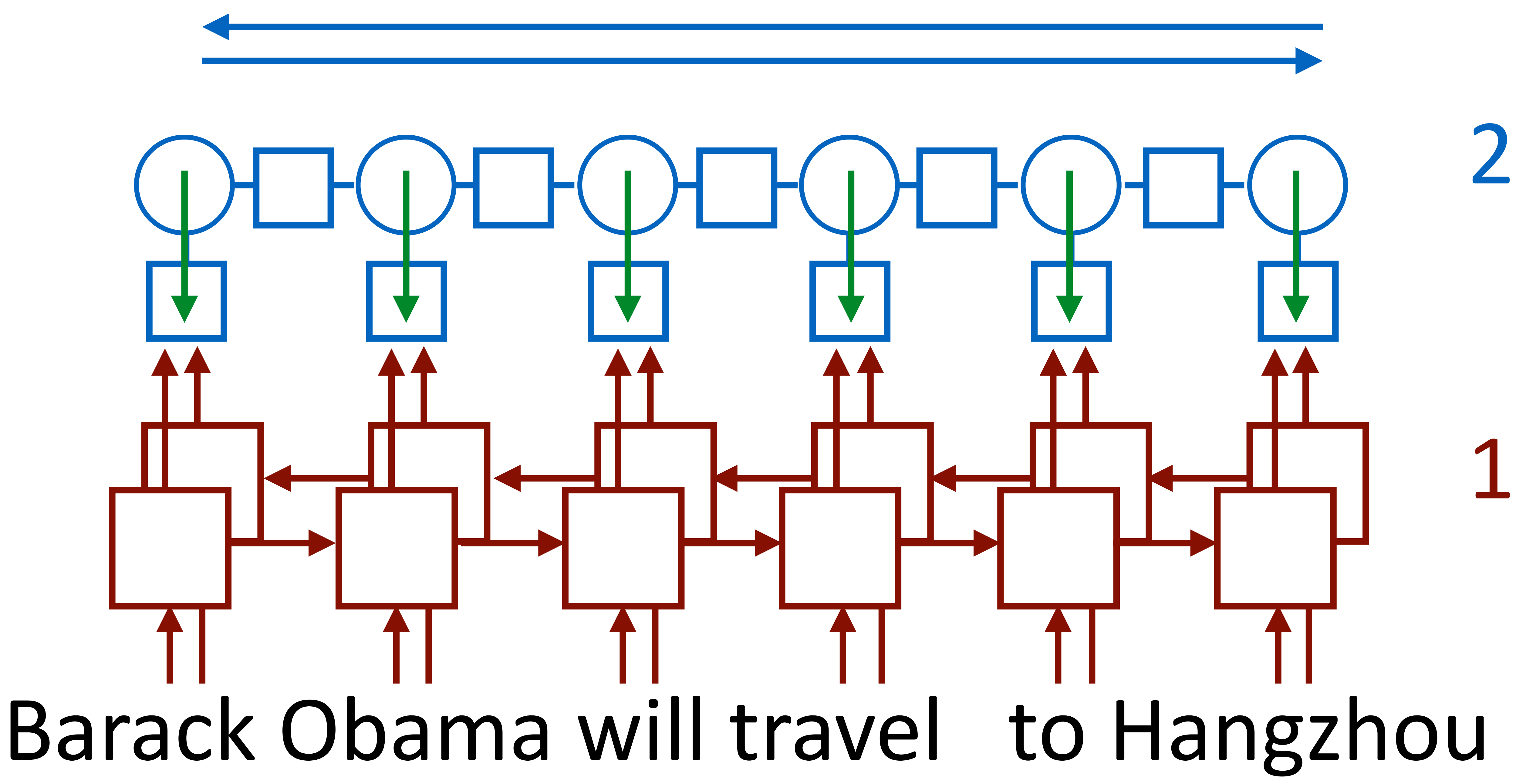
- For linear model: $\frac{\partial \phi_{e,i}}{w_i} = f_{e,i}(y_i, i, \mathbf{x})$
- For neural model: compute gradient of phi w.r.t. parameters of neural net

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



2) Run forward-backward

3) Compute error signal

1) Compute f(**x**)

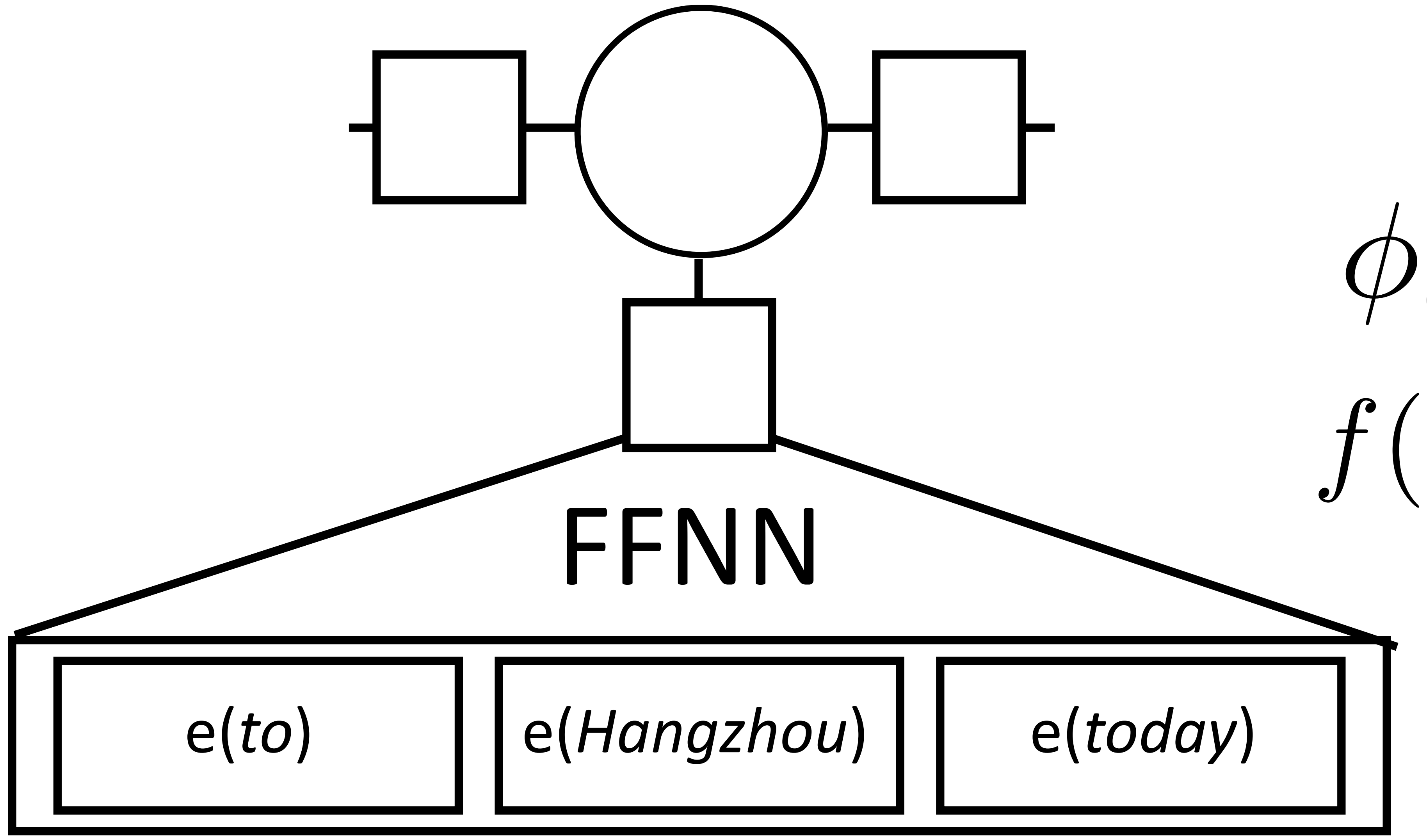
4) Backprop (no knowledge of sequential structure required)

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



previous word curr word next word

to ***Hangzhou*** *today*

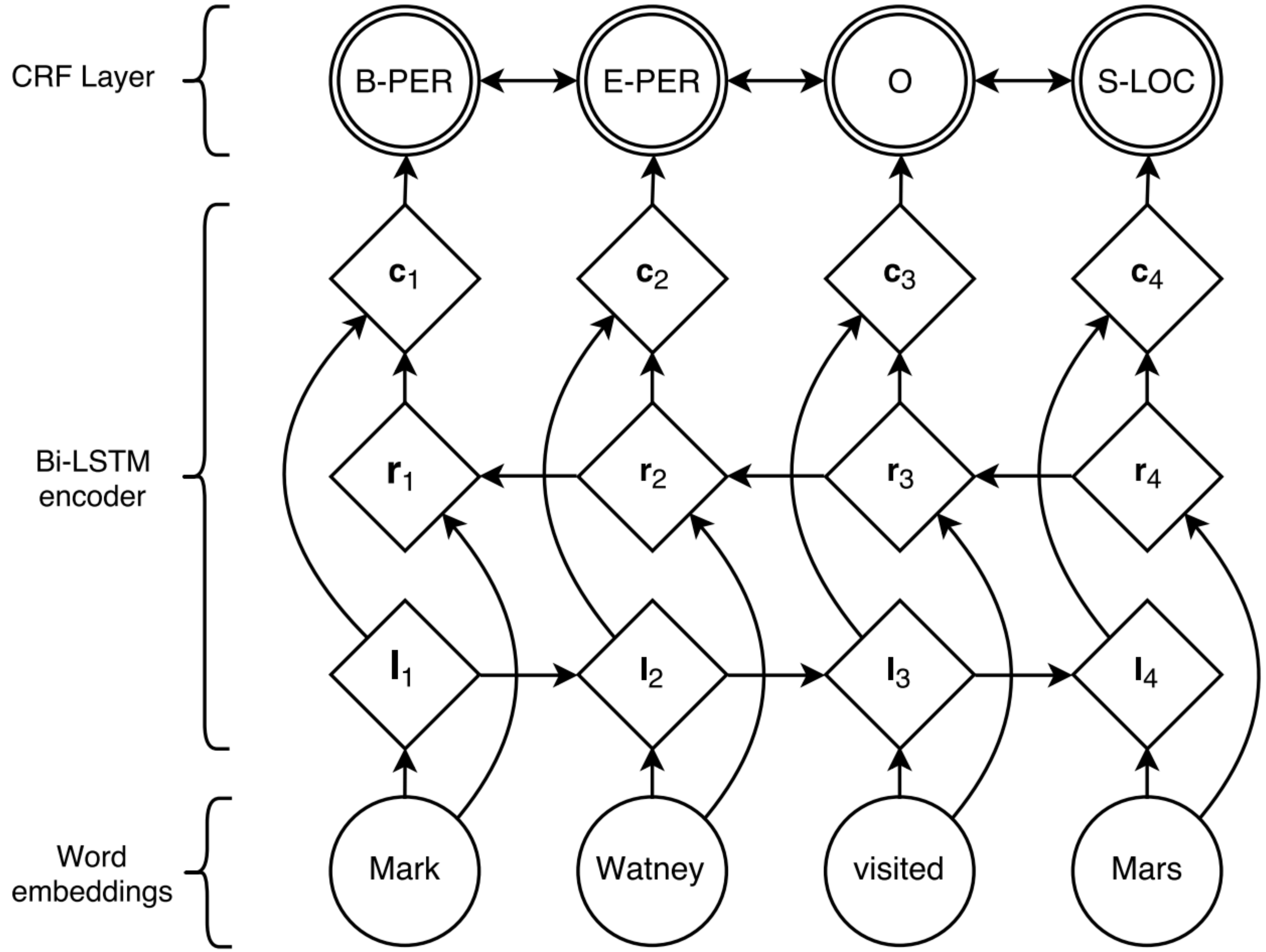
$$\phi_e = Wg(Vf(\mathbf{x}, i))$$

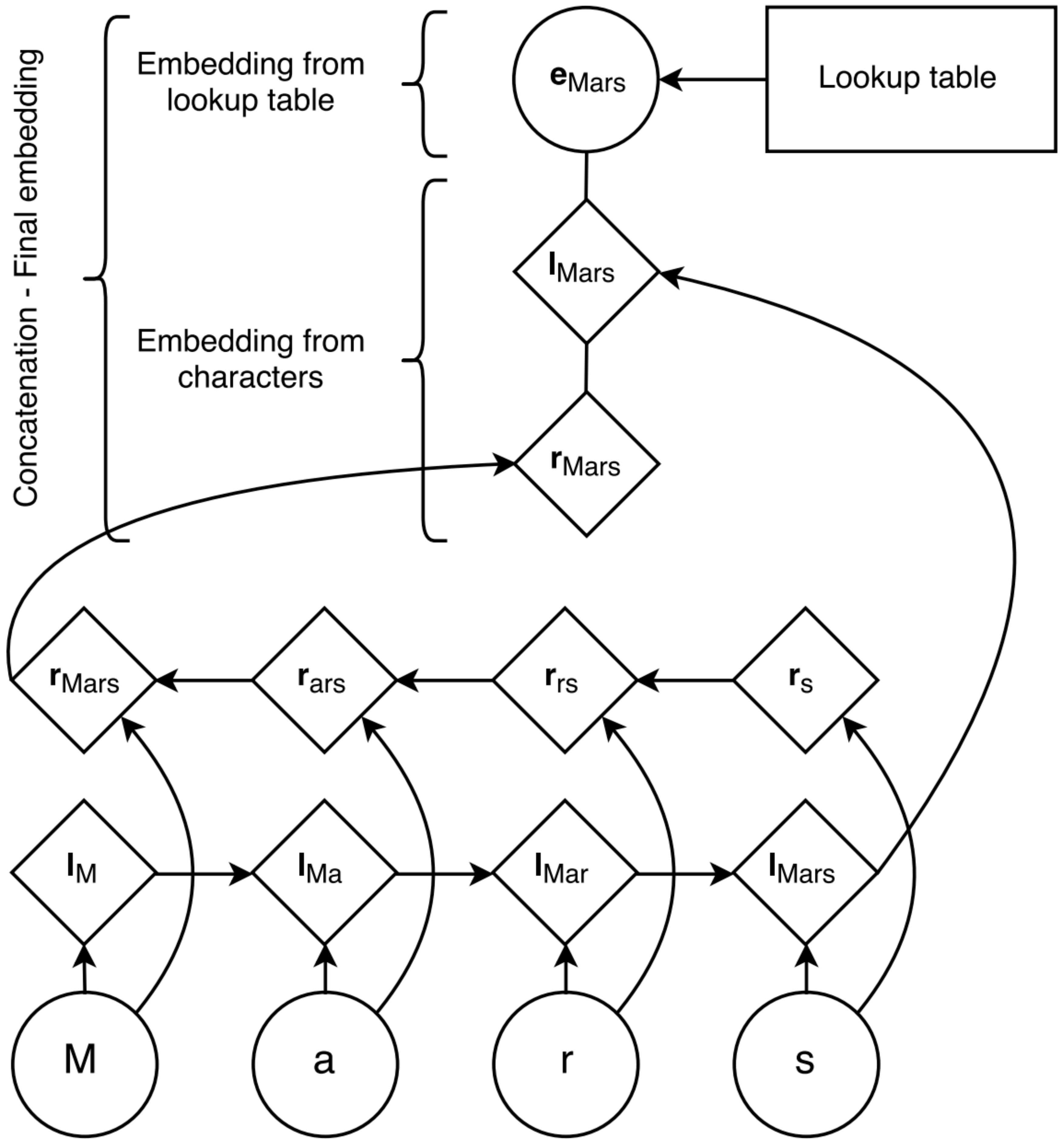
$$f(\mathbf{x}, i) = [\mathrm{emb}(\mathbf{x}_{i-1}), \mathrm{emb}(\mathbf{x}_i), \mathrm{emb}(\mathbf{x}_{i+1})]$$

Applications

Neural CRFs with LSTMs

- Neural CRF using character LSTMs to compute word representations





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

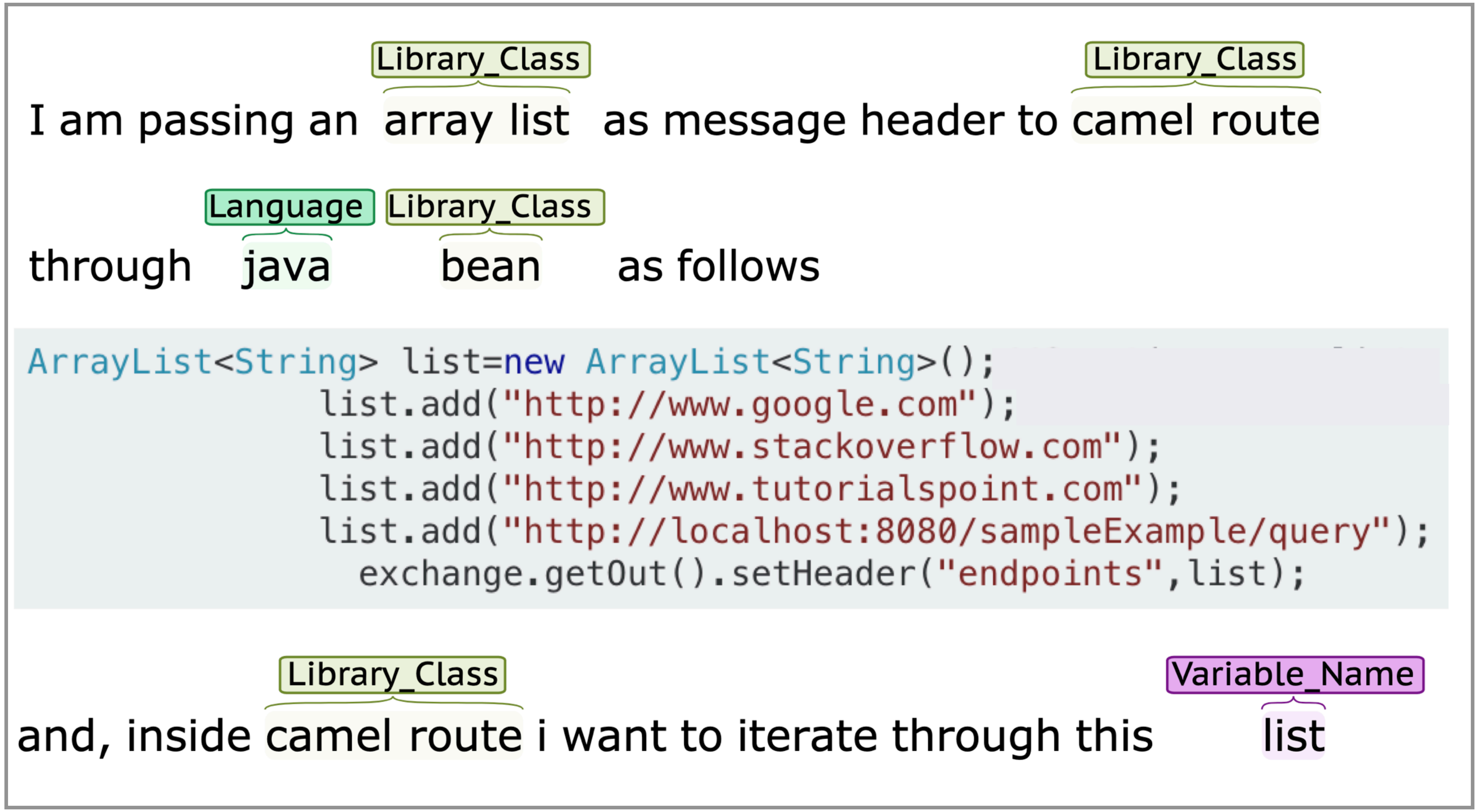
Neural CRFs with LSTMs

- 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	$\mathbf{F_1}$
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**

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

NER in StackOverflow



Jeniya Tabassum, Mounica Maddela, Alan Ritter, Wei Xu. “Code and Named Entity Recognition in StackOverflow” (ACL 2020)

NER in StackOverflow

StackOverflow NER Corpus

stackoverflow → 1237 Question-Answer Threads

Archive timeline: (2008 - 2018)

- → Most upvoted answer
- → Accepted answer
- → 2 randomly selected answer

15,372 sentences

20 Entity Types

Code Entity Types

Class, Value, Function, HTML_XML_Tag, In_Line_code, Data_Type, Variable, Library

Natural Language Entity

Algorithm, Application, Data_Structure, File_Type, Version, Language, Website, File_Name, Operating_System, User_Name

Jeniya Tabassum, Mounica Maddela, Alan Ritter, Wei Xu. "Code and Named Entity Recognition in StackOverflow" (ACL 2020)

NER in StackOverflow

Two Main Challenges

(1) Polysemy — e.g., “key”, “windows”.
(2) Inline code — code-switch between human and programming languages.

Before adding element to array, check if key is numeric is is_numeric($key) function. If it return false, then, covert key to integer using typecasting, (int)$key.

Now, the array will have numeric keys only and can be ordered.

share improve this answer follow

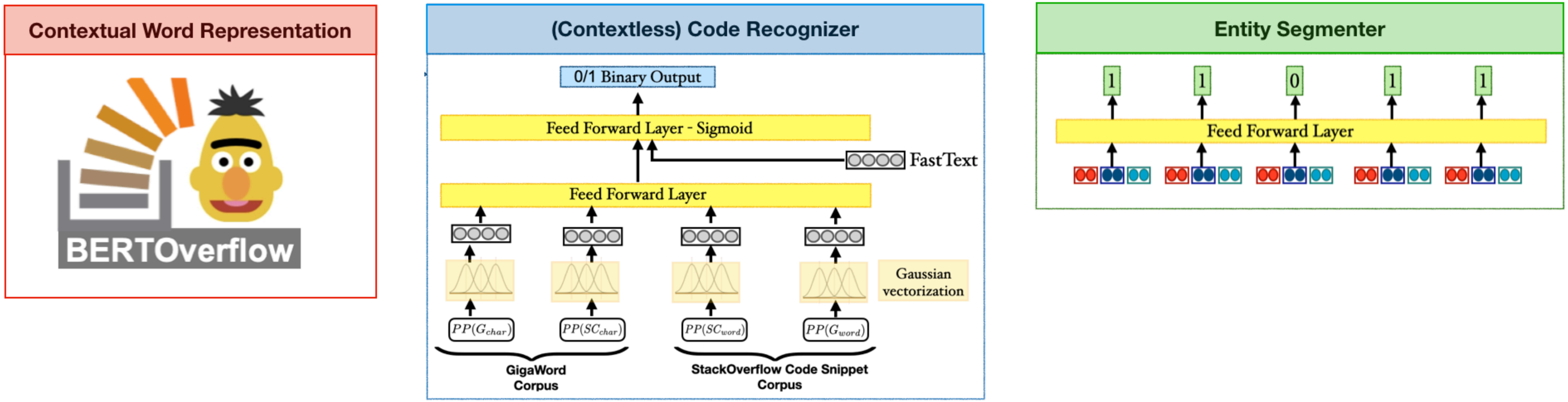
answered Oct 23 '15 at 9:16
Ravneet
300 ● 1 ● 5

Jeniya Tabassum, Mounica Maddela, Alan Ritter, Wei Xu. “Code and Named Entity Recognition in StackOverflow” (ACL 2020)

NER in StackOverflow

SoftNER Model

Combines BERTOverflow with domain-specific embeddings (Code Recognizer & Entity Segmenter) via attention.

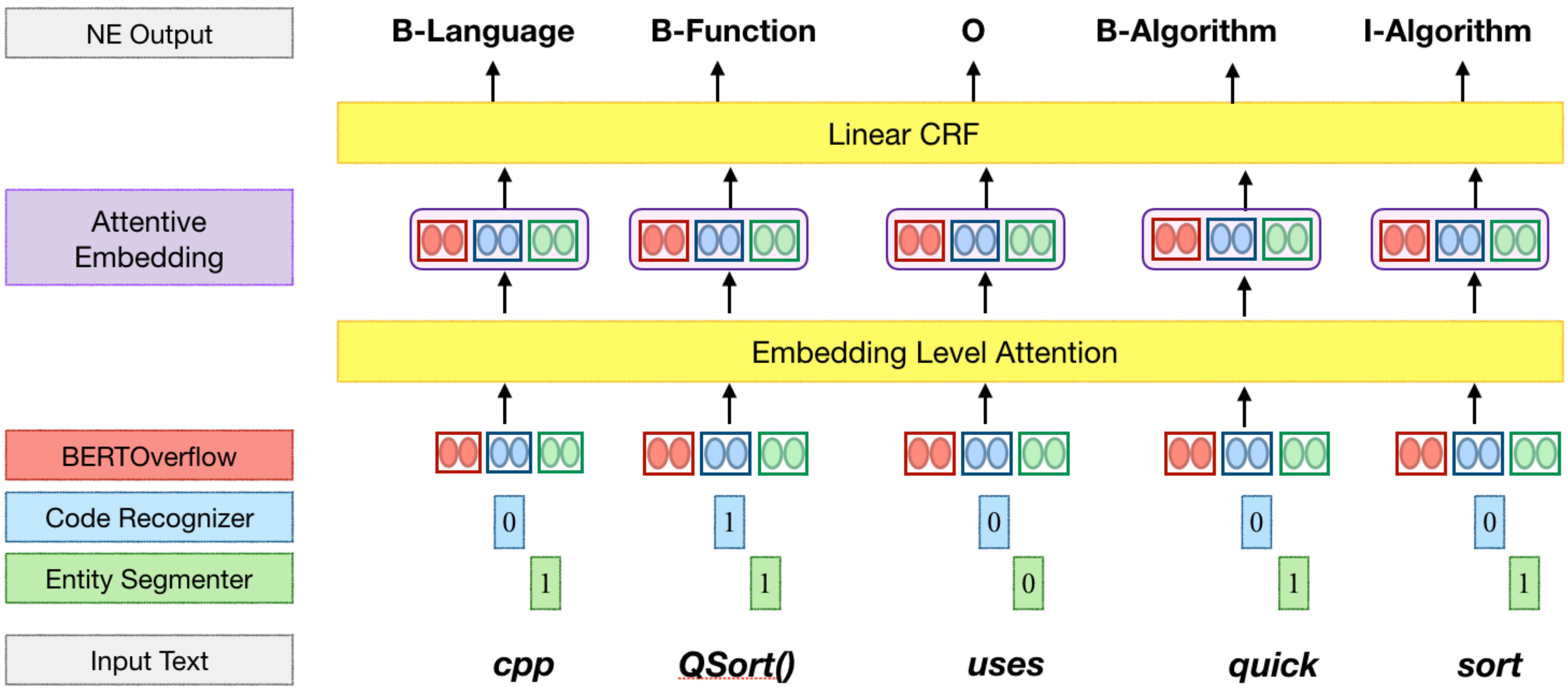


Jeniya Tabassum, Mounica Maddela, Alan Ritter, Wei Xu. "Code and Named Entity Recognition in StackOverflow" (ACL 2020)

NER in StackOverflow

SoftNER Model

Combines BERTOverflow with domain-specific embeddings (Code Recognizer & Entity Segmenter) via attention.



Jeniya Tabassum, Mounica Maddela, Alan Ritter, Wei Xu. "Code and Named Entity Recognition in StackOverflow" (ACL 2020)

Semantic Role Labeling

- Find out 5W in text — “who did what to whom, when and where”
- Identify predicate, disambiguate it, identify that predicate’s arguments

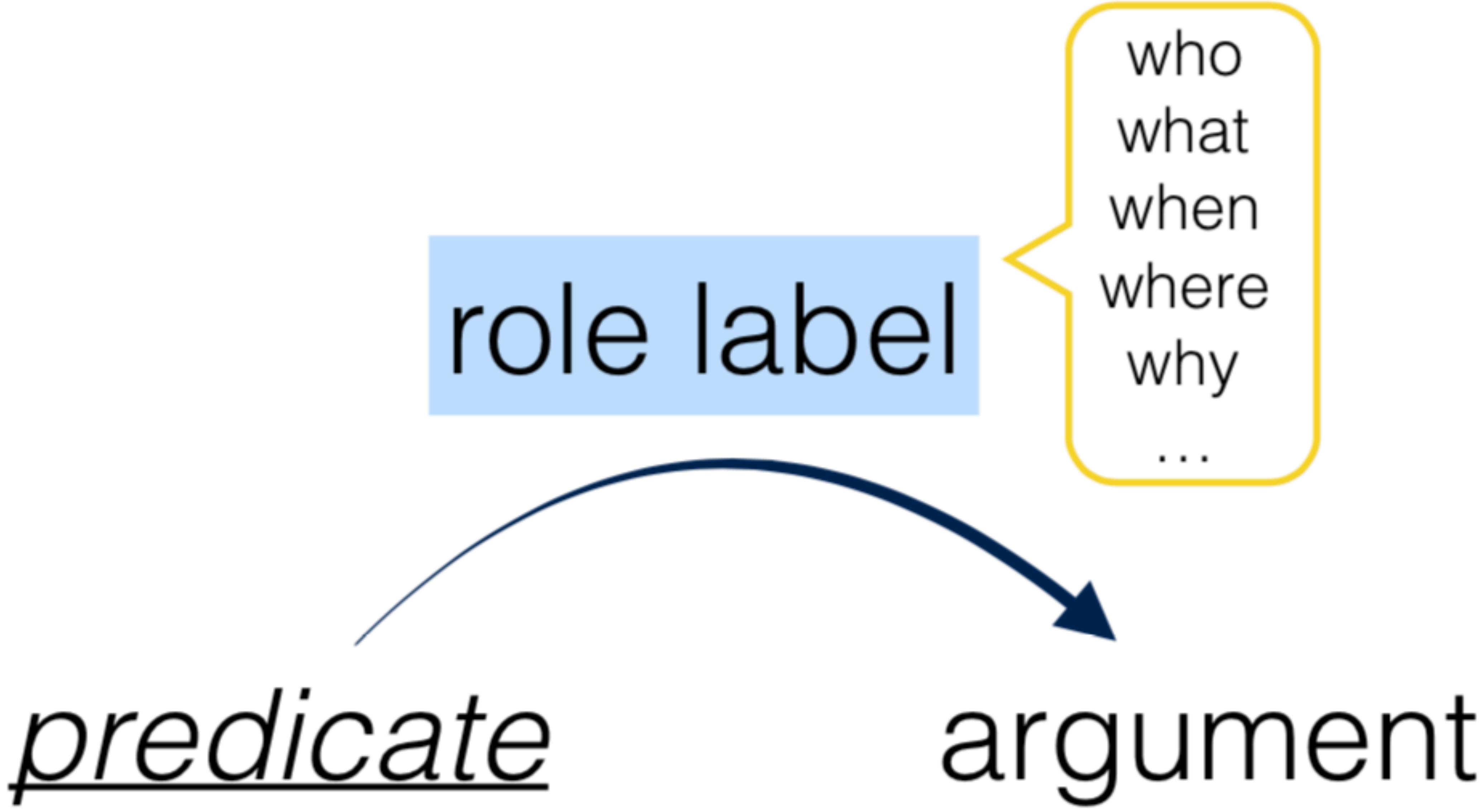


Figure from He et al. (2017)

Semantic Role Labeling

The robot *broke* my favorite mug with a wrench.

My mug *broke* into pieces immediately.

Figure from He et al. (2017)

Semantic Role Labeling

The robot *broke* my favorite mug with a wrench.

My mug *broke* into pieces immediately.

Figure from He et al. (2017)

Semantic Role Labeling

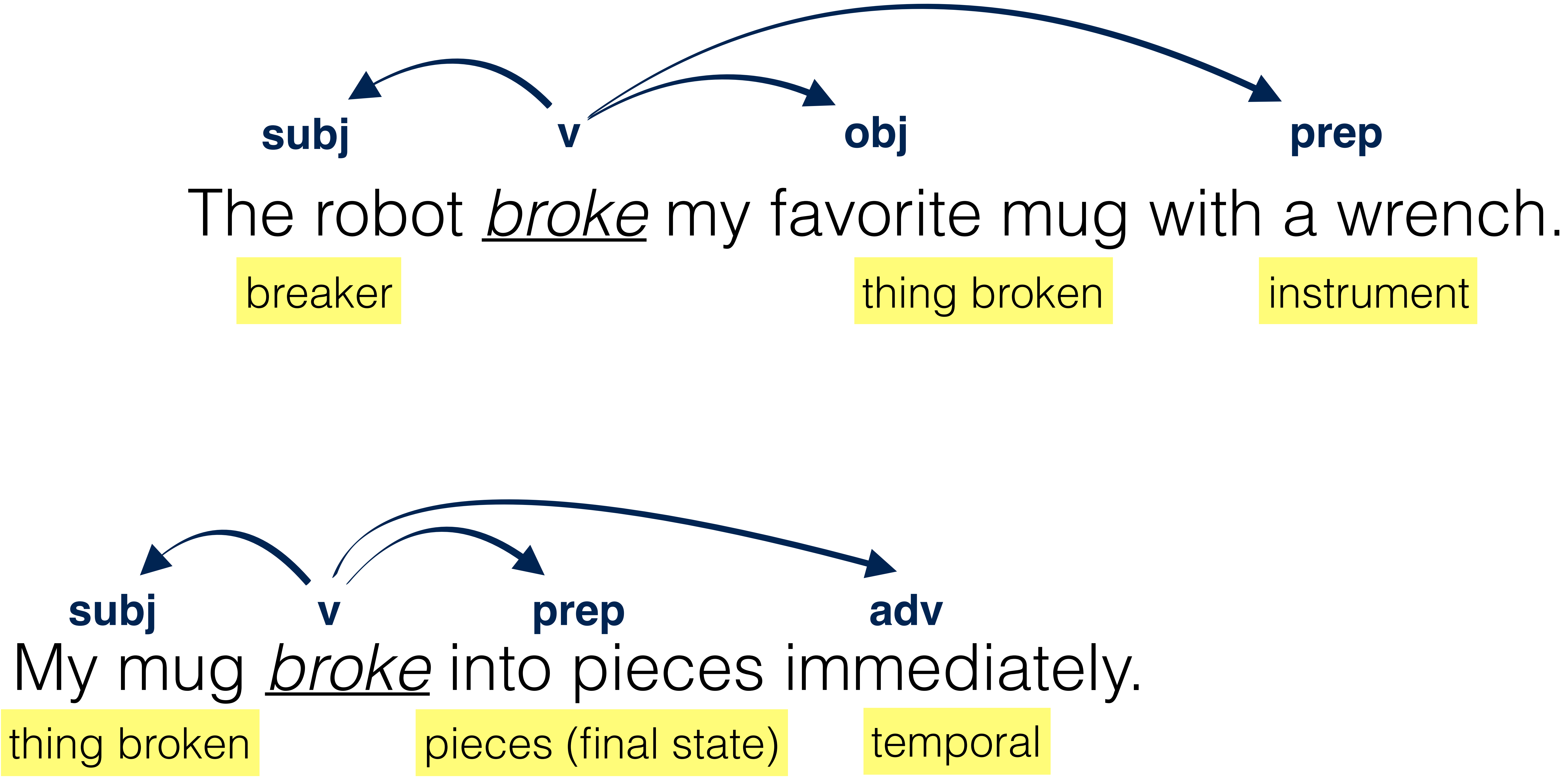


Figure from He et al. (2017)

Semantic Role Labeling

subj v obj prep

The robot *broke* my favorite mug with a wrench.

breaker — ARG0 | thing broken — ARG1 | instrument — ARG2

subj v prep adv

My mug *broke* into pieces immediately.

thing broken — ARG1 | pieces (final state) — ARG3 | temporal — ARGM-TMP

Frame: *break.01*

role	description
ARG0	breaker
ARG1	thing broken
ARG2	instrument
ARG3	pieces
ARG4	broken away from what?

The Proposition Bank (PropBank)

Core roles:
Verb-specific roles (ARG0-ARG5) defined in frame files

Frame: *break.01*

role	description
ARG0	breaker
ARG1	thing broken
ARG2	instrument

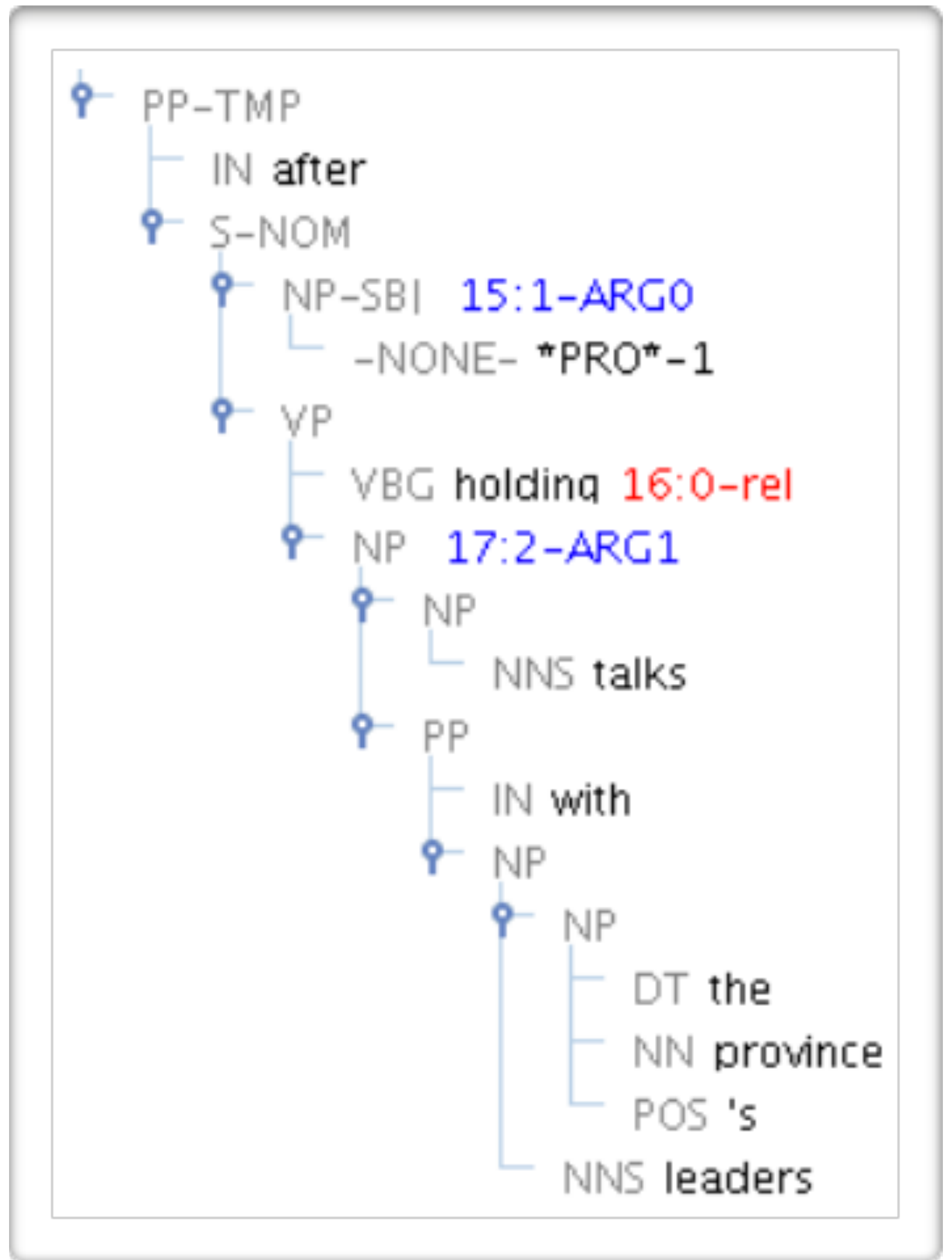
Frame: *buy.01*

role	description
ARG0	buyer
ARG1	thing bough
ARG2	seller
ARG3	price paid
ARG4	benefactive

Adjunct roles:
(ARGM-) shared across verbs

role	description
TMP	temporal
LOC	location
MNR	manner
DIR	direction
CAU	cause
PRP	purpose
…	

Annotated on top of the Penn Treebank Syntax



PropBank Annotation Guidelines, Bonial et al., 2010

Figure from He et al. (2017)

Syntax vs. Semantics

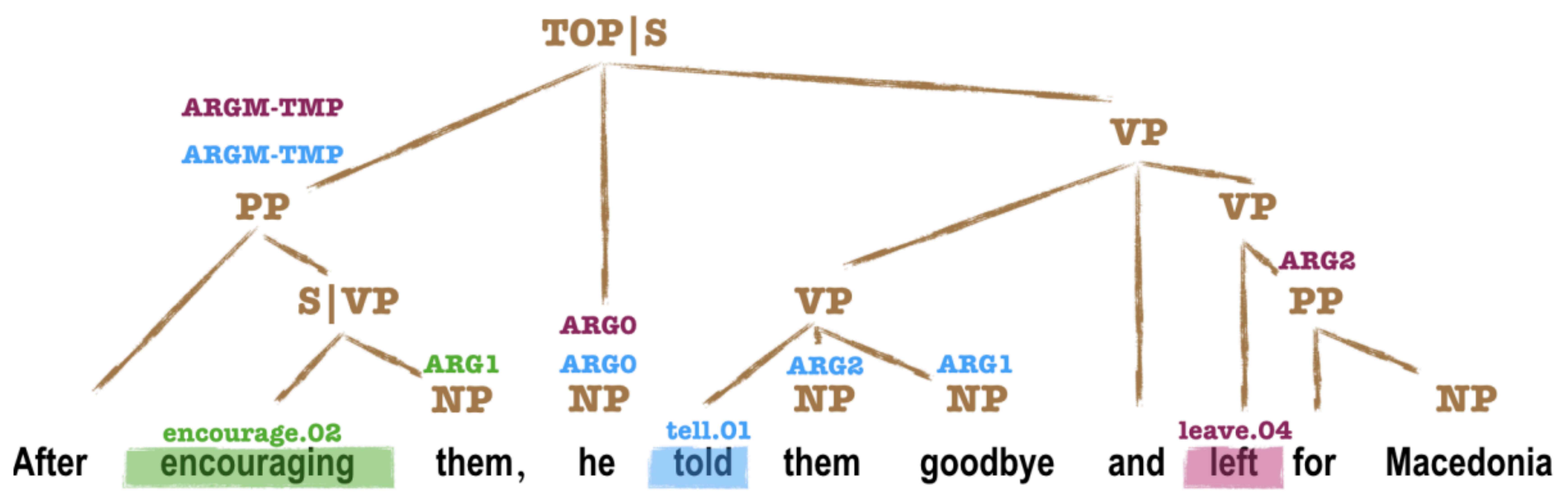


Figure 1.2: Syntax and semantics are closely related. The phrase-syntactic tree is shown in brown above the sentence. Semantic role labeling (SRL) structures from PropBank (Palmer et al., 2005) are shown alongside, in green, blue and magenta. Under SRL, words in the sentence that indicate stand-alone events are selected as predicates. These are shown as highlighted leaf nodes—“encouraging”, “told” and “left”. Each predicate is disambiguated to its relevant sense shown above it. Arguments to the predicates are are annotated on top of syntactic nodes, with the role labels color-coded by the predicate. SRL substructures (predicates, arguments) thus fully overlap with phrase-syntactic nodes.

Figure from Swayamdipta (2019)

Question-Answer Driven SRL

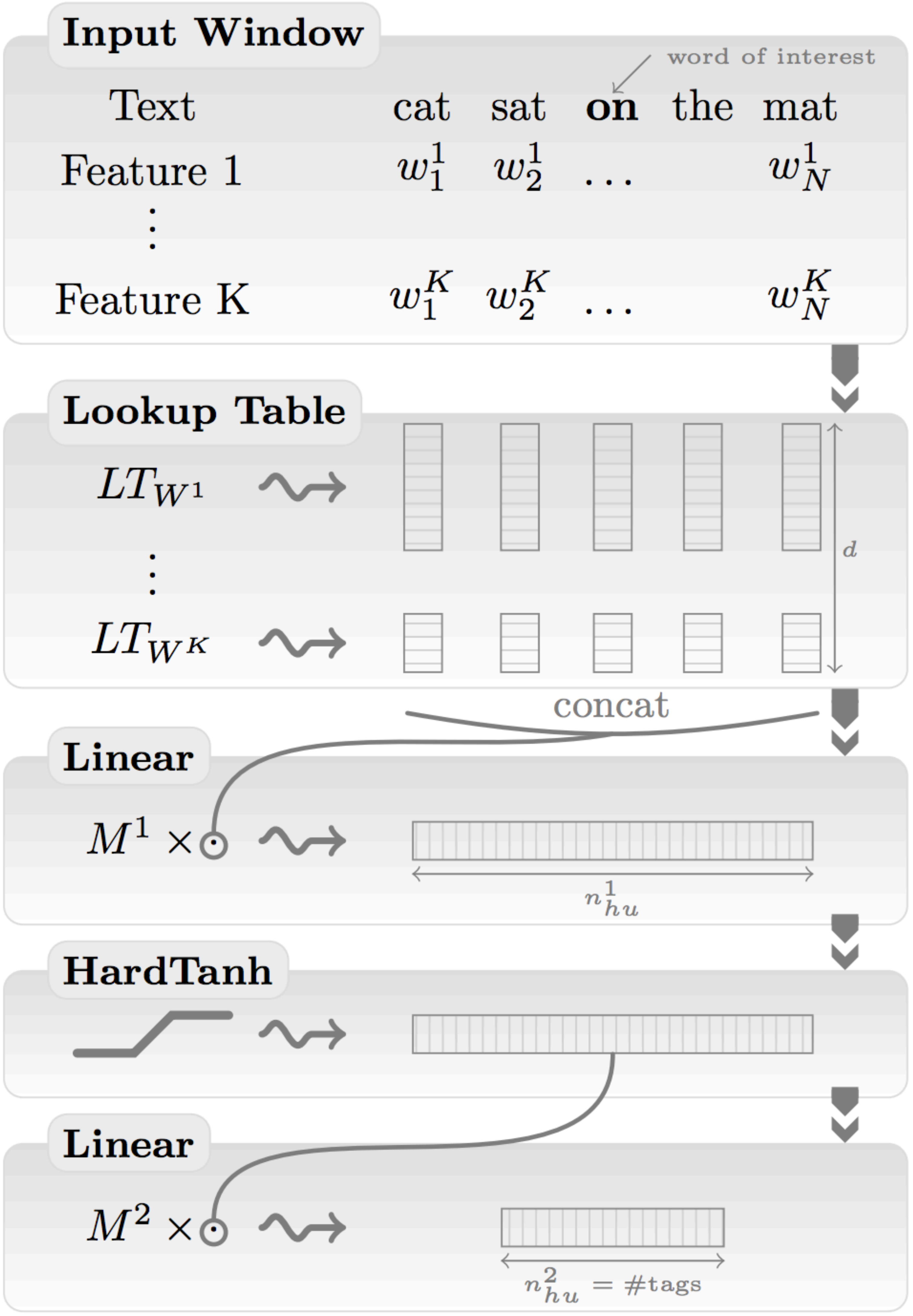
In 1950 Alan M. Turing ***published*** "Computing machinery and intelligence" in Mind, in which he ***proposed*** that machines could be ***tested*** for intelligence ***using*** questions and answers.

Predicate		Question	Answer
published	1	Who published something?	Alan M. Turing
	2	What was published?	"Computing Machinery and Intelligence"
	3	When was something published?	In 1950
proposed	4	Who proposed something?	Alan M. Turing
	5	What did someone propose?	that machines could be tested for intelligent using questions and answers
	6	When did someone propose something?	In 1950
tested	7	What can be tested?	machines
	8	What can something be tested for?	intelligence
	9	How can something be tested?	using questions and answers
using	10	What was being used?	questions and answers
	11	Why was something being used?	tested for intelligence

Figure from FitzGerald et al. (2018)

"NLP (Almost) From Scratch"

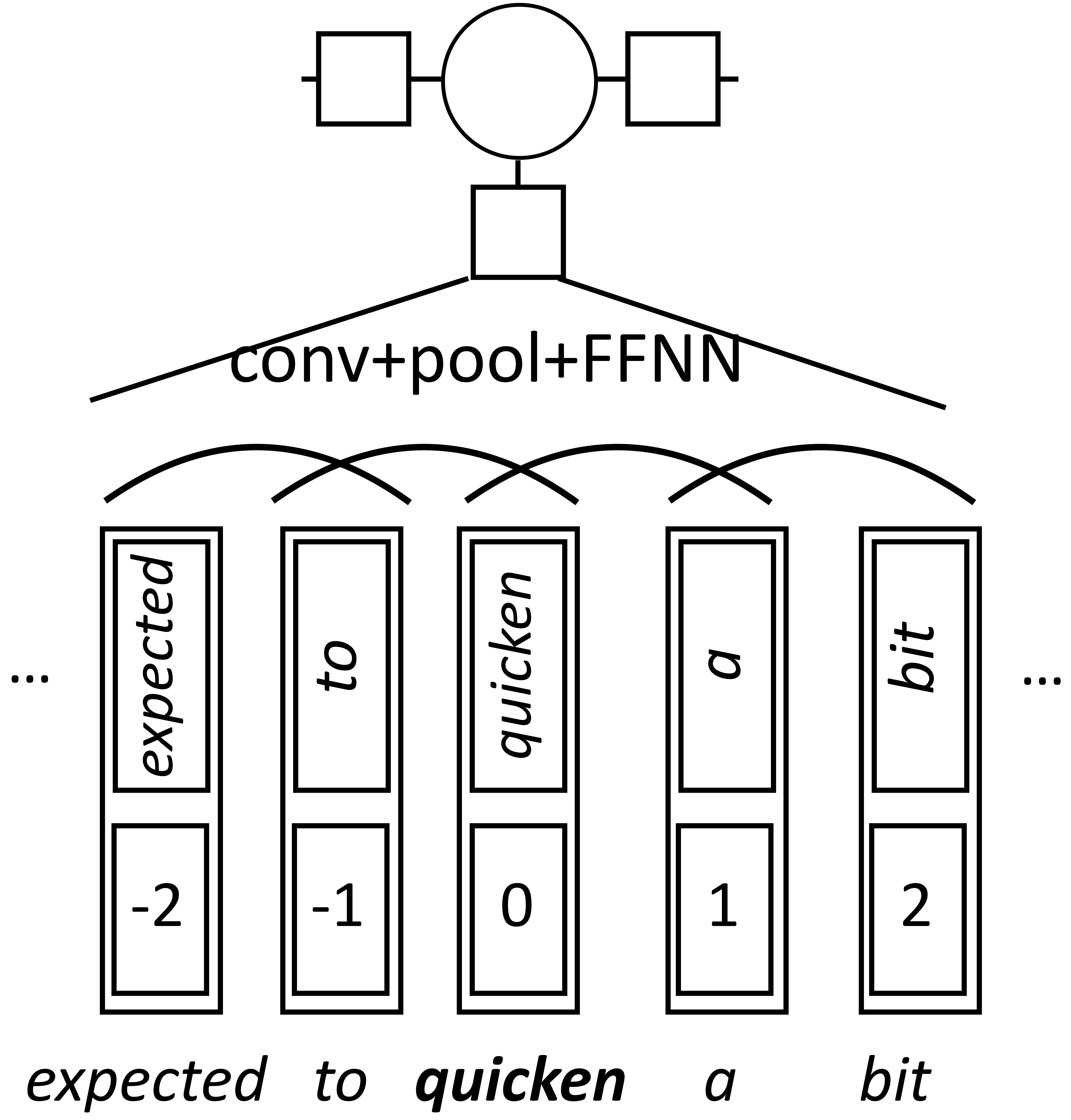
Approach	POS (PWA)	CHUNK (F1)	NER (F1)	SRL (F1)
Benchmark Systems	97.24	94.29	89.31	77.92
NN+WLL	96.31	89.13	79.53	55.40
NN+SLL	96.37	90.33	81.47	70.99
NN+WLL+LM1	97.05	91.91	85.68	58.18
NN+SLL+LM1	97.10	93.65	87.58	73.84
NN+WLL+LM2	97.14	92.04	86.96	58.34
NN+SLL+LM2	97.20	93.63	88.67	74.15



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

Collobert, Weston, et al. 2008, 2011

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
- Use this for SRL: the verb (predicate) is at position 0, CNN looks at the whole sentence “relative” to the verb

CNN NCRFs vs. FFNN NCRFs

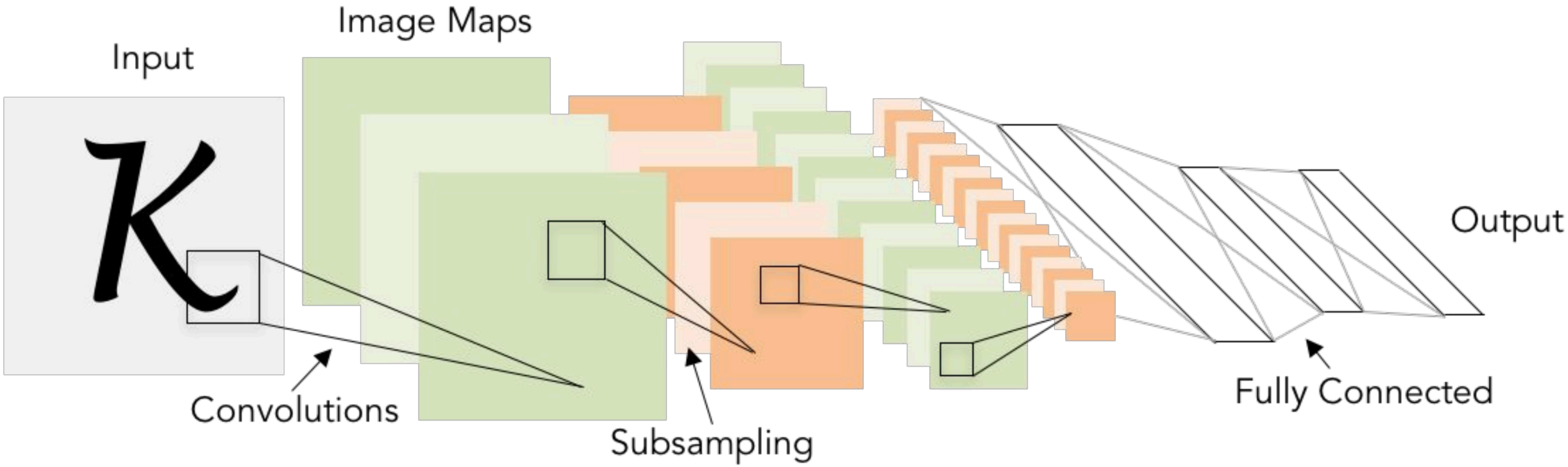
Approach	**POS** (PWA)	**CHUNK** (F1)	**NER** (F1)	**SRL** (F1)
Benchmark Systems	97.24	94.29	89.31	77.92
	Window Approach			
NN+SLL+LM2	97.20	93.63	88.67	–
	Sentence Approach			
NN+SLL+LM2	97.12	93.37	88.78	74.15

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

Collobert and Weston 2008, 2011

CNN

A Bit of History



https://www.youtube.com/watch?v=FwFduRA_L6Q

LeCun et al. (1998), earlier work in 1980s

ImageNet - Object Recognition

Steel drum

The Image Classification Challenge:

1,000 object classes

1,431,167 images

Output:
Scale
T-shirt
Steel drum
Drumstick
Mud turtle

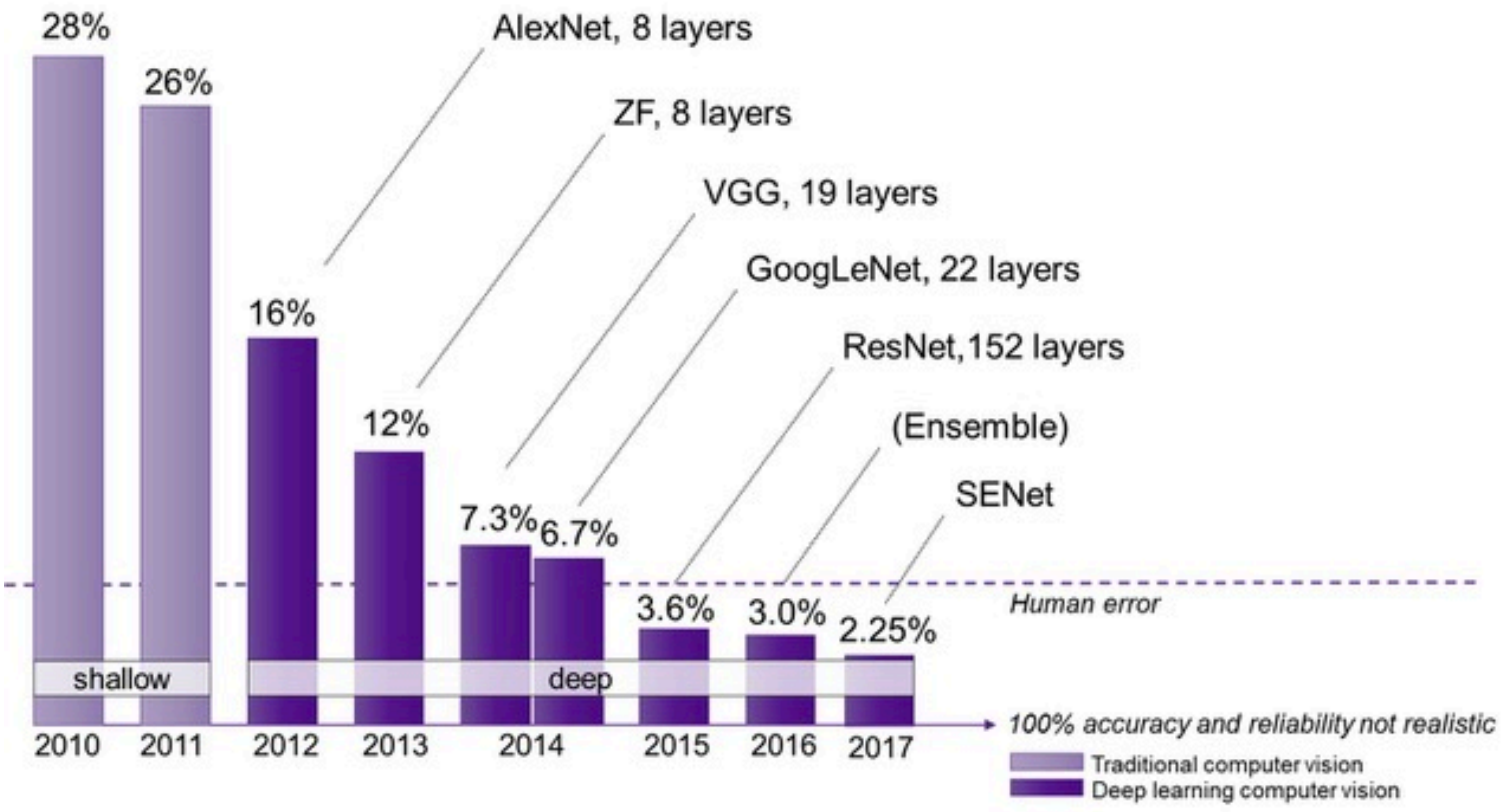
✔

Output:
Scale
T-shirt
Giant panda
Drumstick
Mud turtle

✗

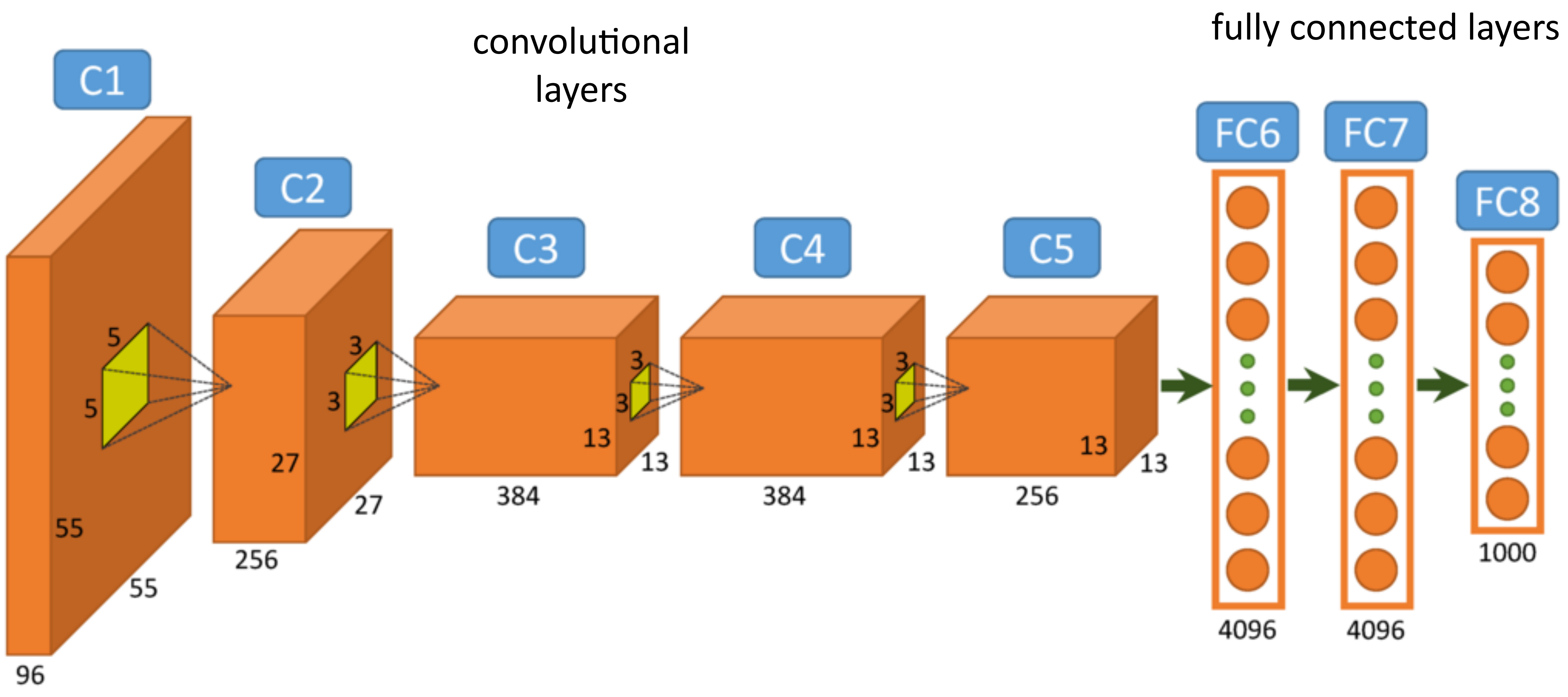
Russakovsky et al. (2012)

ImageNet - Object Recognition



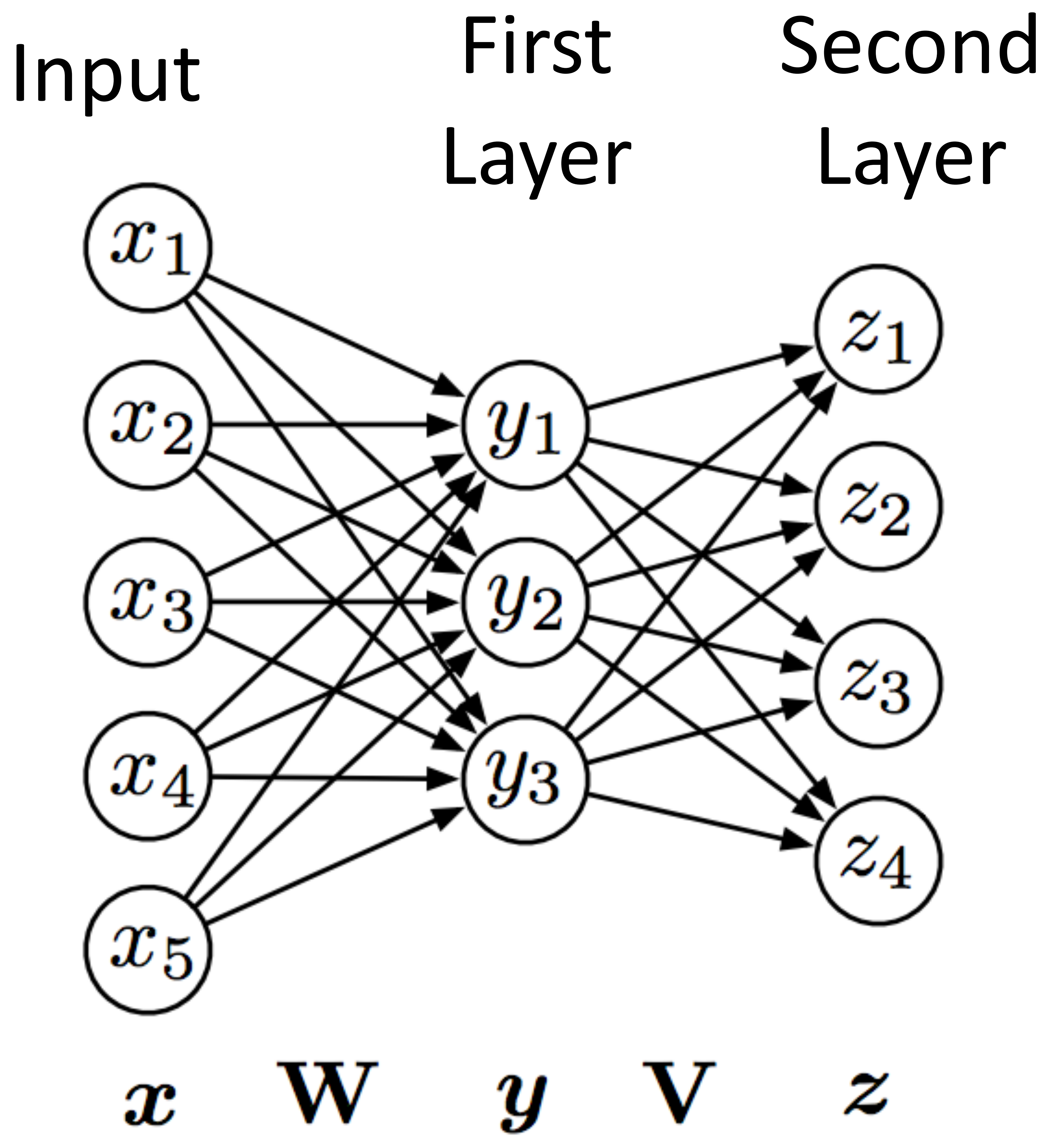
Convolutional Neural Networks

- AlexNet - one of the first strong results
- more filters per layer as well as stacked convolutional layers
- use of ReLU for the non-linear part instead of Sigmoid or Tanh



Krizhevsky et al. (2012)

Feedforward Neural Networks (Recap)



$$\boldsymbol{y} = g(\mathbf{W}\boldsymbol{x} + \boldsymbol{b})$$

$$\mathbf{z} = g(\mathbf{V}\mathbf{y} + \mathbf{c})$$

$$\mathbf{z} = g(\mathbf{V}\underbrace{g(\mathbf{W}\mathbf{x} + \mathbf{b})}_{\text{output of first layer}} + \mathbf{c})$$

"Feedforward" computation (not recurrent)

Check: what happens if no nonlinearity?
More powerful than basic linear models?

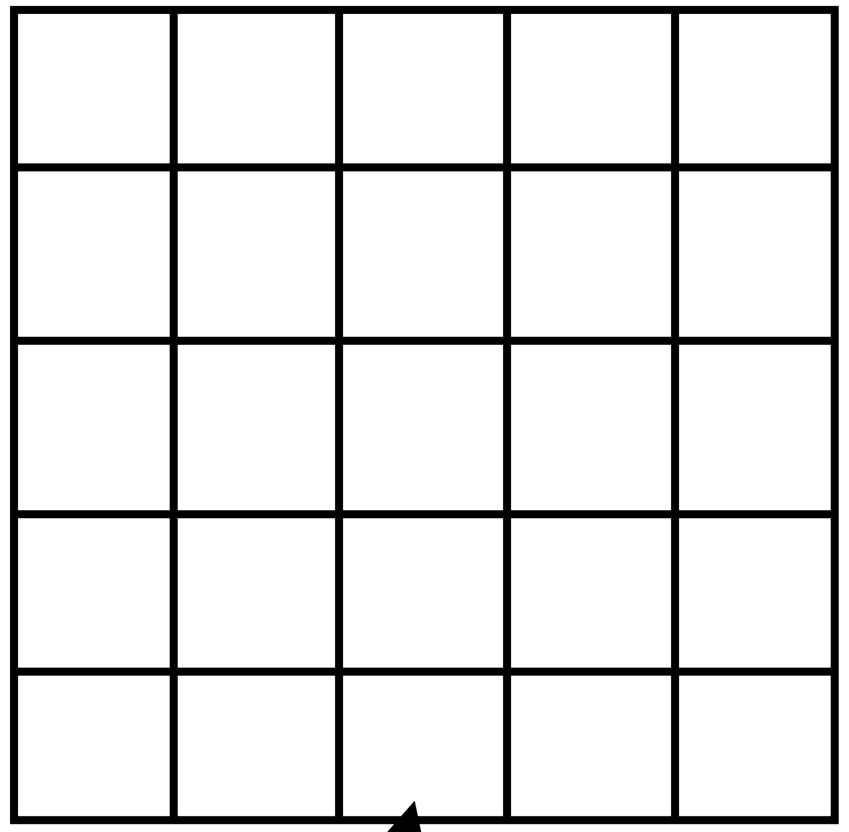
$$\mathbf{z} = \mathbf{V}(\mathbf{W}\mathbf{x} + \mathbf{b}) + \mathbf{c}$$

Adopted from Chris Dyer

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 x n x k



filter: m x m x k

sum over dot products

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

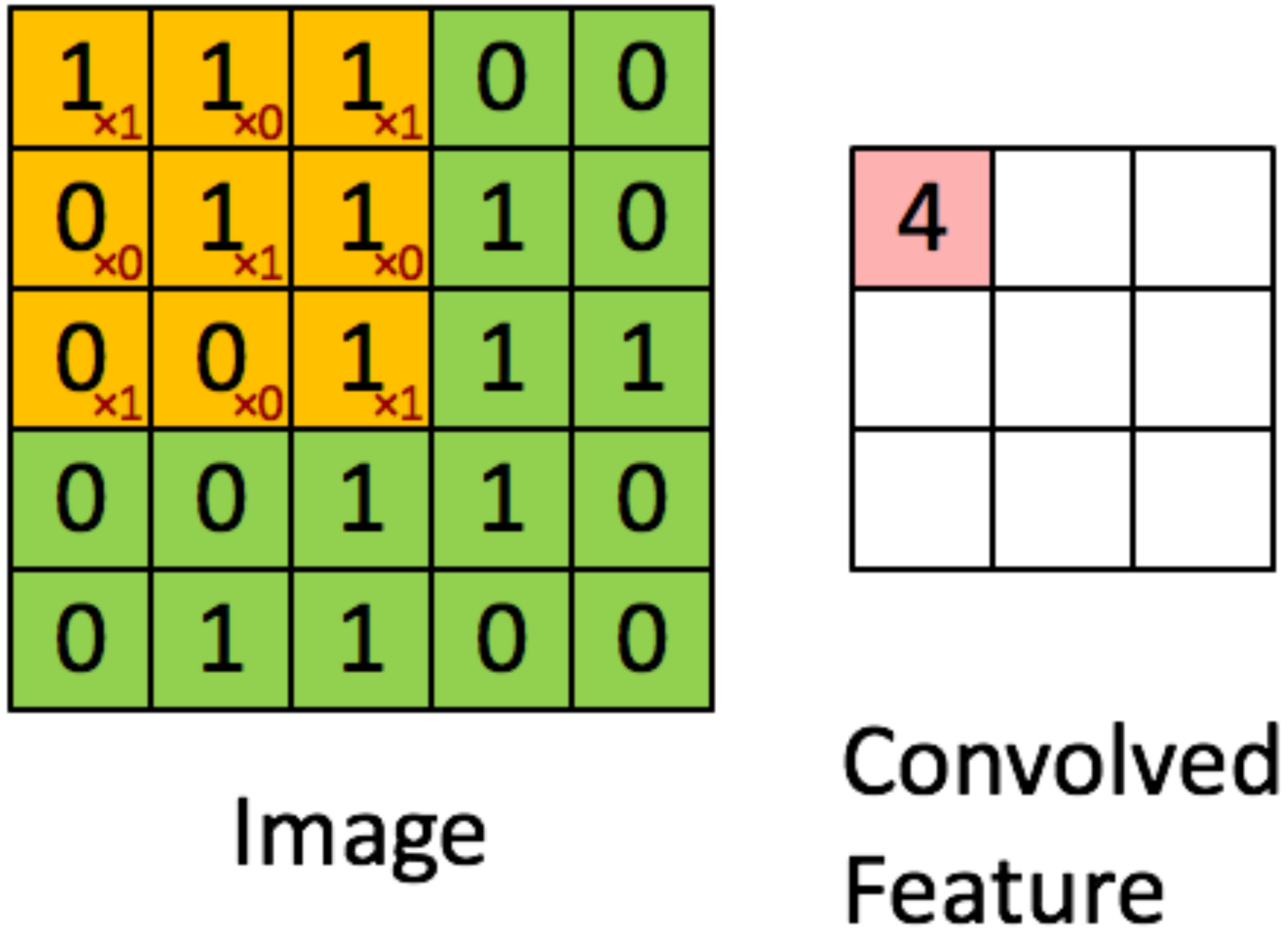
offsets

Each of these cells is a vector with multiple values

Images: RGB values (3 dim)

Convolutional Layer

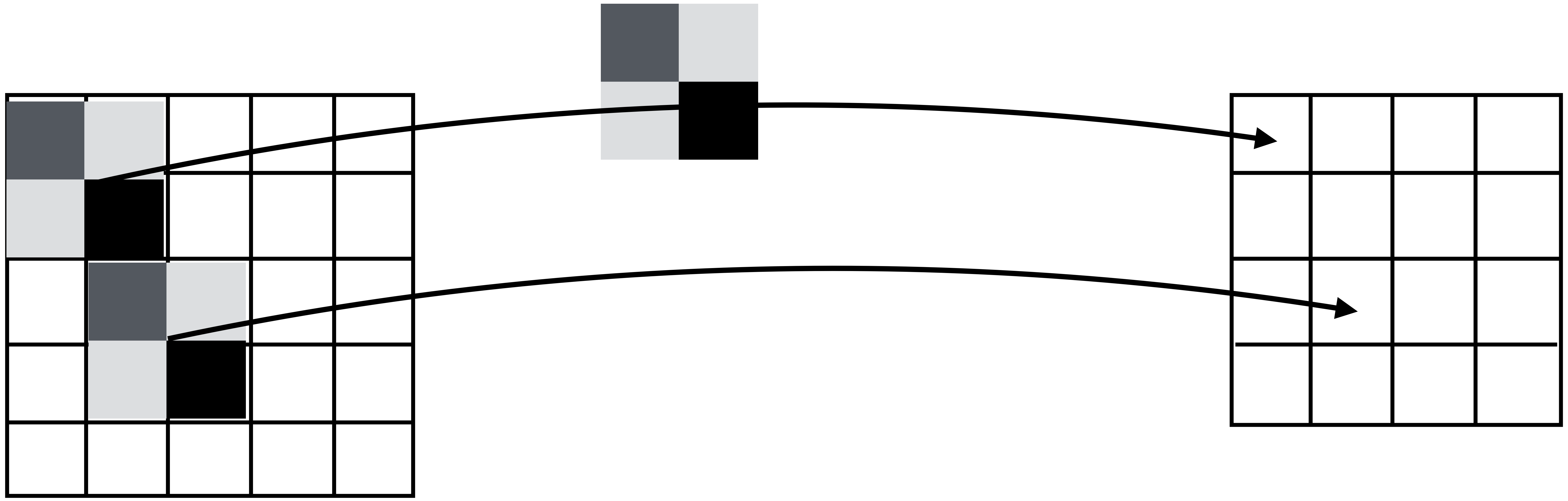
- An animated example: k = 1, and a filter of size 3x3.



Convolutional Layer

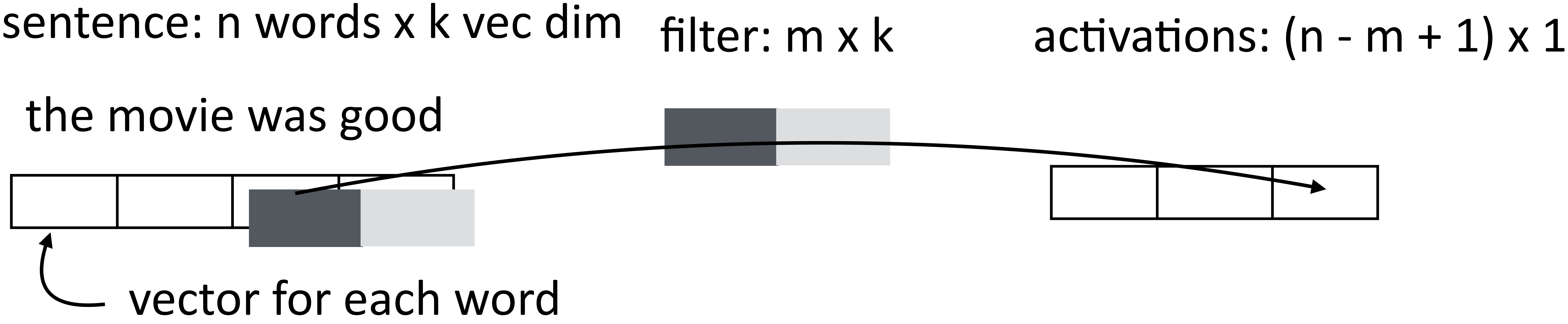
- Applies a *filter* over patches of the input and returns that filter's activations
- Convolution: take dot product of filter with a patch of the input

image: n x n x k   filter: m x m x k   activations: (n - m + 1) x (n - m + 1) x 1



Convolutions for NLP

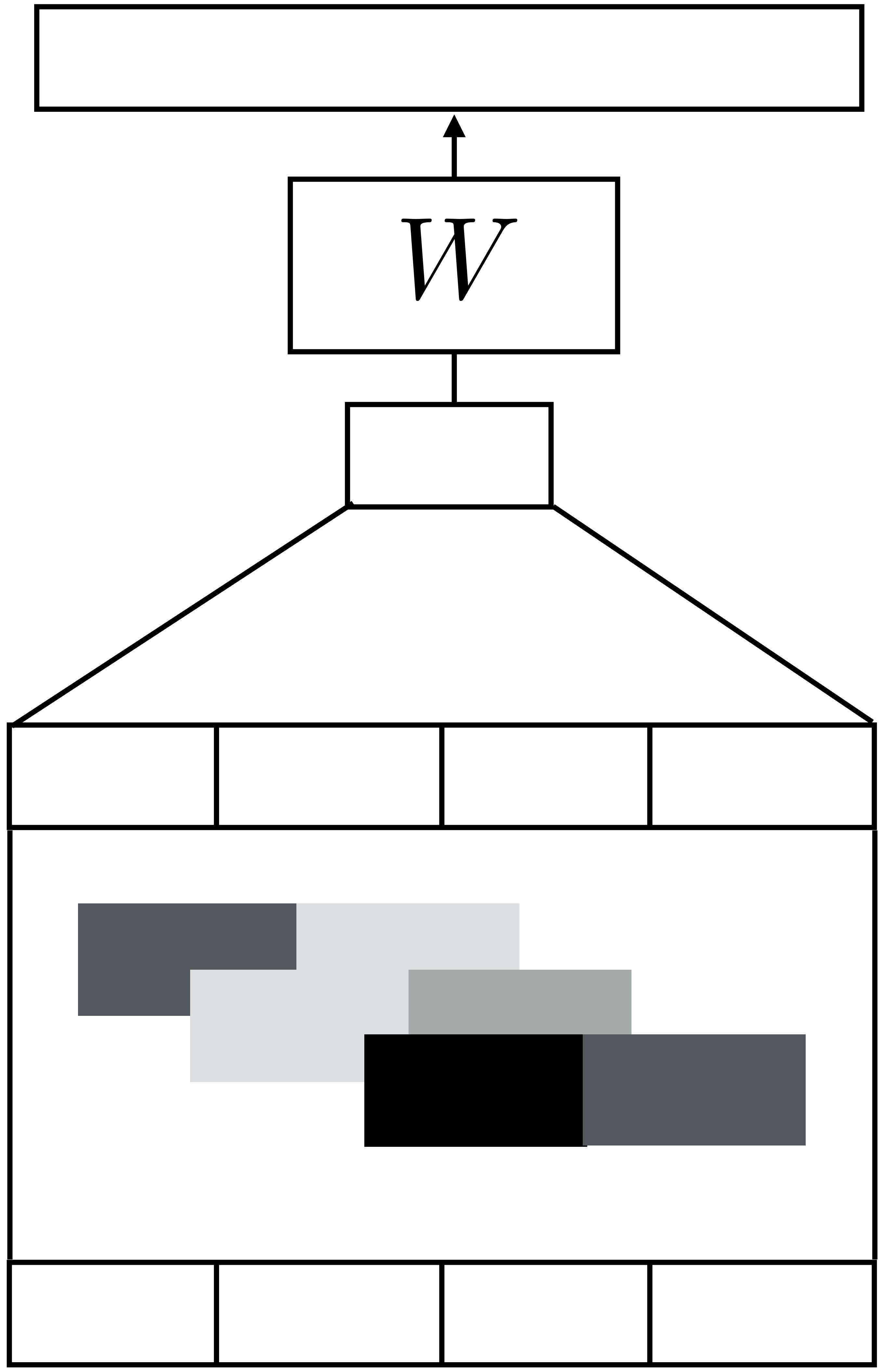
- Input and filter are 2-dimensional instead of 3-dimensional



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

CNNs for Sentiment

CNNs for Sentiment Analysis



the movie was good

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

projection + softmax

c-dimensional vector

max pooling over the sentence

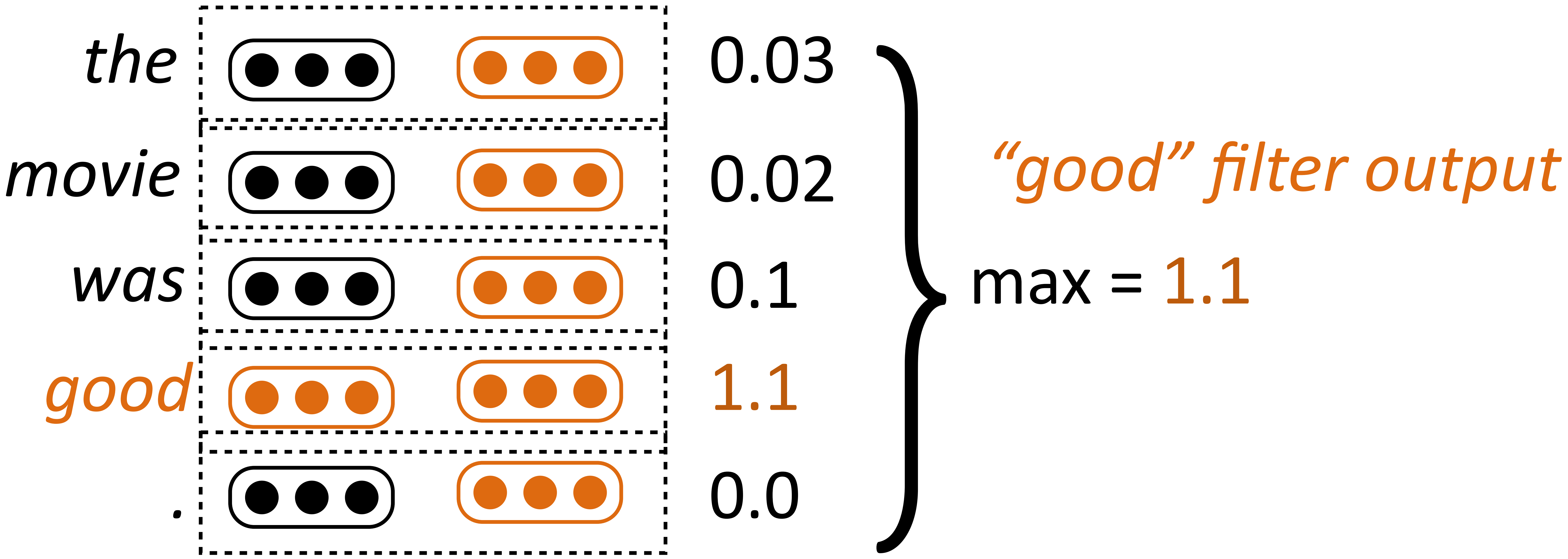
n x c

c filters,
m x k each

n x k

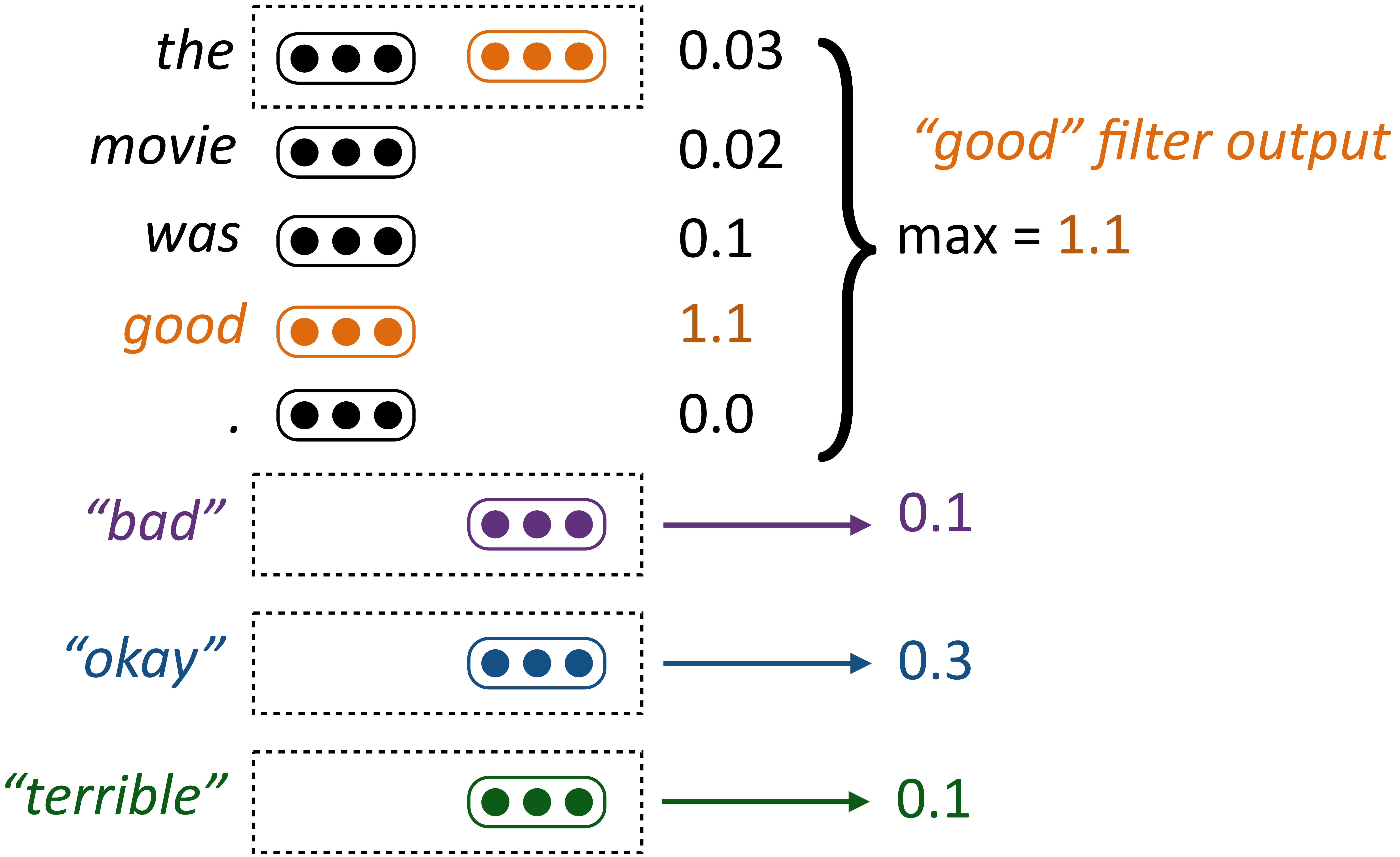
- Max pooling: return the max activation of a given filter over the entire sentence; like a logical OR (sum pooling is like logical AND)

Understanding CNNs for Sentiment

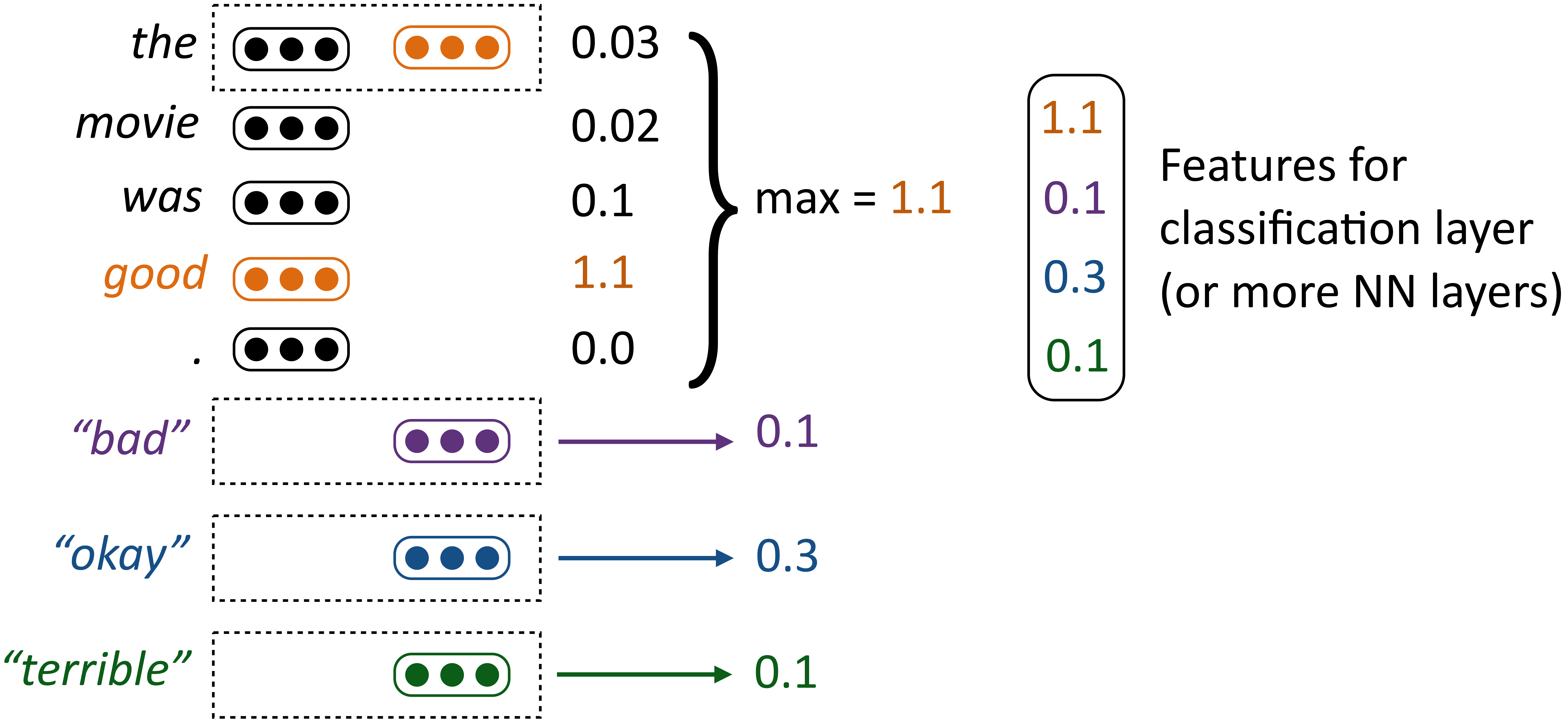


- Filter "looks like" the things that will cause it to have high activation

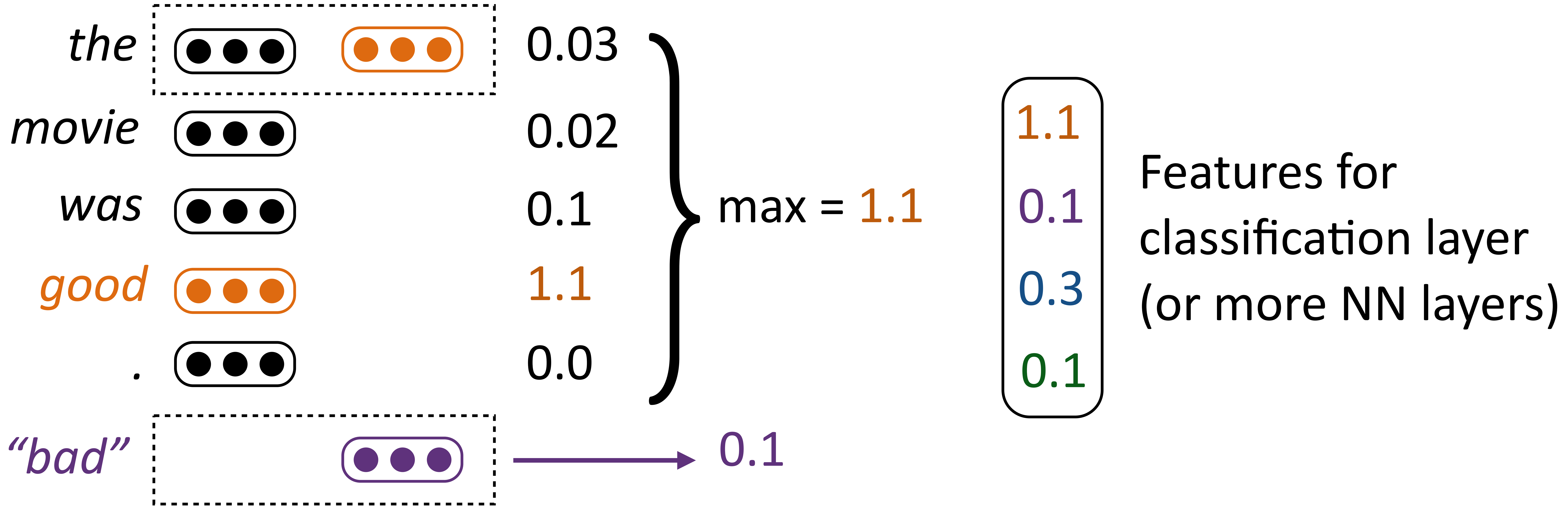
Understanding CNNs for Sentiment



Understanding CNNs for Sentiment

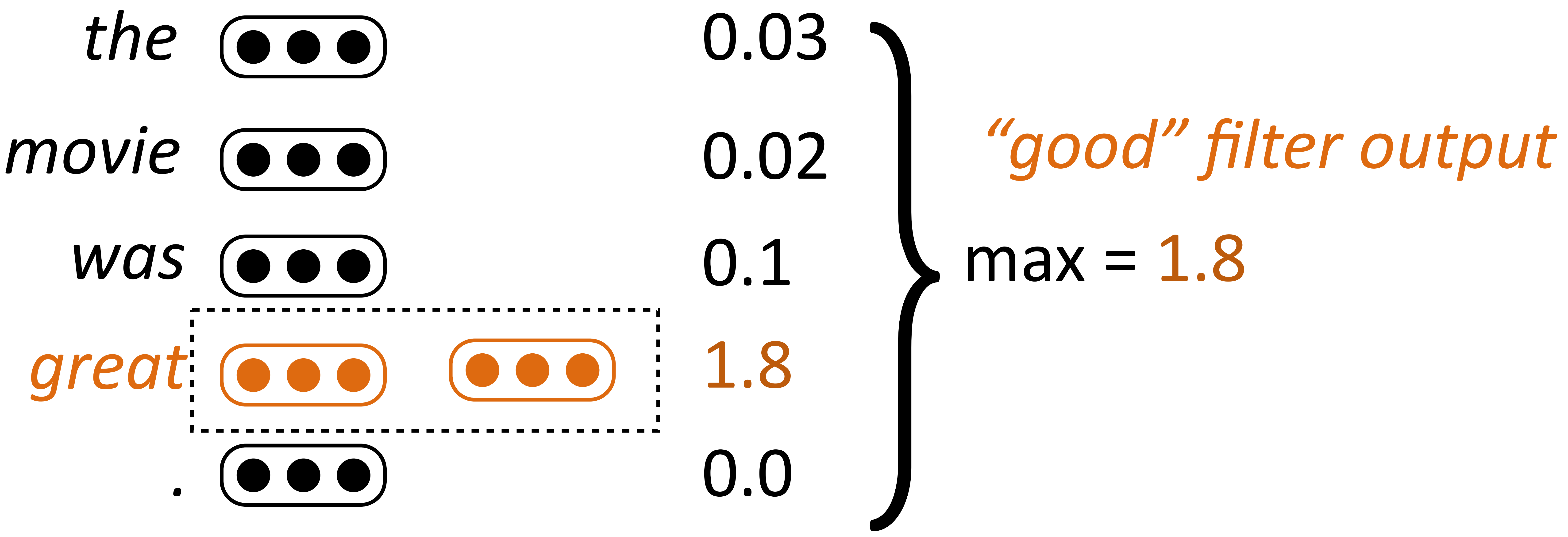


Understanding CNNs for Sentiment



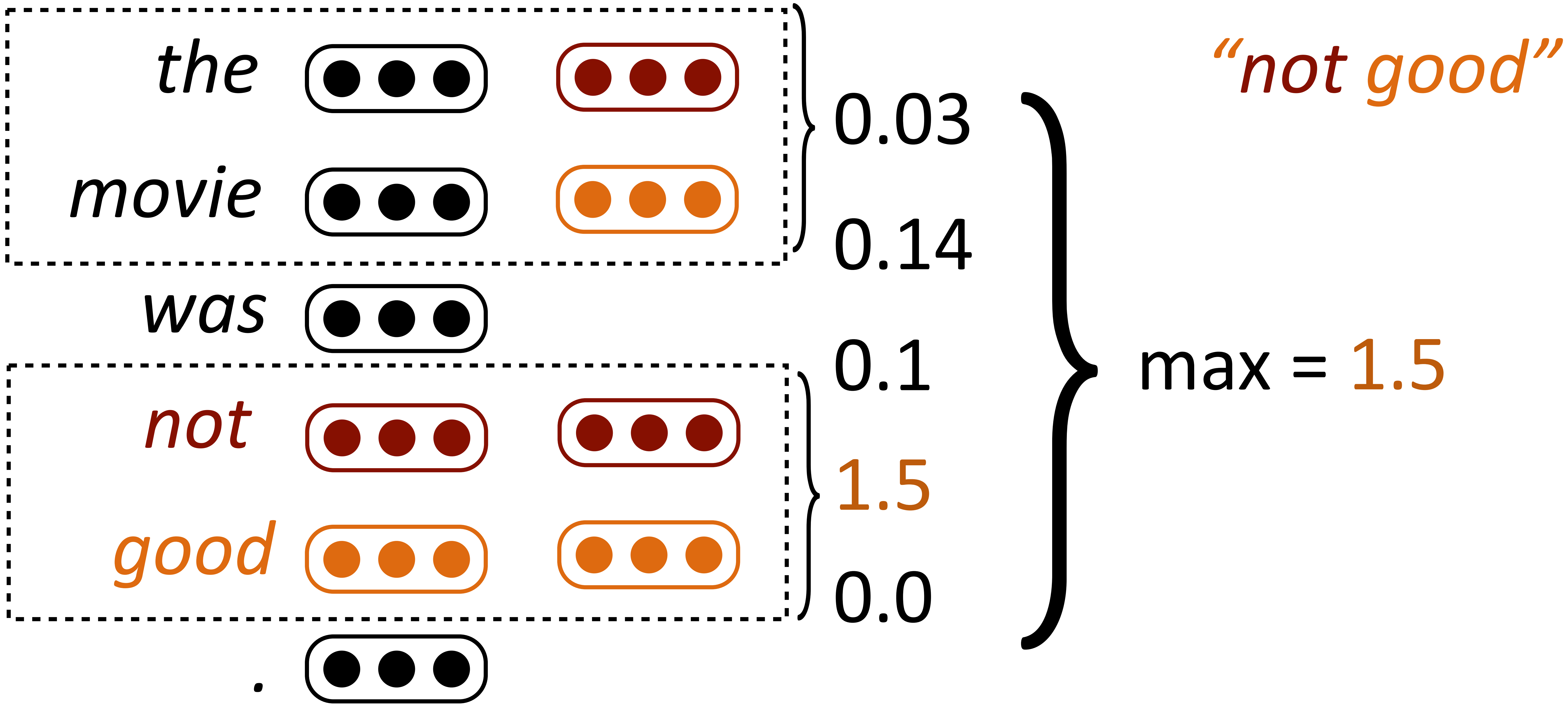
- Takes variable-length input and turns it into fixed-length output
- Filters are initialized randomly and then learned

Understanding CNNs for Sentiment



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

Understanding CNNs for Sentiment



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

What can CNNs learn?

- CNNs let us take advantage of word similarity

really not very good vs. *really not very enjoyable*

- CNNs are translation-invariant like bag-of-words

The movie was bad, but blah blah blah … vs. *… blah blah blah, but the movie was bad.*

- CNNs can capture local interactions with filters of width > 1

It was not good, it was actually quite bad vs. *it was not bad, it was actually quite good*

CNNs: Implementation

- Input is batch_size x n x k matrix, filters are c x m x 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

CLASS `torch.nn.Conv1d(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True, padding_mode='zeros')` [SOURCE]

Applies a 1D convolution over an input signal composed of several input planes.

In the simplest case, the output value of the layer with input size (N, C_{in}, L) and output $(N, C_{\text{out}}, L_{\text{out}})$ can be precisely described as:

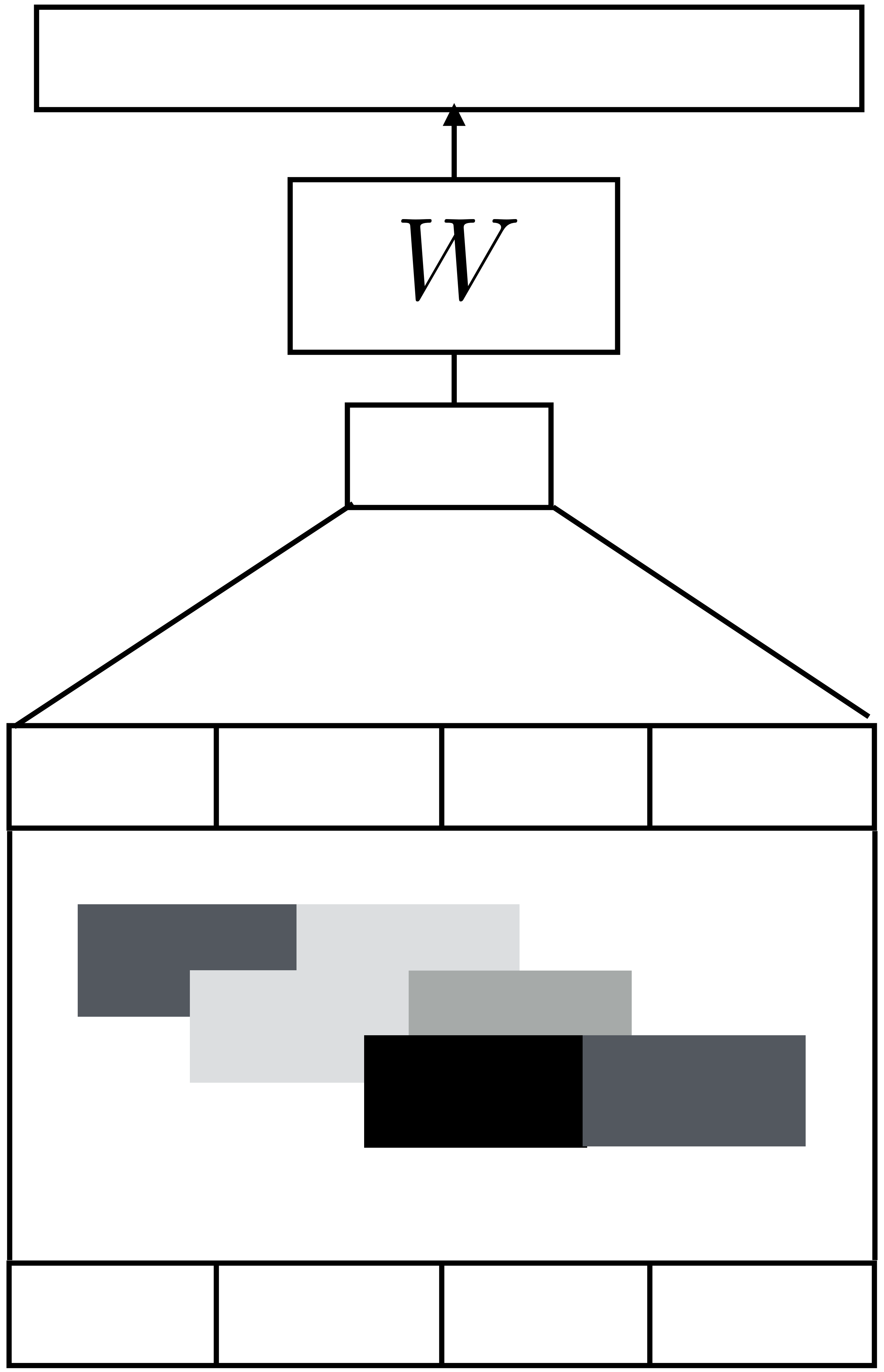
$$\text{out}(N_i, C_{\text{out}_j}) = \text{bias}(C_{\text{out}_j}) + \sum_{k=0}^{C_{in}-1} \text{weight}(C_{\text{out}_j}, k) \star \text{input}(N_i, k)$$

where $\star$ is the valid cross-correlation operator, N is a batch size, C denotes a number of channels, L is a length of signal sequence.

- `stride` controls the stride for the cross-correlation, a single number or a one-element tuple.
- `padding` controls the amount of implicit zero-paddings on both sides for `padding` number of points.

CNNs for Sentence Classification

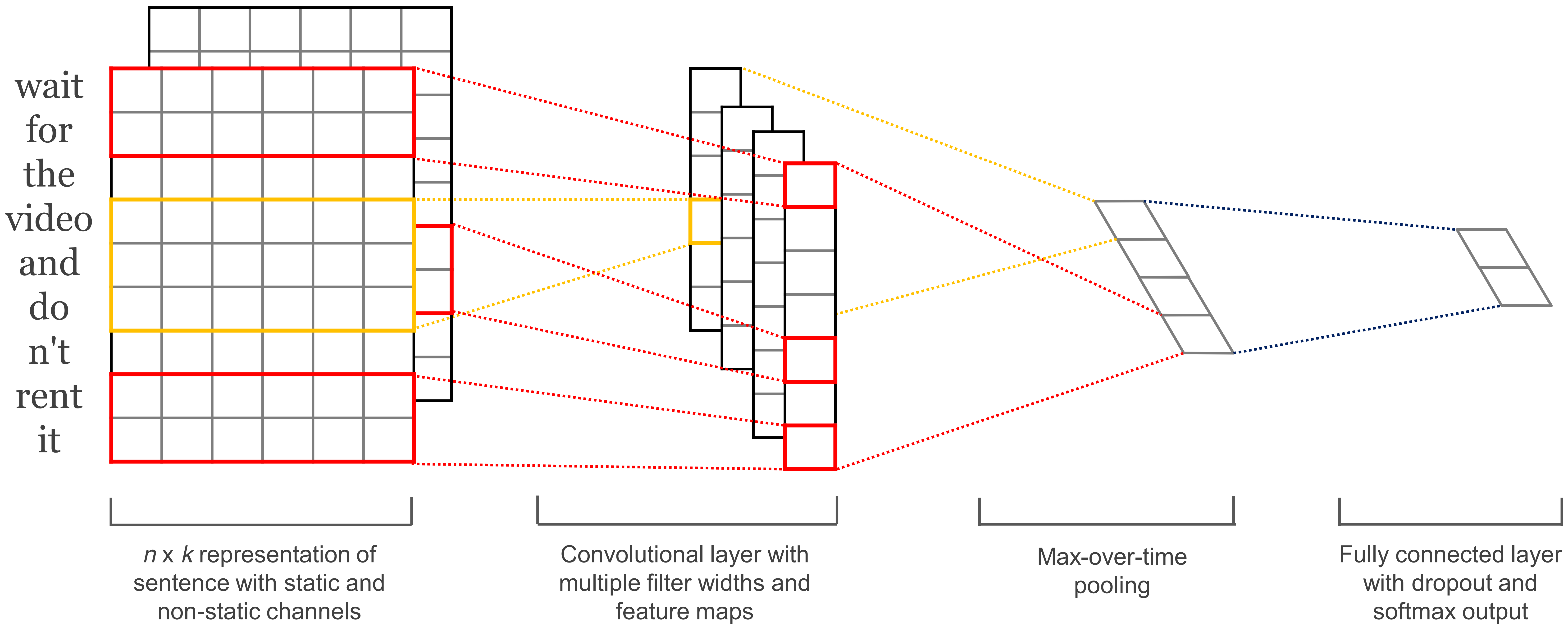
- Question classification, sentiment, etc.
- Conv+pool, then use feedforward layers to classify
- Can use multiple types of input vectors (fixed initializer and learned)



the movie was good

Kim (2014)

CNNs for Sentence Classification



Kim (2014)

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

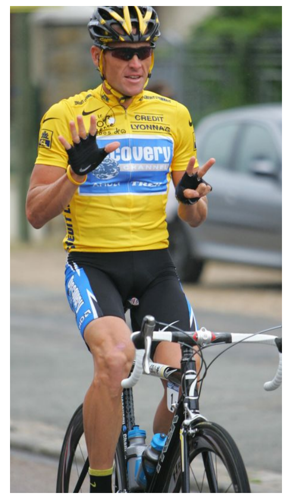
Kim (2014)

Entity Linking

- CNNs can produce good representations of both sentences and documents like typical bag-of-words features
- Can distill topic representations for use in entity linking

that they had disqualified **Armstrong** from his seven consecutive

cycling domain



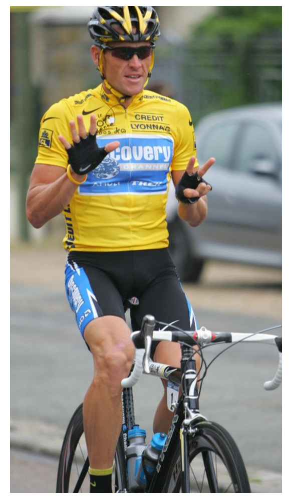
Lance Armstrong

geopolitical domain

Armstrong County

Entity Linking

Although he originally won the event, the United States Anti-Doping Agency announced in August 2012 that they had disqualified **Armstrong** from his seven consecutive Tour de France wins from 1999–2005.



Lance Edward Armstrong is an American former professional road cyclist

Armstrong County is a county in Pennsylvania…

CNN

CNN

CNN

Document topic vector d

Article topic vector a_{Lance}

Article topic vector a_{County}

$$s_{\text{Lance}} = d \cdot a_{\text{Lance}} \qquad s_{\text{County}} = d \cdot a_{\text{County}}$$

$$P(y|\mathbf{x}) = \text{softmax}(\mathbf{s})$$

Francis-Landau et al. (2016)

Entity Linking

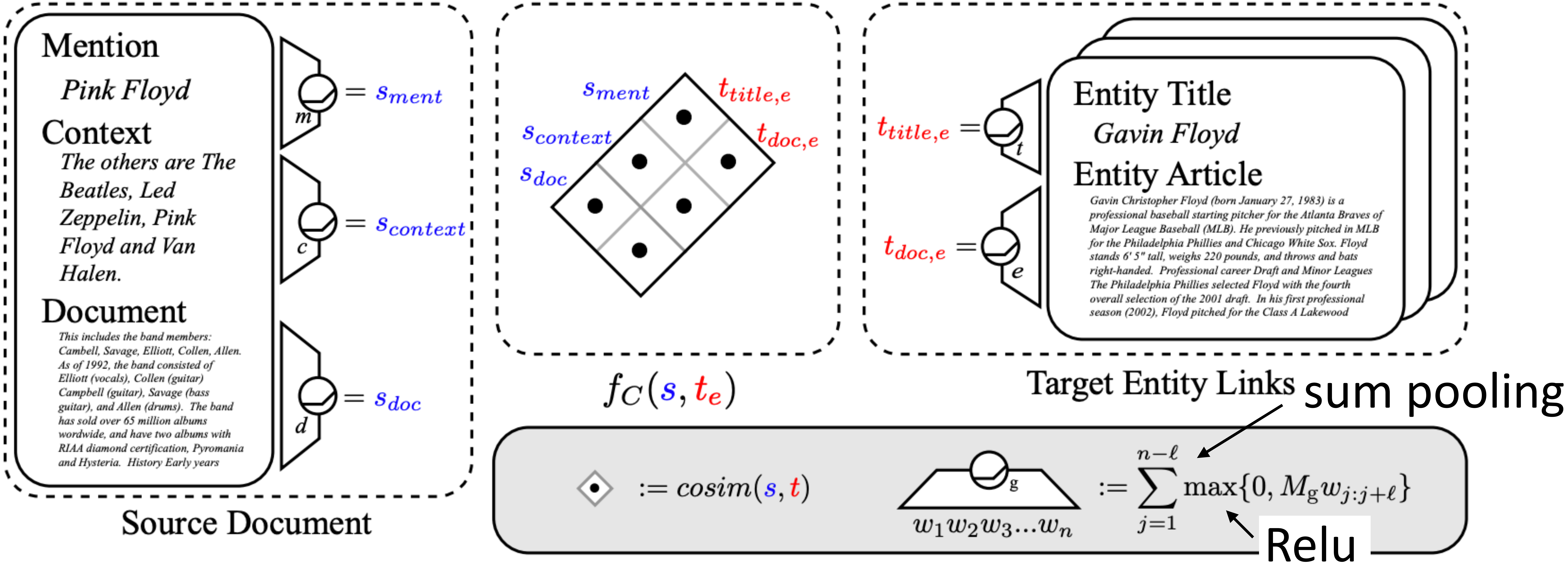
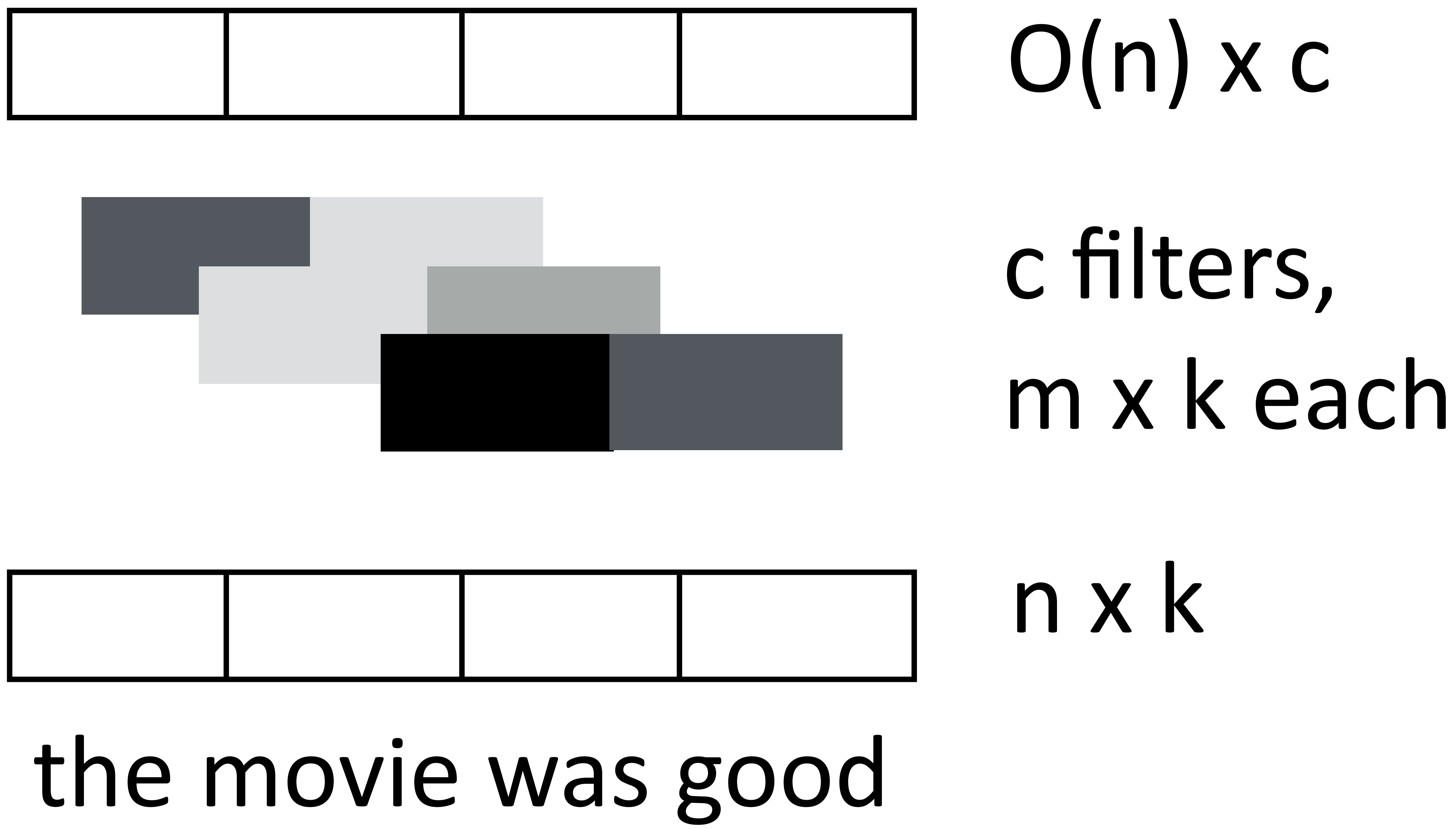
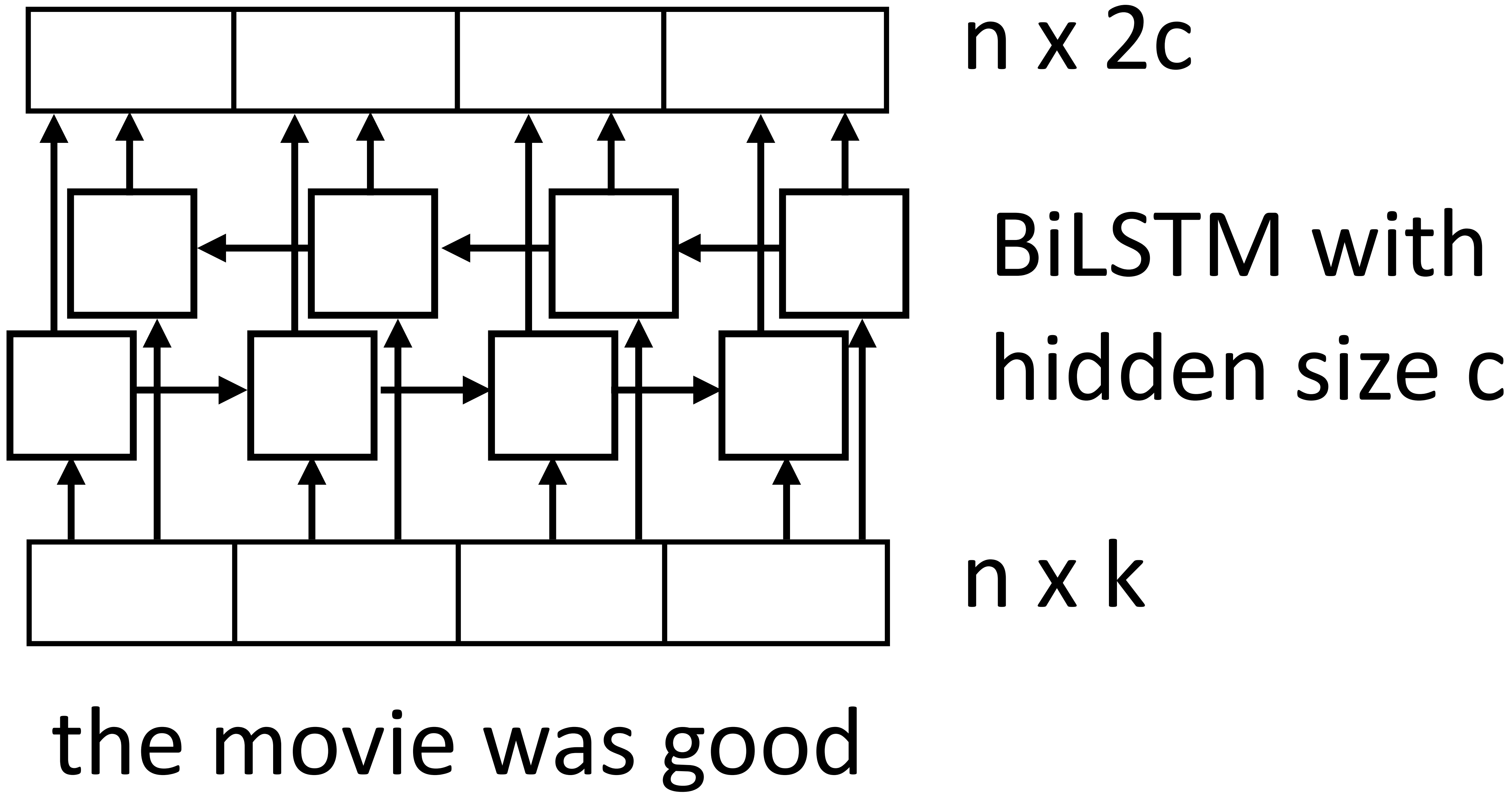


Figure 1: Extraction of convolutional vector space features $f_C(x, t_e)$. Three types of information from the input document and two types of information from the proposed title are fed through convolutional networks to produce vectors, which are systematically compared with cosine similarity to derive real-valued semantic similarity features.

Francis-Landau et al. (2016)

Compare: CNNs vs. LSTMs

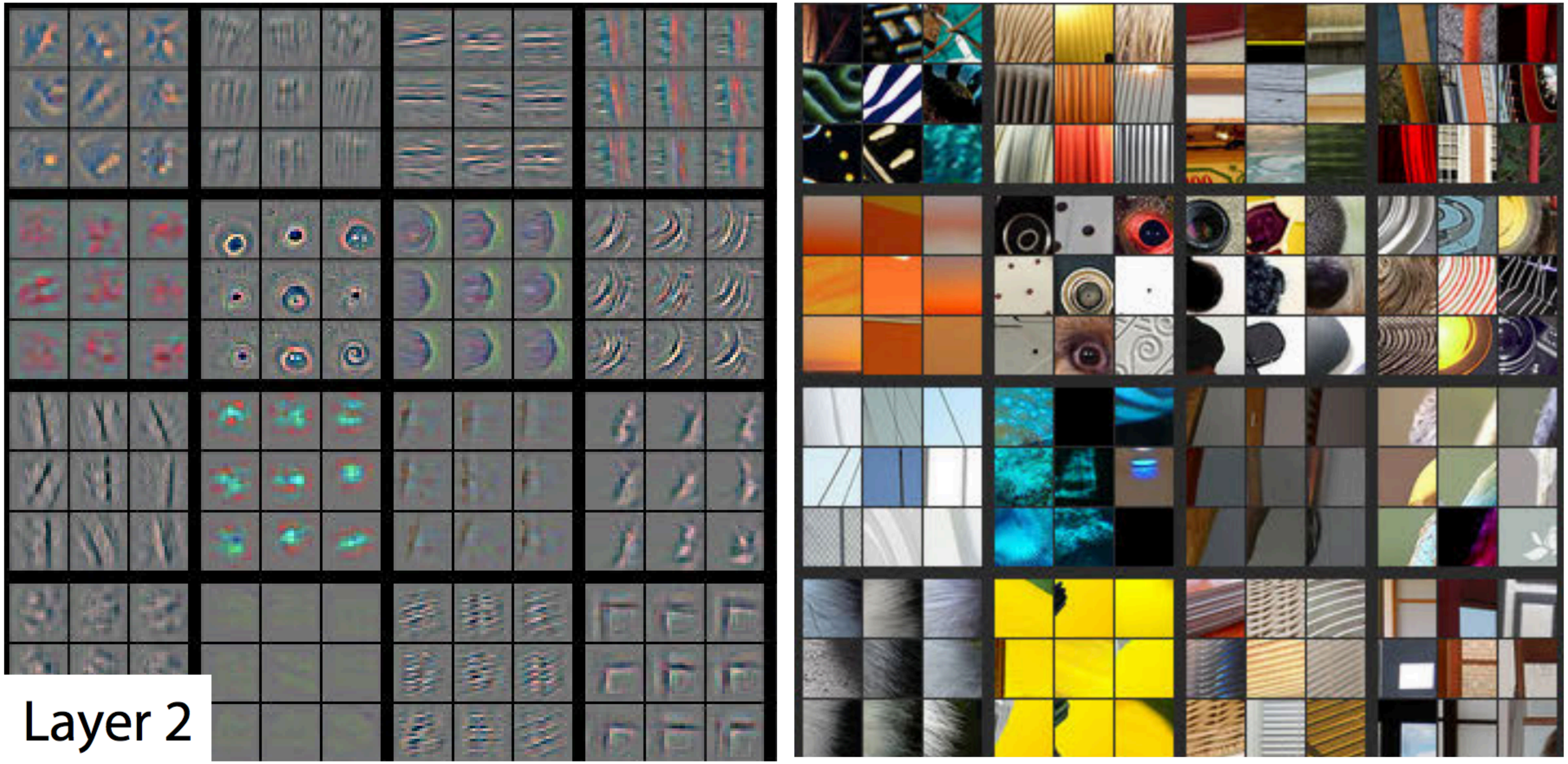




- Both LSTMs and convolutional layers transform the input using context
- LSTM: “globally” looks at the entire sentence (but local for many problems)
- CNN: local depending on filter width + number of layers

Deep Convolutional Networks

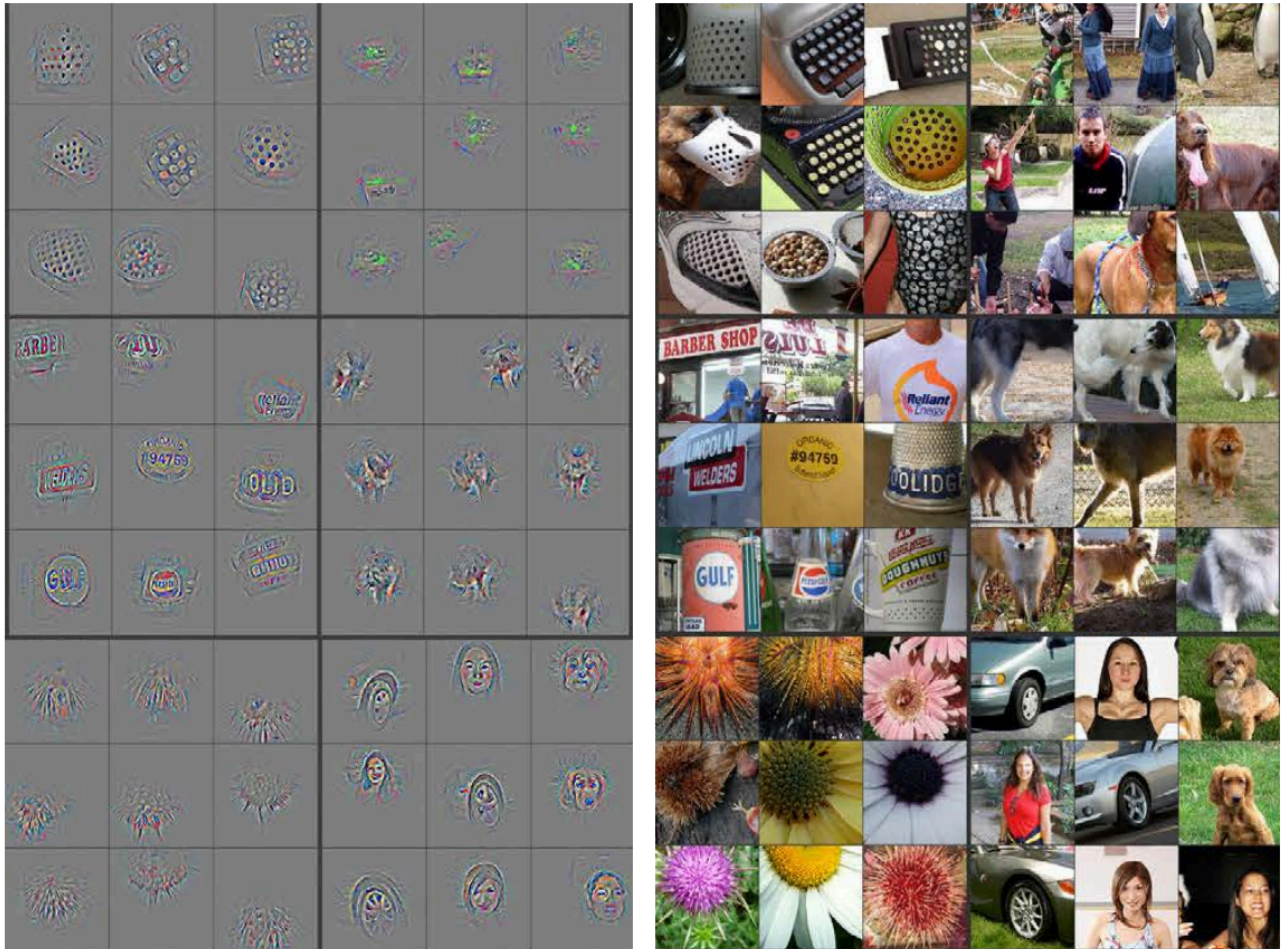
- Low-level filters: extract low-level features from the data



Zeiler and Fergus (2014)

Deep Convolutional Networks

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



Zeiler and Fergus (2014)

Takeaways

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

Sentence Alignment

- Neural CRF for Sentence Alignment

Professional editors rewrite news articles into 4 different readability levels for grade 3-12 students.

Chao Jiang, Mounica Maddela, Wuwei Lan, Yang Zhong, Wei Xu. “Neural CRF Model for Sentence Alignment in Text Simplification” (ACL 2020)

Sentence Alignment

- Neural CRF for Sentence Alignment

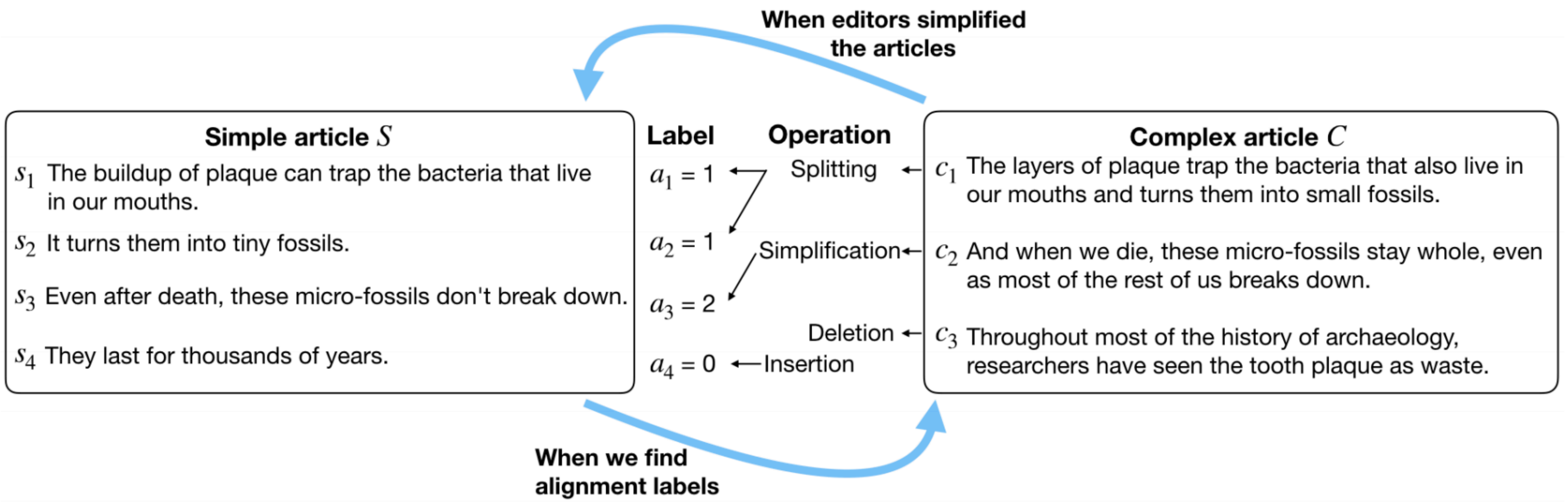
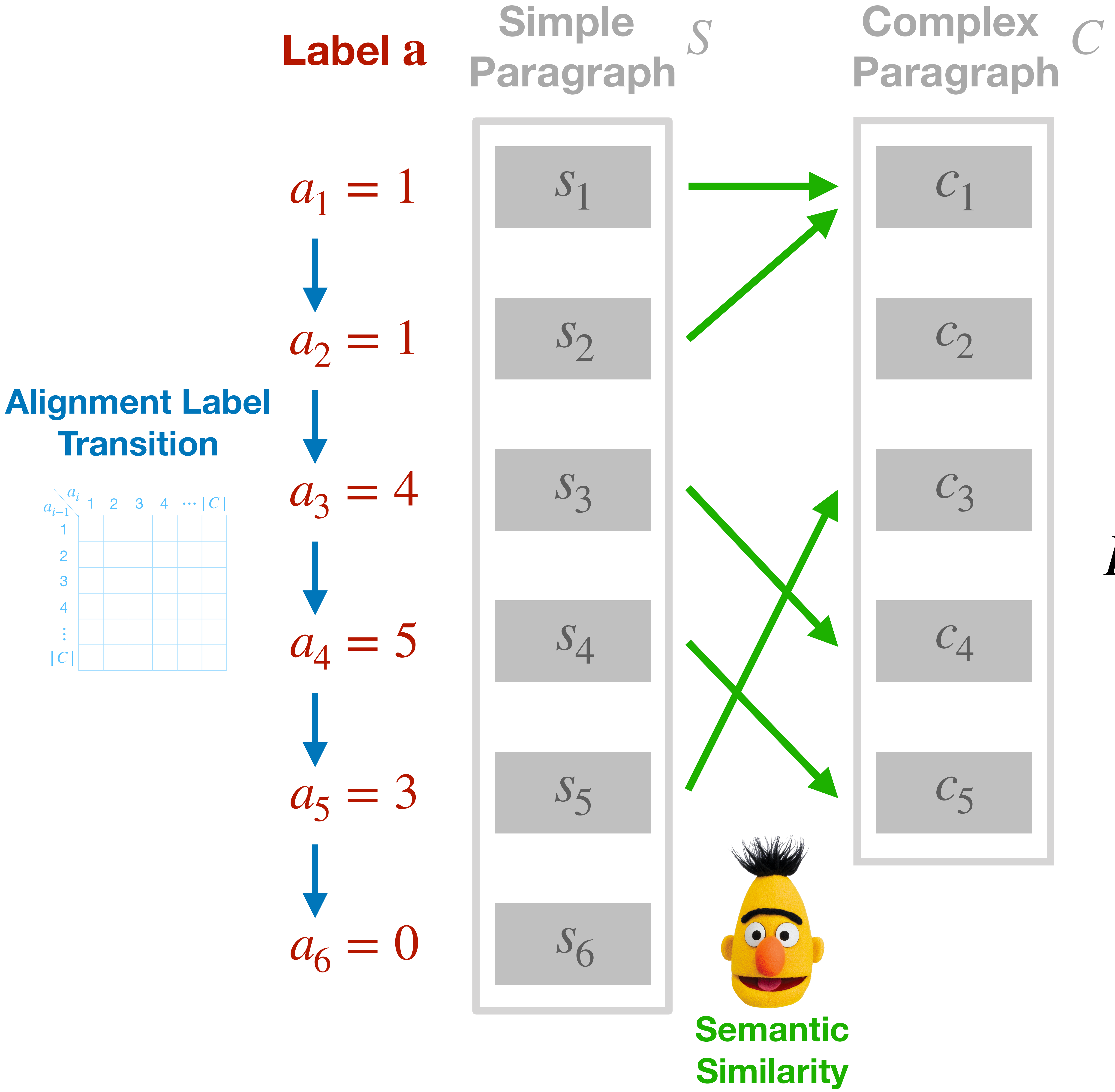


Figure 1: An example of sentence alignment between an original news article (right) and its simplified version (left) in Newsela. The label a_i for each simple sentence s_i is the index of complex sentence c_{a_i} it aligns to.

Chao Jiang, Mounica Maddela, Wuwei Lan, Yang Zhong, Wei Xu. “Neural CRF Model for Sentence Alignment in Text Simplification” (ACL 2020)

Sentence Alignment



Semantic Similarity

Alignment Label Transition

$$\Psi(\mathbf{a}, S, C) = \sum_{i=0}^{|S|} sim(s_i, c_{a_i}) + T(a_i, a_{i-1})$$

$$P(\mathbf{a} \mid S, C) = \frac{\exp(\Psi(\mathbf{a}, S, C))}{\sum_{\mathbf{a} \in \mathbf{A}} \exp(\Psi(\mathbf{a}, S, C))}$$

Sequential CRF

all possible alignments (dynamic programming)

Chao Jiang, Mounica Maddela, Wuwei Lan, Yang Zhong, Wei Xu. "Neural CRF Model for Sentence Alignment in Text Simplification" (ACL 2020)

Sentence Alignment

- Structure prediction + BERT$_{finetune}$ → A neural CRF alignment model.

		aligned + partial vs. others*		
		Precision	Recall	F1
Greedy	JaccardAlign (Xu et al., 2015)	98.66	67.58	80.22
Dynamic Programming	MASSAlign (Paetzold et al., 2017)	95.49	82.27	88.39
Greedy	CATS (Štajner et al., 2018)	88.56	91.31	89.92
Threshold	BERT$_{finetune}$	94.99	89.62	92.22
Threshold	BERT$_{finetune}$ + paragraph alignment	98.05	88.63	93.10
CRF	Our CRF aligner	97.86	91.31	95.59

+5.7

* Results are on the manually annotated Newsela dataset.

Chao Jiang, Mounica Maddela, Wuwei Lan, Yang Zhong, Wei Xu. "Neural CRF Model for Sentence Alignment in Text Simplification" (ACL 2020)