

CS11-747 Neural Networks for NLP

Convolutional Networks for Text

Graham Neubig



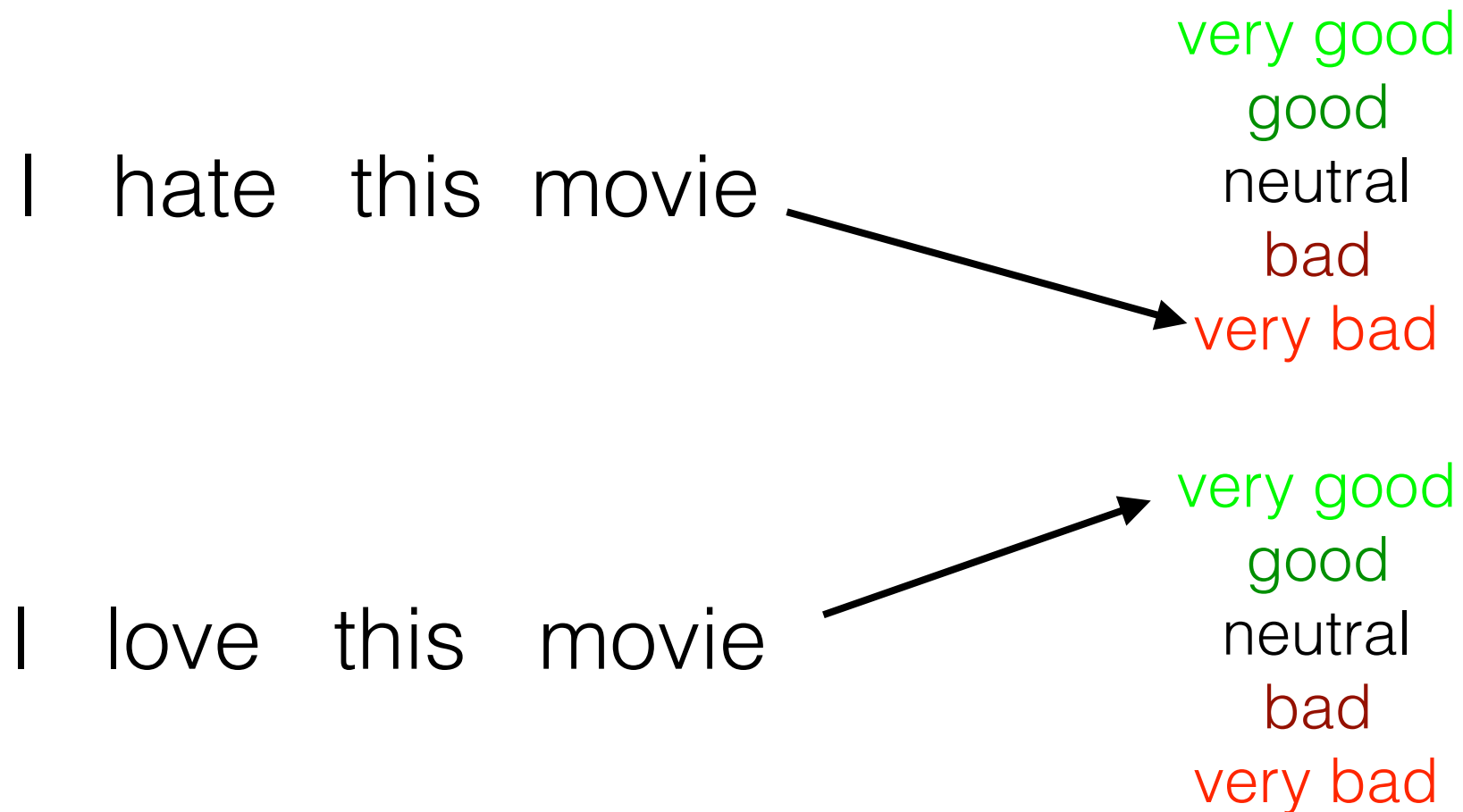
Carnegie Mellon University

Language Technologies Institute

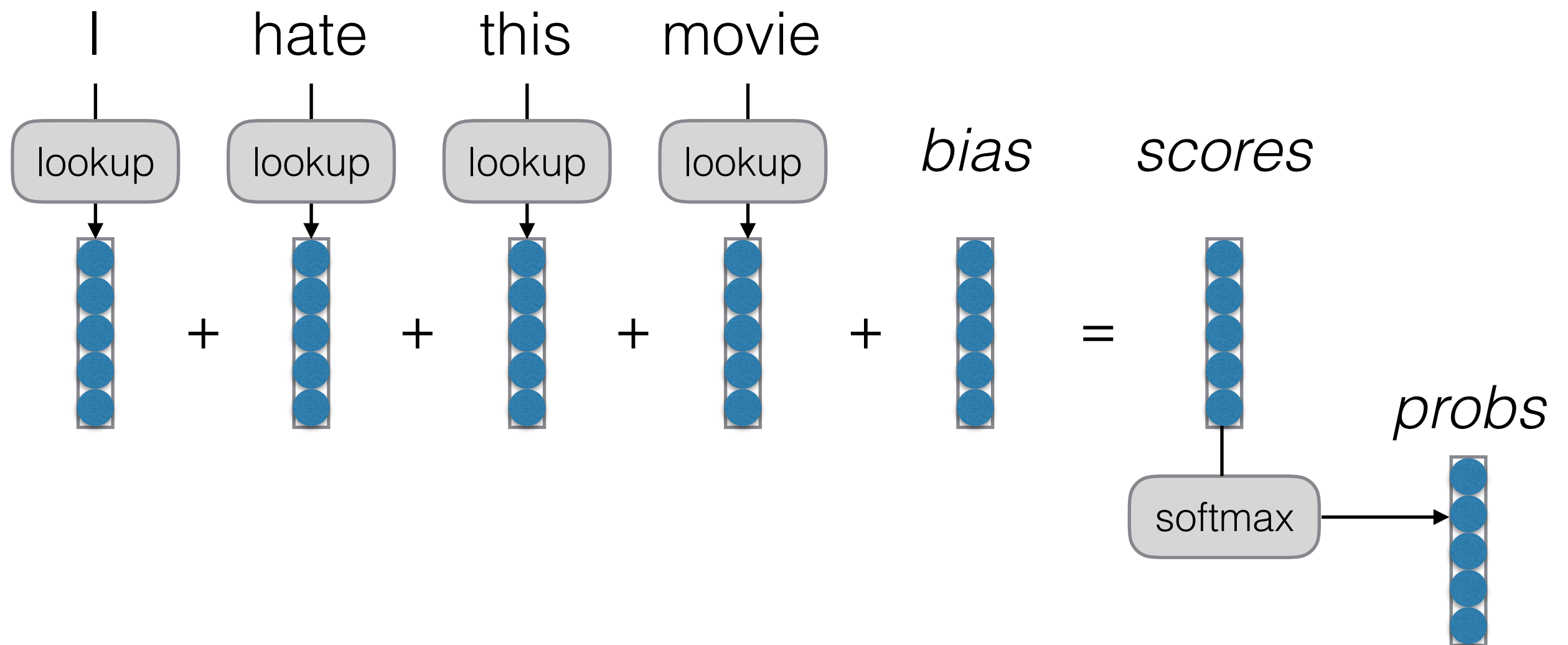
Site

<https://phontron.com/class/nn4nlp2019/>

An Example Prediction Problem: Sentence Classification




A First Try: Bag of Words (BOW)



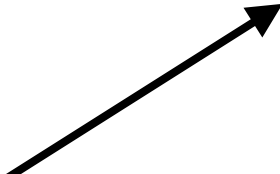
Build It, Break It

I don't love this movie



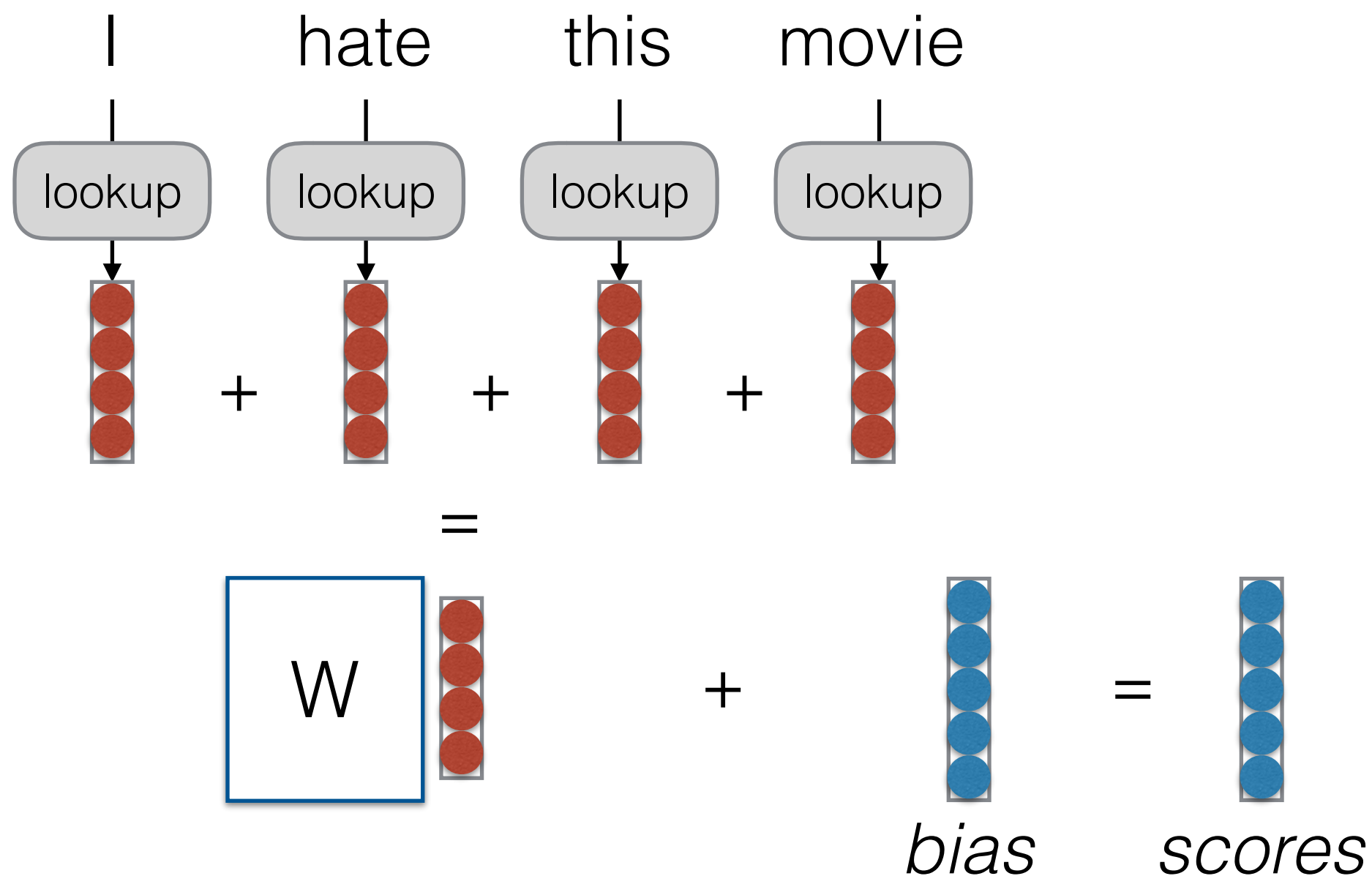
very good
good
neutral
bad
very bad

There's nothing I don't love about this movie

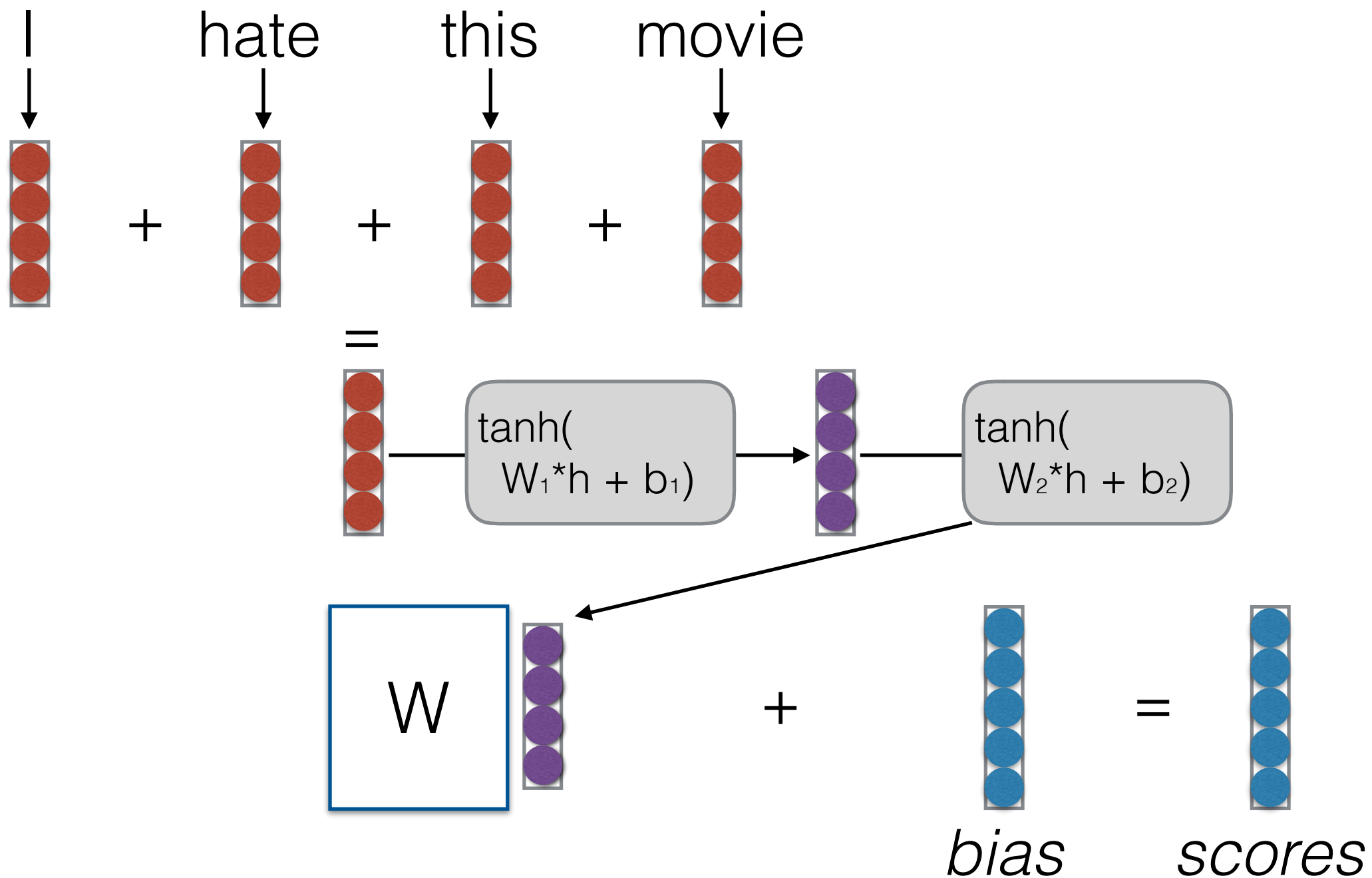


very good
good
neutral
bad
very bad

Continuous Bag of Words (CBOW)



Deep CBOW

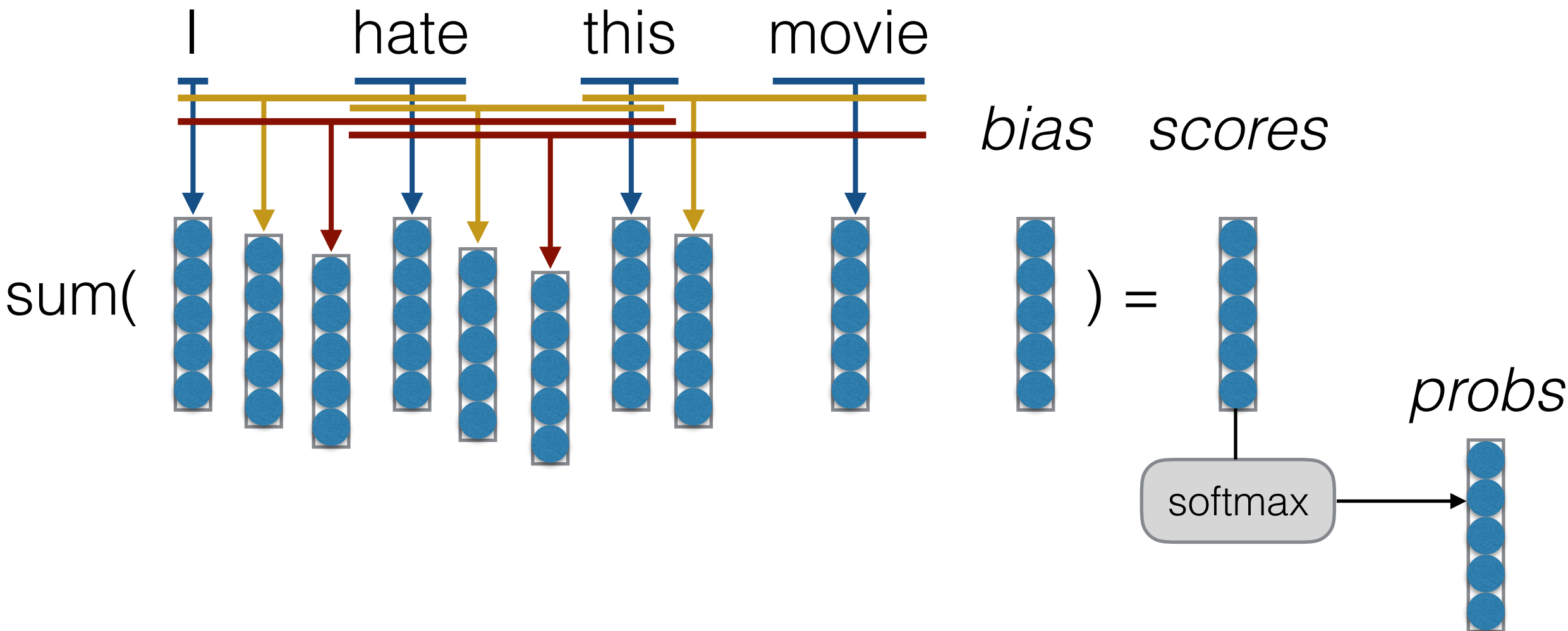


What do Our Vectors Represent?

- We can learn feature combinations (a node in the second layer might be “feature 1 AND feature 5 are active”)
- e.g. capture things such as “not” AND “hate”
- BUT! Cannot handle “not hate”

Handling Combinations

Bag of n-grams



Why Bag of n-grams?

- Allow us to capture combination features in a simple way “don’t love”, “not the best”
- Works pretty well



François Chollet  @fchollet · 2 Nov 2016

We are releasing an open dataset for theorem proving, HolStep:
openreview.net/forum?id=ryuxY... - can you beat our 83% accuracy baseline?

1

51

123



Hal Daumé III @haldaume3 · 2 Nov 2016

.@fchollet sure, I'll play. 85%, took me about an hour. (totally possible I did something wrong in preprocessing though!)

```
cat train/* | ./holstep2vw.pl | shuffle | vw --binary --loss_function logistic --ngram 6 -k -c --passes
5 -b33 -f model.ngram6 --holdout_off
0.269470 0.247070 16384 16384.0 -1.0000 -1.0000 2859
0.239288 0.209106 32768 32768.0 -1.0000 -1.0000 4791
0.210785 0.182281 65536 65536.0 1.0000 1.0000 2085
0.184792 0.158798 131072 131072.0 1.0000 1.0000 4023
0.166405 0.148018 262144 262144.0 -1.0000 -1.0000 9369
0.152111 0.137817 524288 524288.0 1.0000 1.0000 1881
0.138991 0.125872 1048576 1048576.0 1.0000 1.0000 3393
0.127713 0.116435 2097152 2097152.0 1.0000 1.0000 1929
0.104631 0.081549 4194304 4194304.0 -1.0000 -1.0000 1797
0.086621 0.068610 8388608 8388608.0 1.0000 -1.0000 1323

finished run
number of examples per pass = 2013046
passes used = 5
weighted example sum = 10065230.000000
weighted label sum = 0.000000
average loss = 0.082794
best constant = 0.000000
best constant's loss = 0.693147
total feature number = 29140509425

% cat test/* | ./holstep2vw.pl | vw --binary -i model.ngram6 -t
average loss = 0.146743

% cat holstep2vw.pl
#!/usr/bin/perl -w
use strict;

my $conJName = ''; my $conJText = ''; my $conJTok = '';
my $depName = ''; my $depText = ''; my $depTok = '';
while (<>) {
    chomp;
    if (/^N/) {
        s/^.\s*//; $conJName = $_;
        $_ = <>; die if not /^C/; s/^.\s*//; $conJText = tokenize($_);
        $_ = <>; die if not /^T/; s/^.\s*//; $conJTok = $_;
    } elsif (/^D/) {
        s/^.\s*//; $depName = $_;
        $_ = <>; die if not /^A/; s/^.\s*//; $depText = tokenize($_);
        $_ = <>; die if not /^T/; s/^.\s*//; $depTok = $_;
    } elsif (/^I/) {
        my $stepLabel = (/^+/? 1 : -1;
        s/^.\s*//; my $stepText = tokenize($_);
        $_ = <>; die if not /^T/; s/^.\s*//; my $stepTok = $_;
        print "$stepLabel\t", 'IC', vu($conJName), 'ID',
            vu($conJTok), 'IS', vu($stepTok), 'IX', vu($conJText), 'IY', vu($depText), 'IZ', vu($ste
        pText), "\n";
    } else { die $_; }
}

sub vu {
    my ($t) = @_;
    chomp $t; $t =~ s/;/_C_/g; $t =~ s/|/_P_/g;
    return $t;
}

sub tokenize {
    my ($t) = @_;
    $t =~ s/([(){}+])/ $1 /g;
    return $t;
}
```

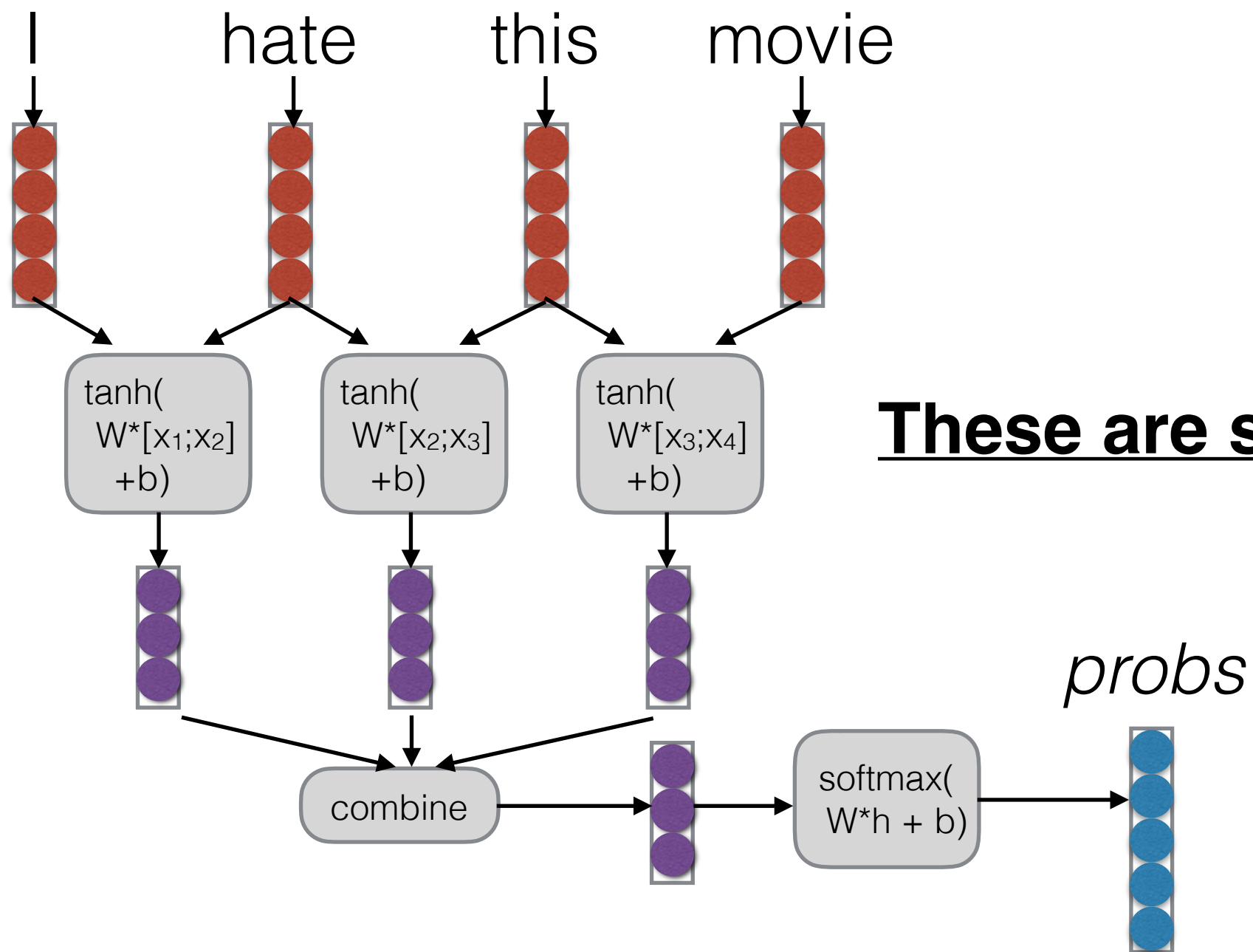
What Problems w/ Bag of n-grams?

- Same as before: parameter explosion
- No sharing between similar words/n-grams

Convolutional Neural Networks (Time-delay Neural Networks)

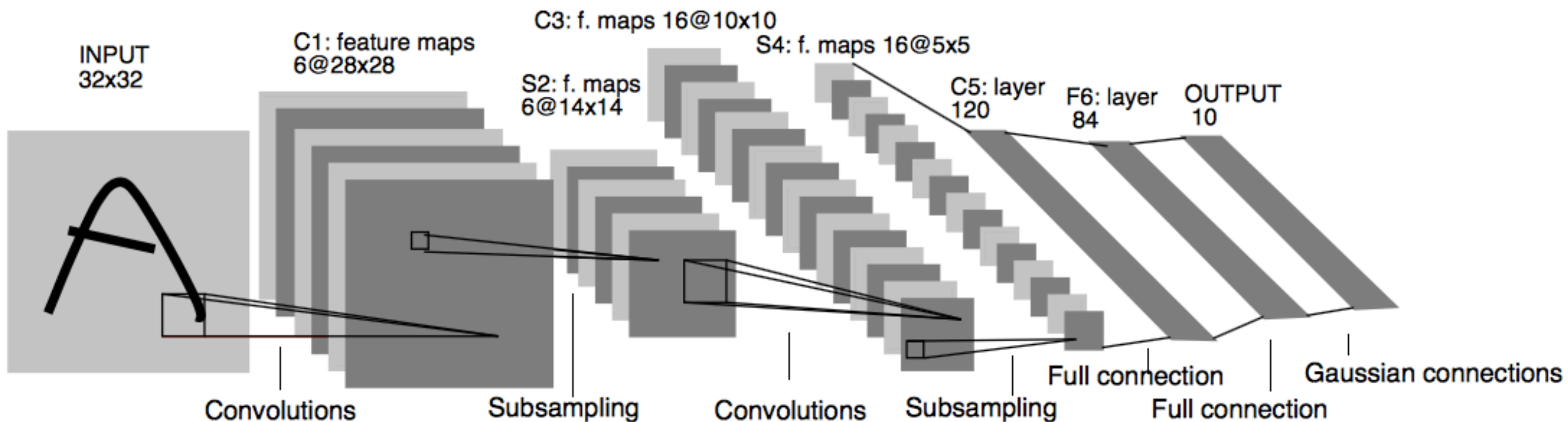
1-dimensional Convolutions / Time-delay Networks

(Waibel et al. 1989)



2-dimensional Convolutional Networks

(LeCun et al. 1997)



Parameter extraction performs a 2D sweep, not 1D

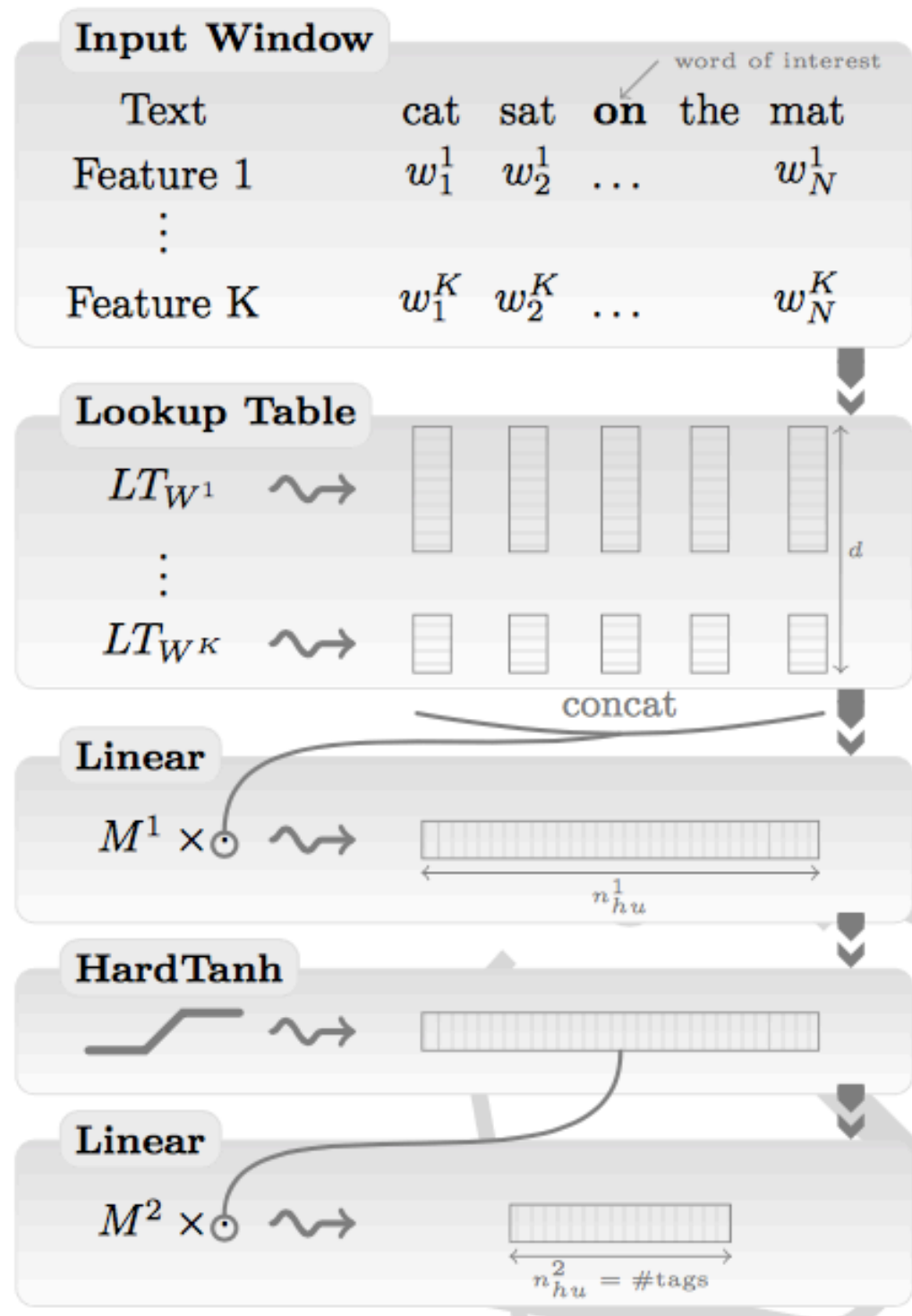
CNNs for Text

(Collobert and Weston 2011)

- Generally based on 1D convolutions
 - But often uses terminology/functions borrowed from image processing for historical reasons
- Two main paradigms:
 - **Context window modeling:** For tagging, etc. get the surrounding context before tagging
 - **Sentence modeling:** Do convolution to extract n-grams, pooling to combine over whole sentence

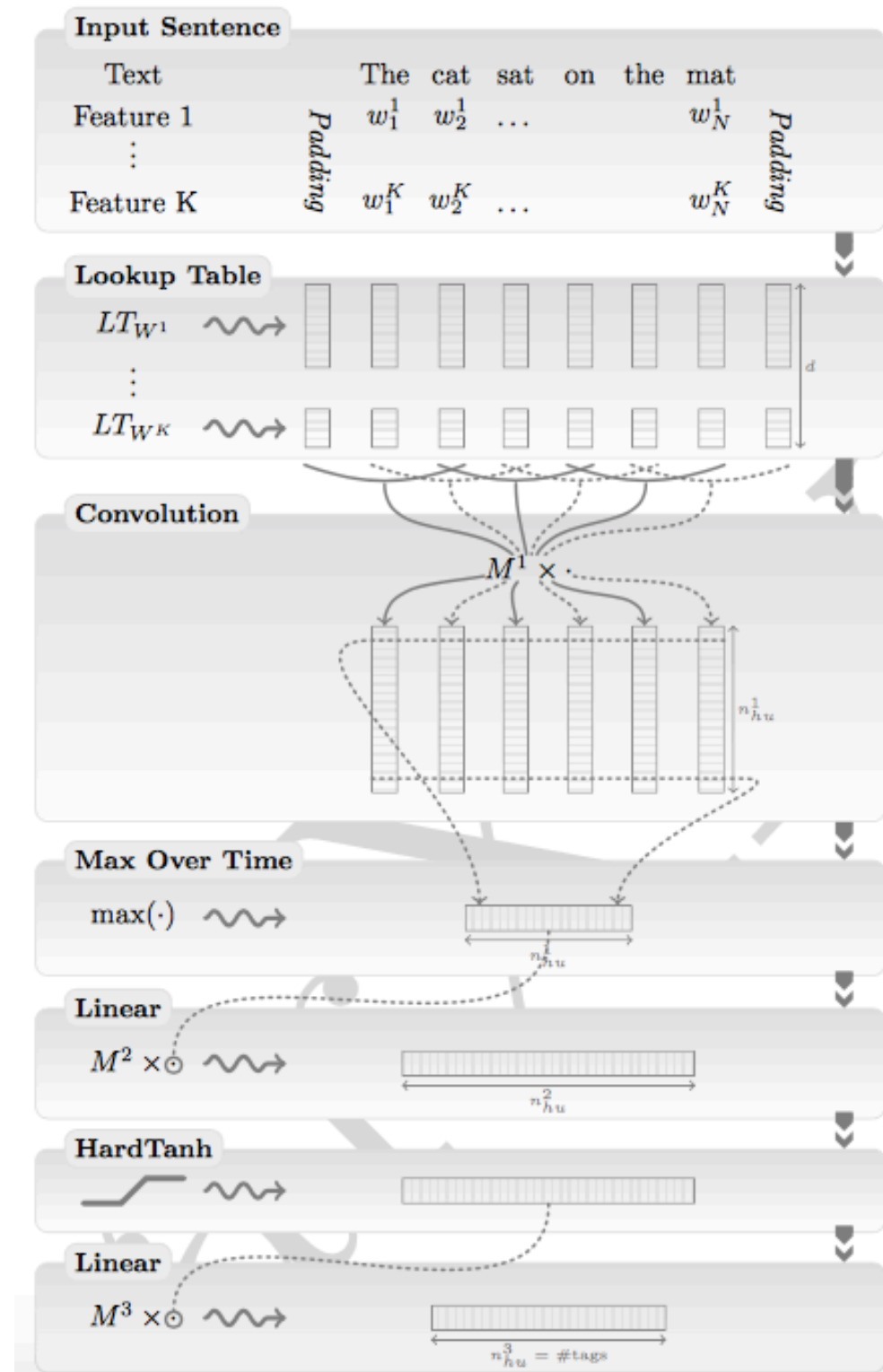
CNNs for Tagging

(Collobert and Weston 2011)



CNNs for Sentence Modeling

(Collobert and Weston 2011)

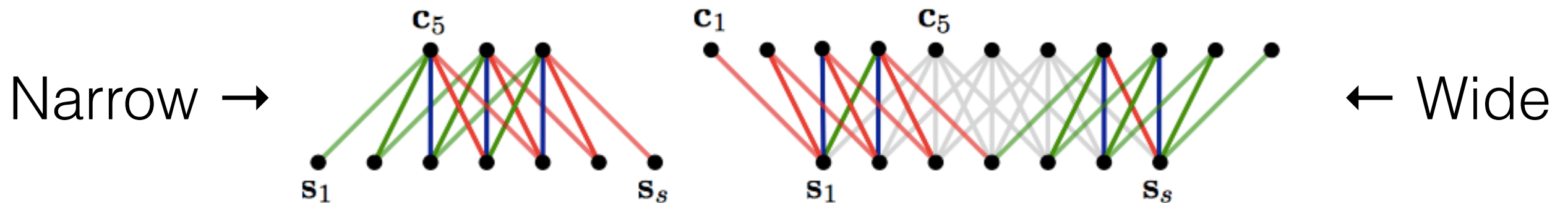


Standard conv2d Function

- 2D convolution function takes input + parameters
- **Input:** 3D tensor
 - rows (e.g. words), columns, features (“channels”)
- **Parameters/Filters:** 4D tensor
 - rows, columns, input features, output features

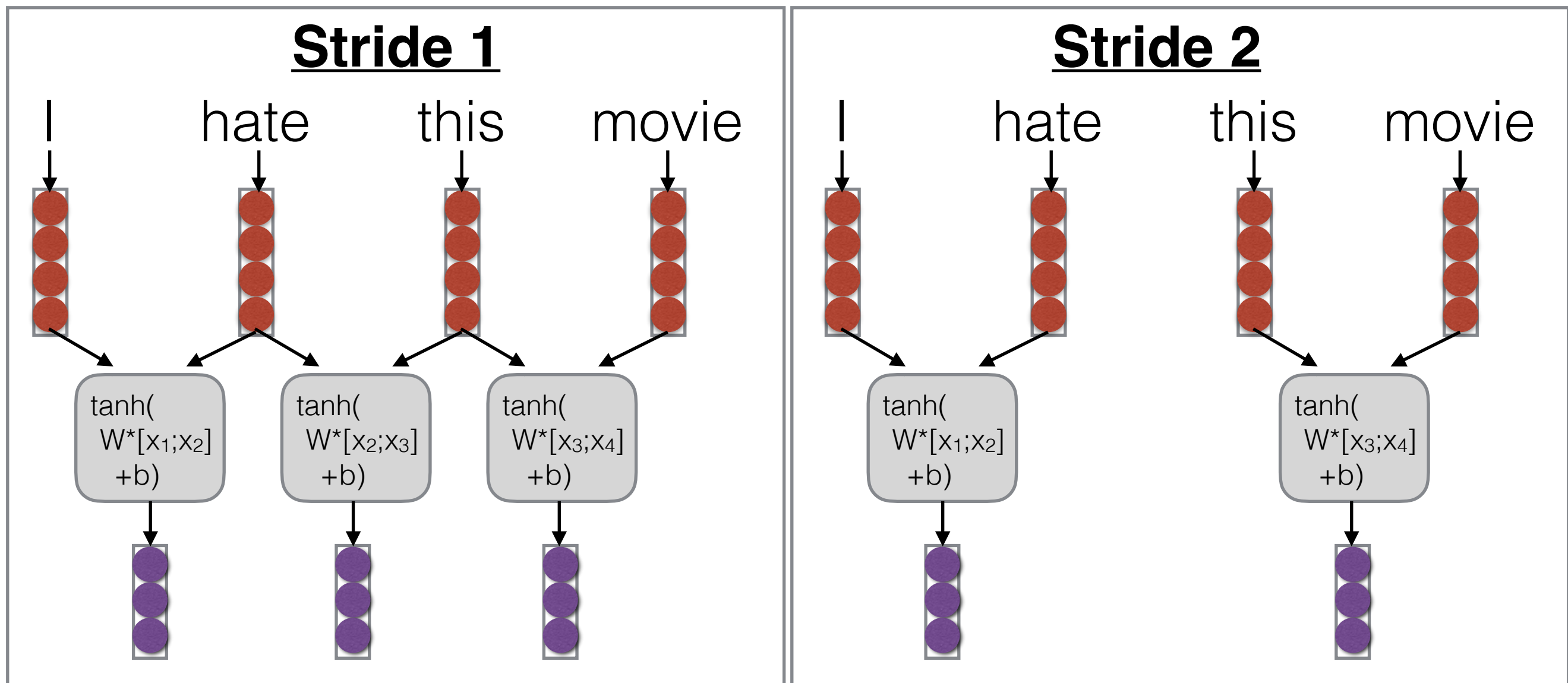
Padding

- After convolution, the rows and columns of the output tensor are either
 - = to rows/columns of input tensor (*“same”* convolution)
 - = to rows/columns of input tensor minus the size of the filter plus one (*“valid”* or *“narrow”*)
 - = to rows/columns of input tensor plus filter minus one (*“wide”*)



Striding

- Skip some of the outputs to reduce length of extracted feature vector



Pooling

- Pooling is like convolution, but calculates some reduction function feature-wise
- **Max pooling:** “Did you see this feature anywhere in the range?” (most common)
- **Average pooling:** “How prevalent is this feature over the entire range”
- **k-Max pooling:** “Did you see this feature up to k times?”
- **Dynamic pooling:** “Did you see this feature in the beginning? In the middle? In the end?”

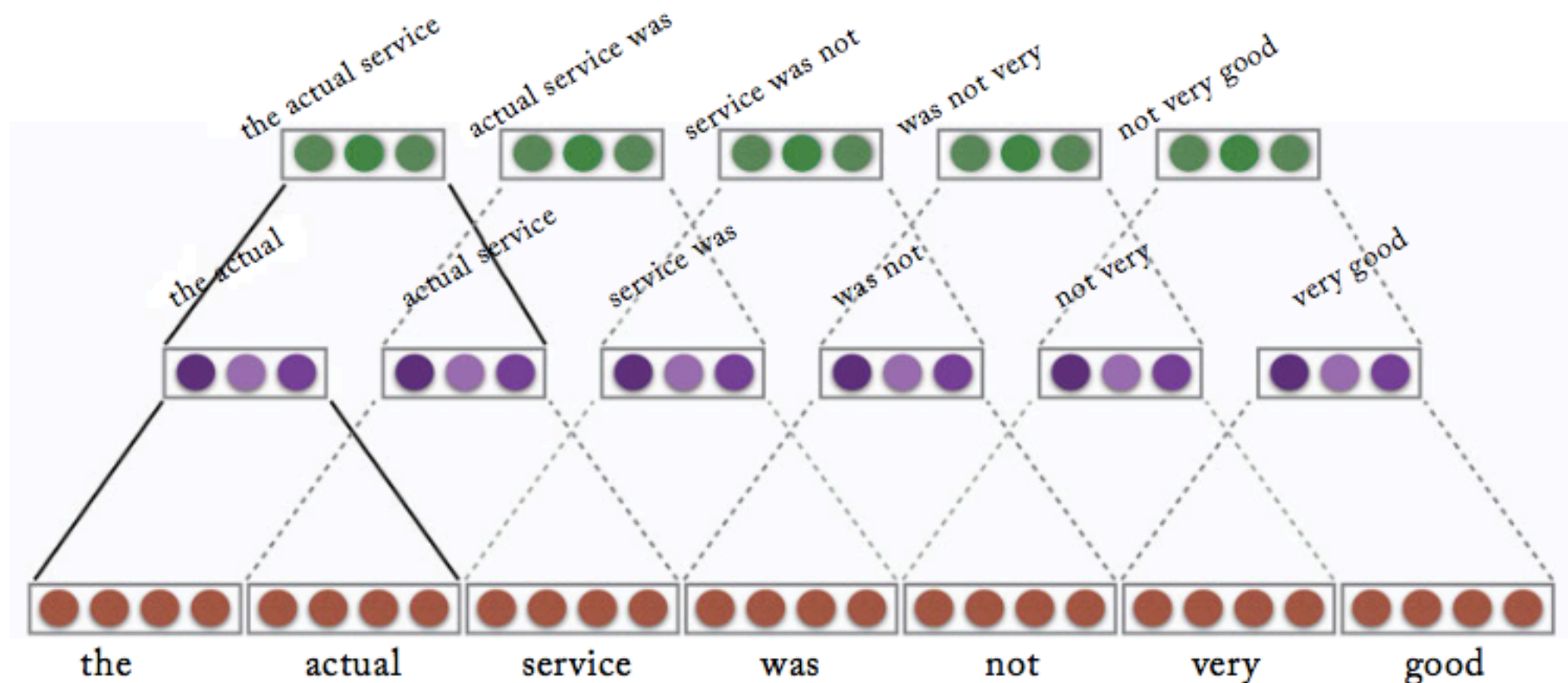
Let's Try It!

`cnn-class.py`

Stacked Convolution

Stacked Convolution

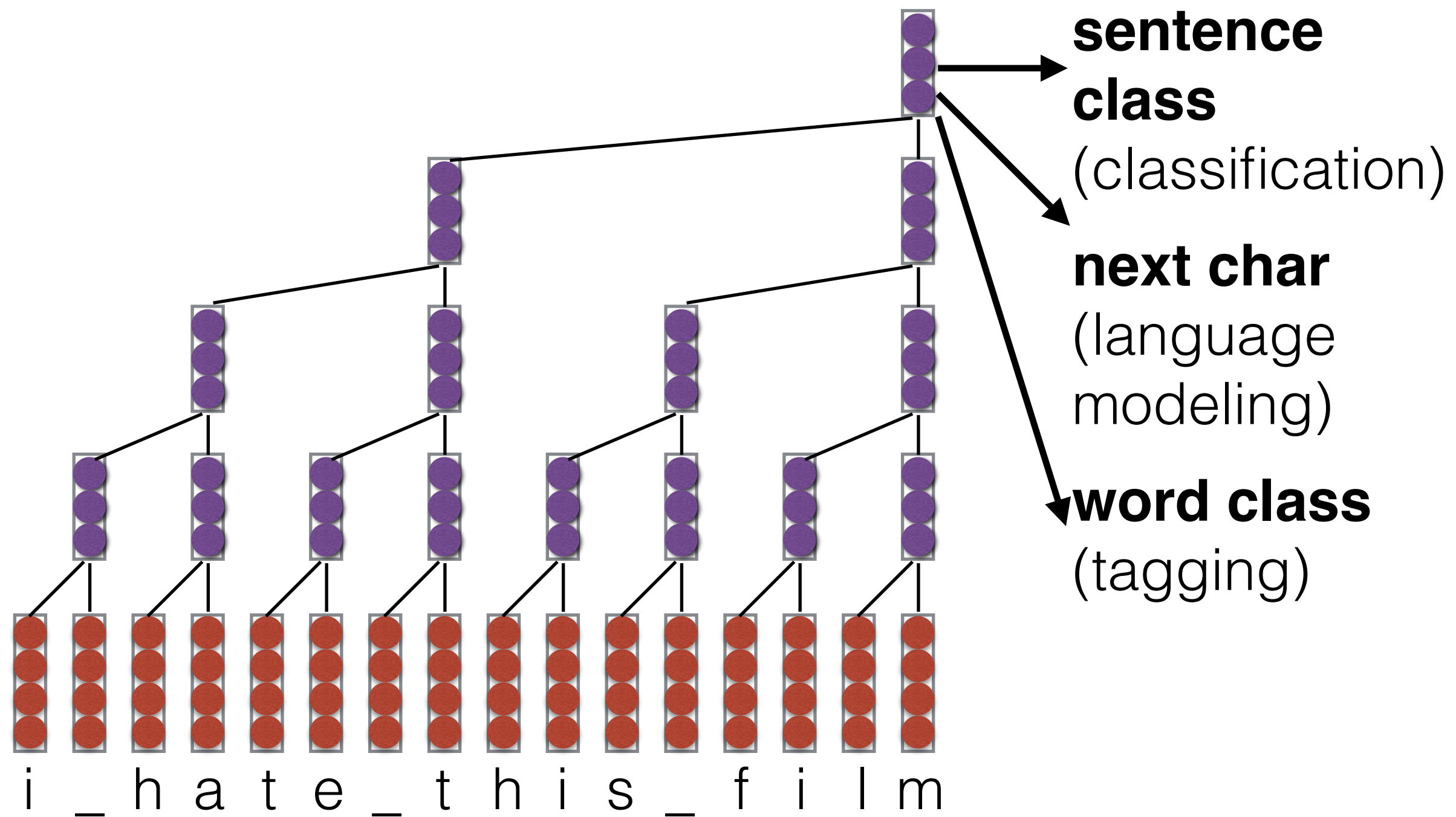
- Feeding in convolution from previous layer results in larger area of focus for each feature



Dilated Convolution

(e.g. Kalchbrenner et al. 2016)

- **Gradually increase stride**, every time step (no reduction in length)



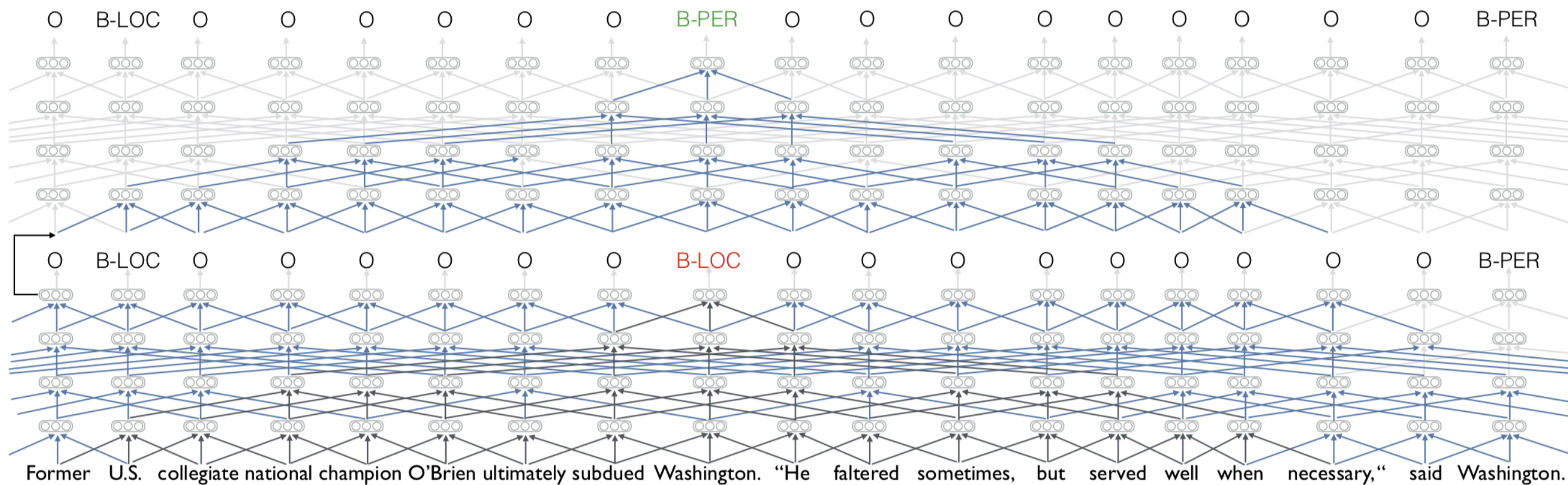
Why (Dilated) Convolution for Modeling Sentences?

- In contrast to recurrent neural networks (next class)
- + Fewer steps from each word to the final representation: RNN $O(N)$, Dilated CNN $O(\log N)$
- + Easier to parallelize on GPU
- - Slightly less natural for arbitrary-length dependencies
- - A bit slower on CPU?

Iterated Dilated Convolution

(Strubell+ 2017)

- Multiple iterations of the same stack of dilated convolutions



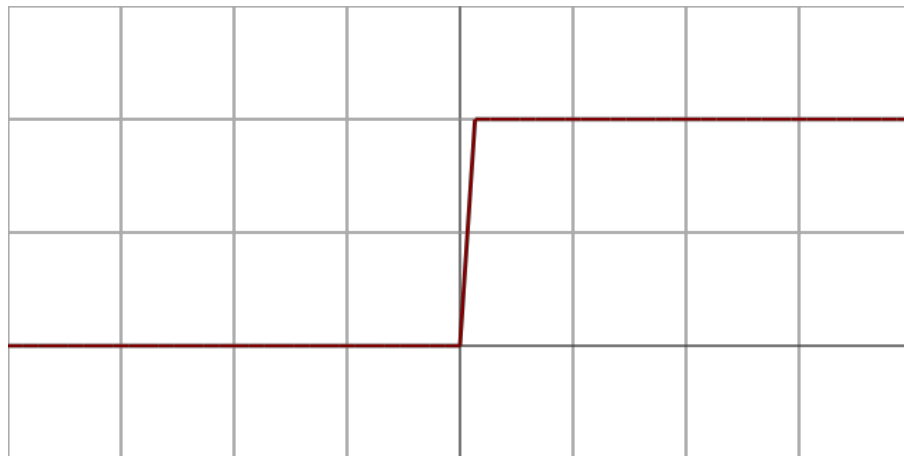
- Wider context, more parameter efficient

An Aside: Non-linear Functions

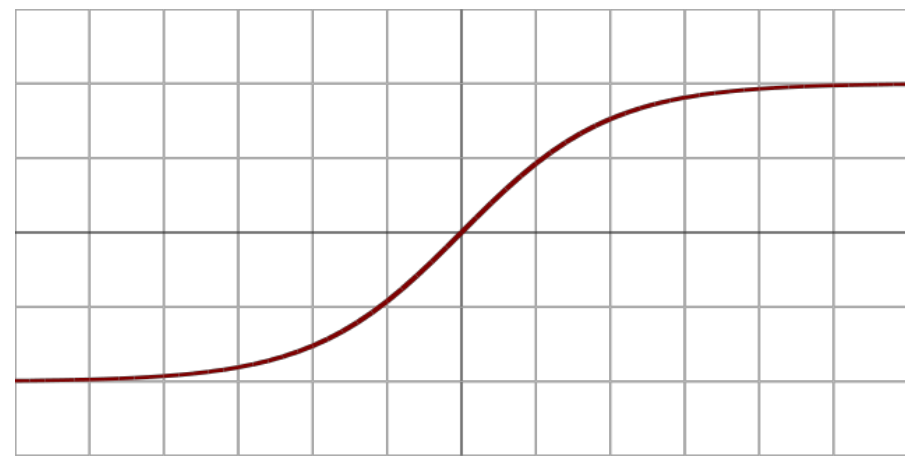
Non-linear Functions

- Proper choice of a non-linear function is essential in stacked networks

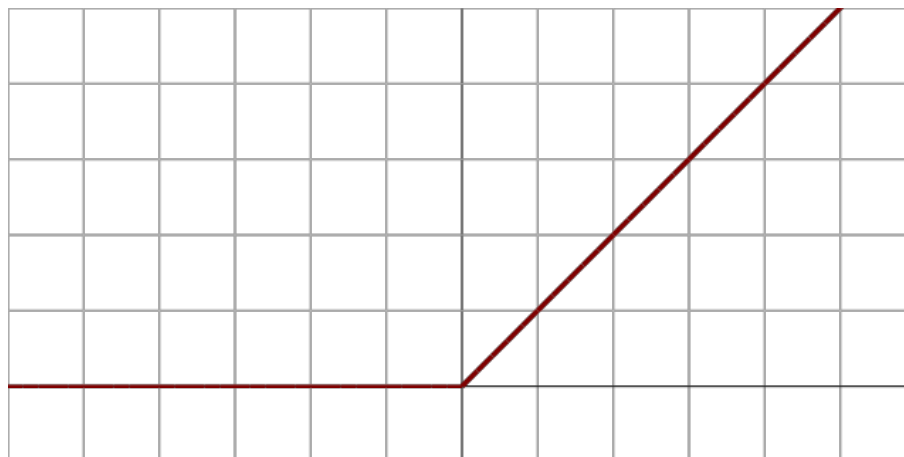
step



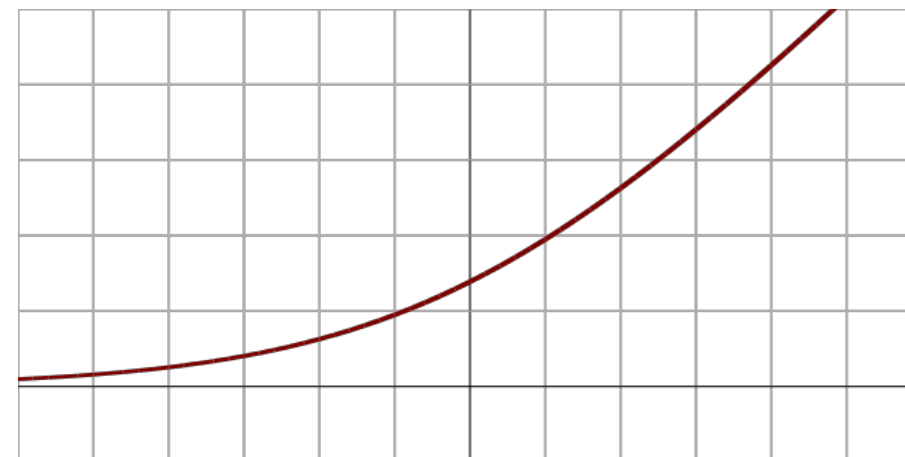
tanh



rectifier
(ReLU)



soft
plus



- Functions such as ReLU or softplus allegedly better at preserving gradients

Which Non-linearity Should I Use?

- Ultimately an empirical question
- Many new functions proposed, but search by Eger et al. (2018) over NLP tasks found that standard functions such as tanh and relu quite robust

sigmoid	$f(x) = \sigma(x) = 1/(1 + \exp(-x))$
swish	$f(x) = x \cdot \sigma(x)$
maxsig	$f(x) = \max\{x, \sigma(x)\}$
cosid	$f(x) = \cos(x) - x$
minsin	$f(x) = \min\{x, \sin(x)\}$
arctid	$f(x) = \arctan(x)^2 - x$
maxtanh	$f(x) = \max\{x, \tanh(x)\}$
lrelu-0.01	$f(x) = \max\{x, 0.01x\}$
lrelu-0.30	$f(x) = \max\{x, 0.3x\}$
penalized tanh	$f(x) = \begin{cases} \tanh(x) & x > 0, \\ 0.25 \tanh(x) & x \leq 0 \end{cases}$
best	penalized tanh (6), swish (6), elu (4), relu (4), lrelu-0.01 (4)
mean	penalized tanh (16), tanh (13) sin (10)

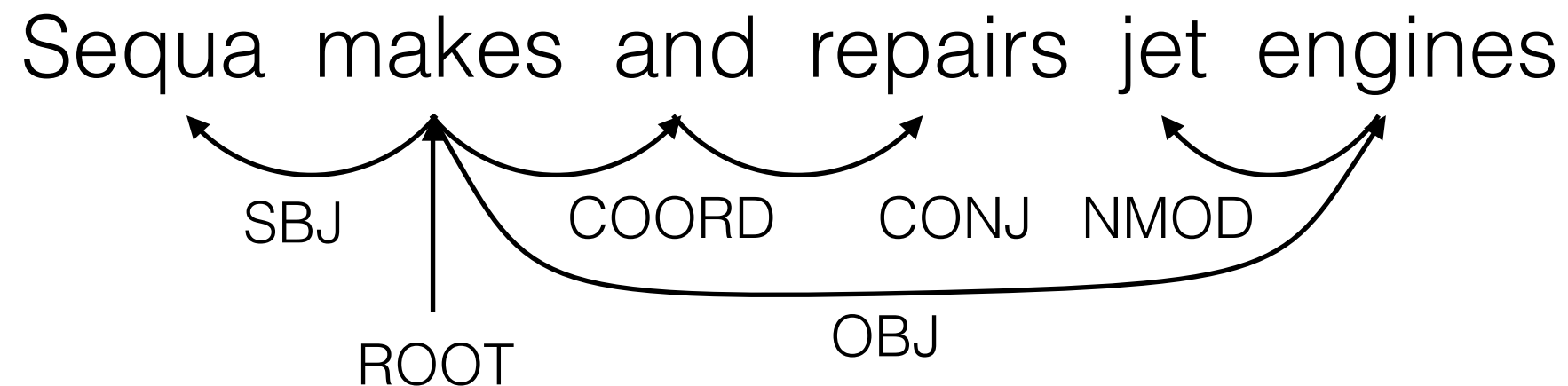
Table 5: Top-3 winner statistics. In brackets: number of times within top-3, keeping only functions with four or more top-3 rankings.

Structured Convolution

Why Structured Convolution?

- Language has structure, would like it to localize features
- e.g. noun-verb pairs very informative, but not captured by normal CNNs

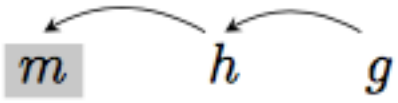
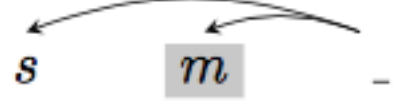

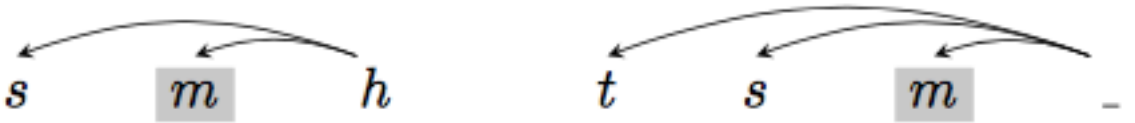
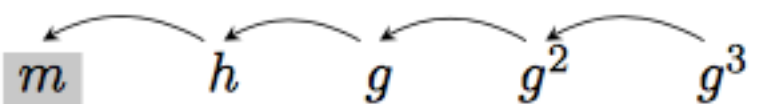
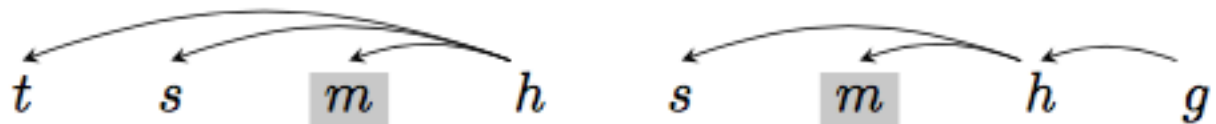
Example: Dependency Structure



Tree-structured Convolution

(Ma et al. 2015)

- Convolve over parents, grandparents, siblings

ancestor paths		siblings	
n	pattern(s)	n	pattern(s)
3		2	
4		3	
5		4	

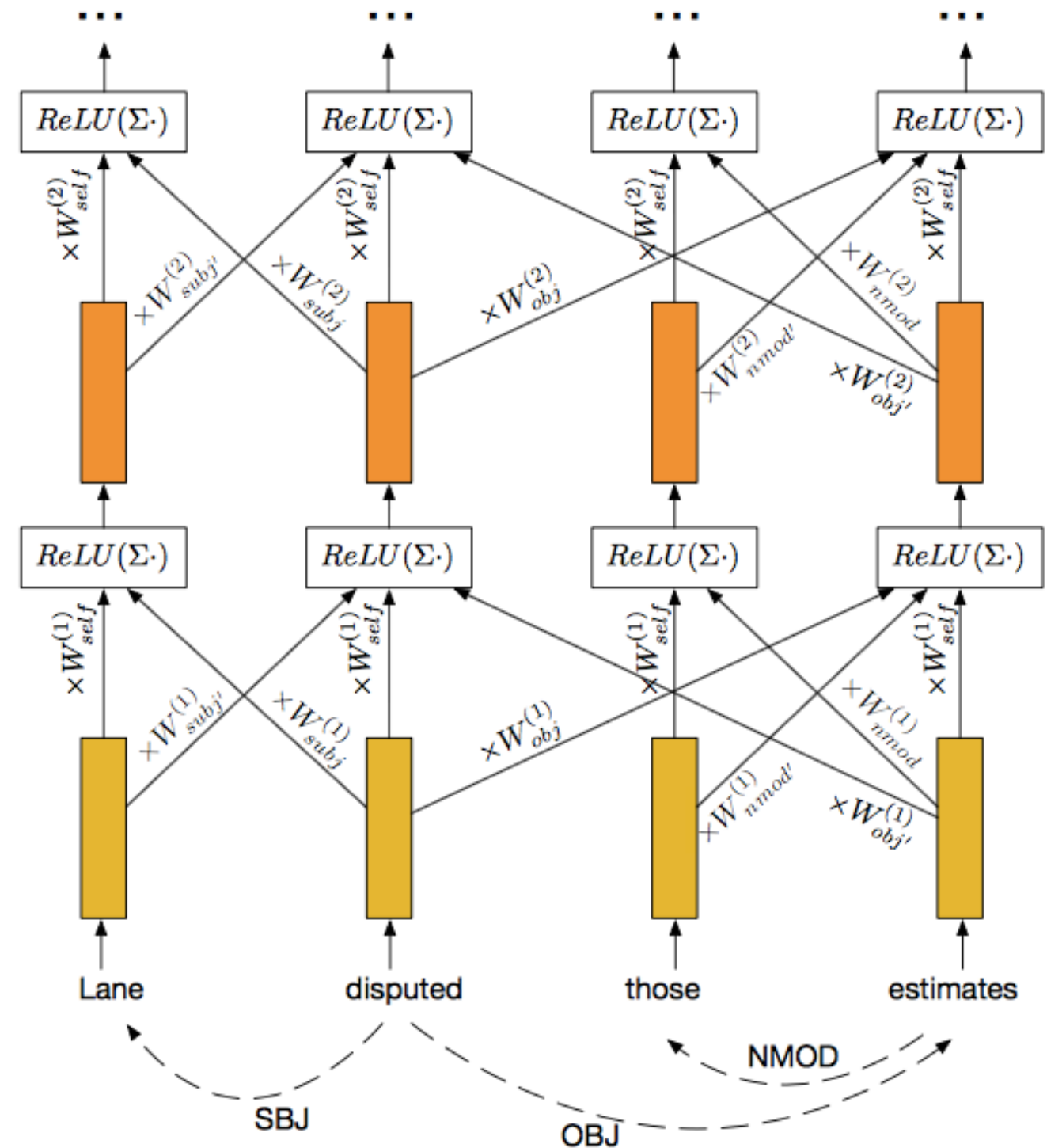
Graph Convolution

(e.g. Marcheggiani et al. 2017)

- Convolution is shaped by graph structure

- For example, dependency tree is a graph with

- Self-loop connections
- Dependency connections
- Reverse connections



Convolutional Models of Sentence Pairs

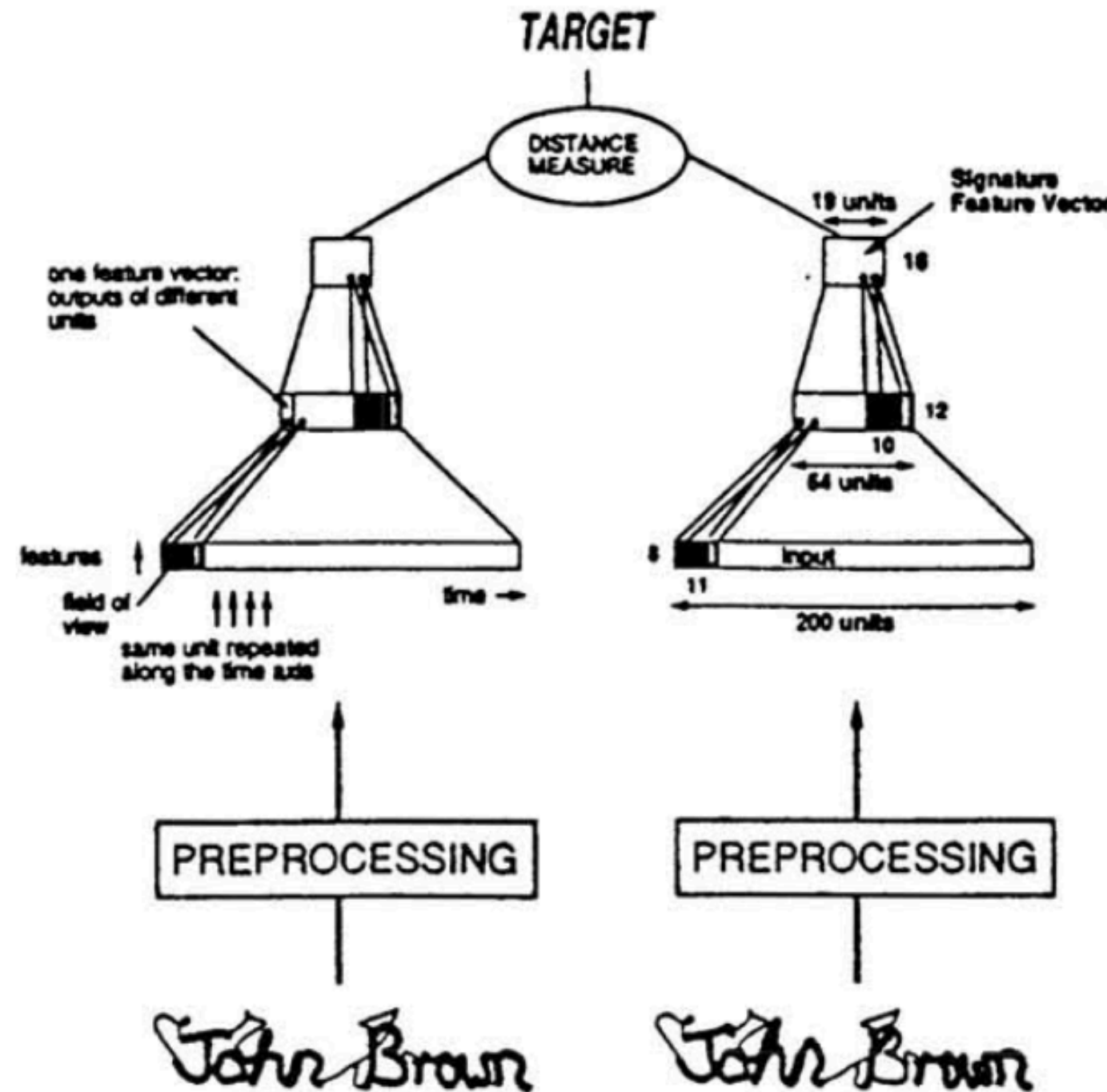
Why Model Sentence Pairs?

- Paraphrase identification / sentence similarity
- Textual entailment
- Retrieval
- (More about these specific applications in two classes)

Siamese Network

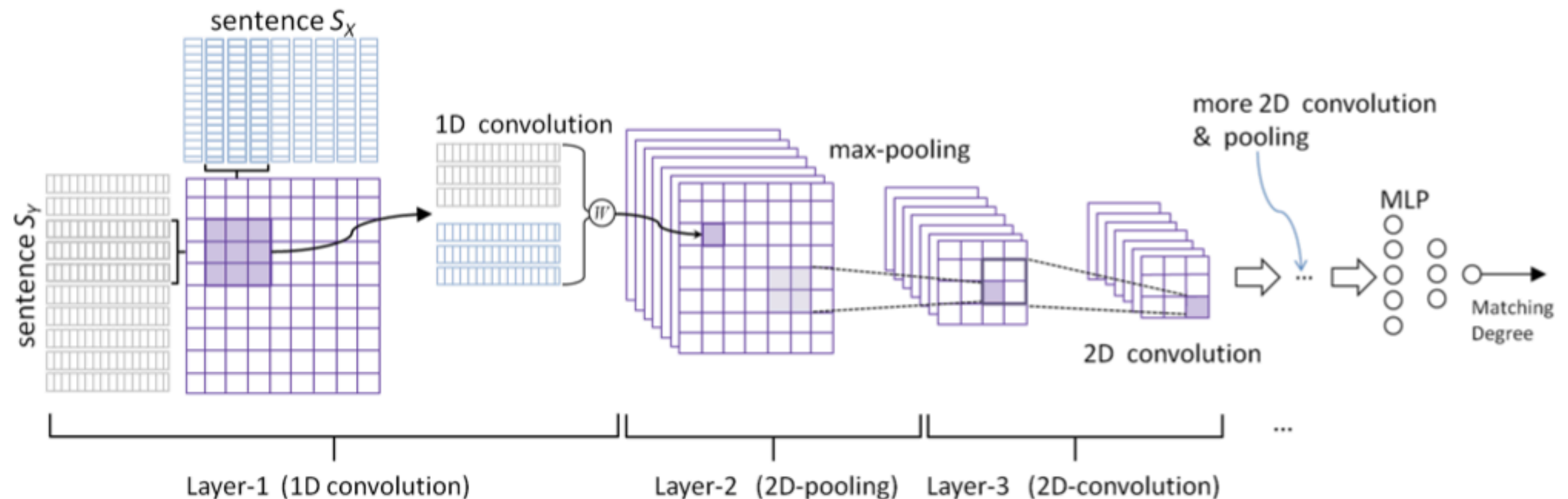
(Bromley et al. 1993)

- Use the same network, compare the extracted representations
- (e.g. Time-delay networks for signature recognition)



Convolutional Matching Model (Hu et al. 2014)

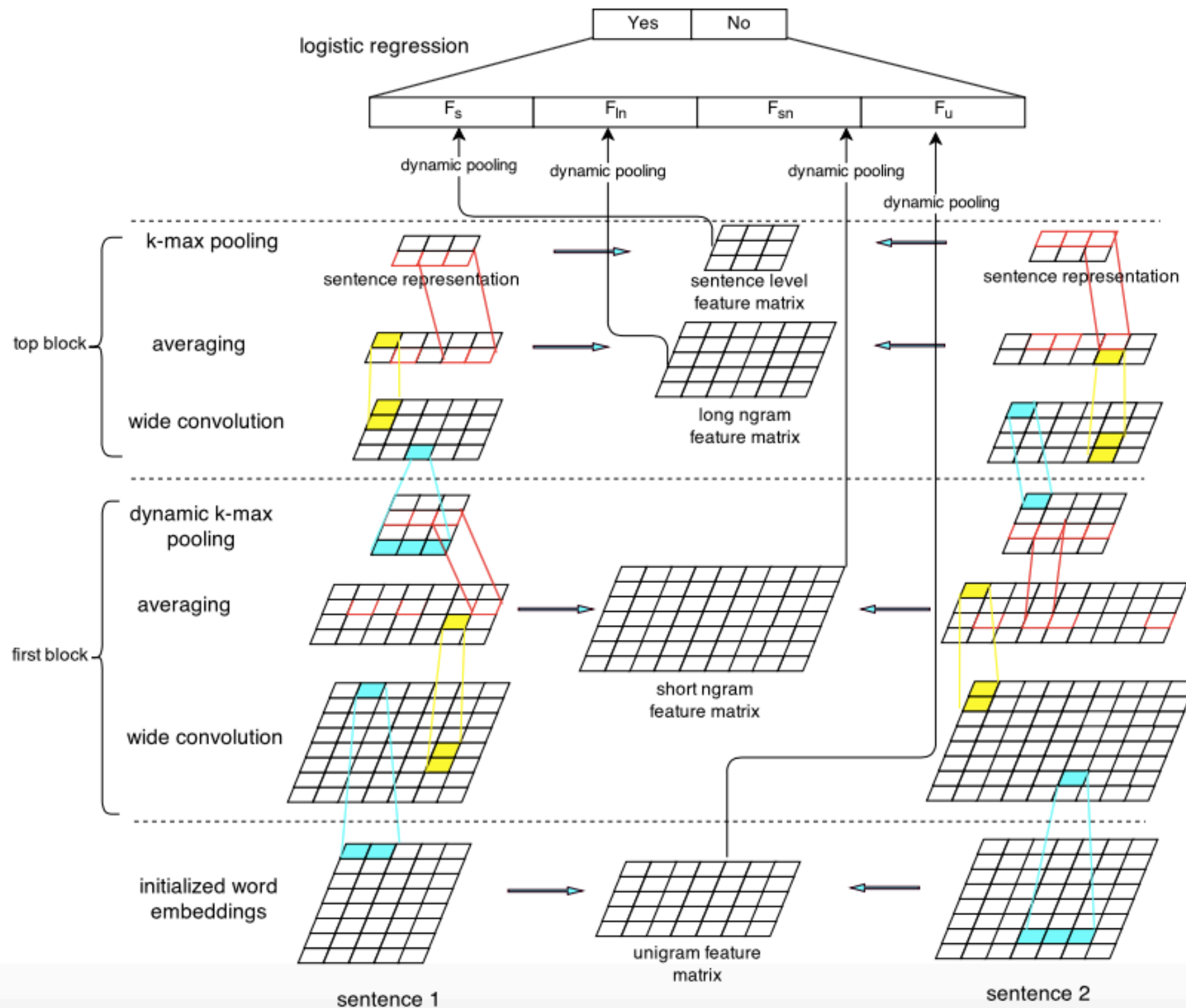
- Concatenate sentences into a 3D tensor and perform convolution



- Shown more effective than simple Siamese network

Convolutional Features

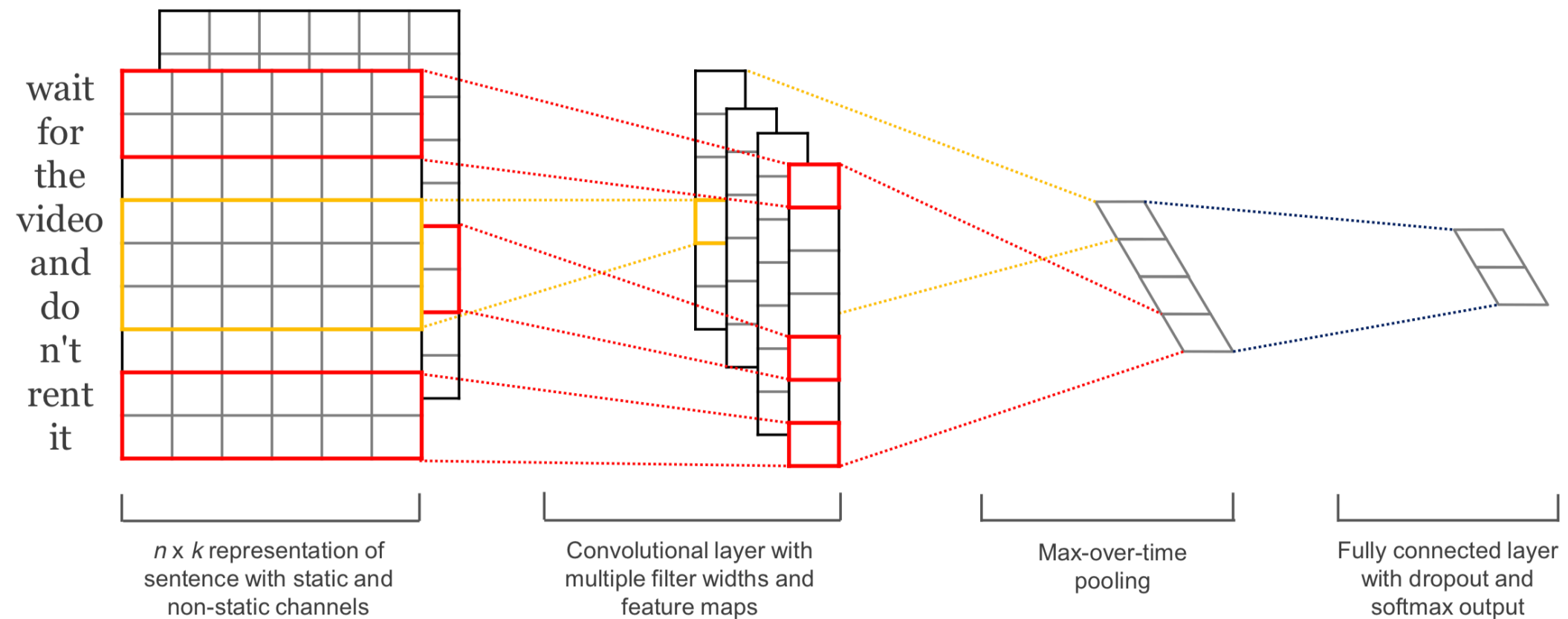
+ Matrix-based Pooling (Yin and Schutze 2015)



Case Study:
Convolutional Networks for Text
Classification (Kim 2015)

Convolution for Sentence Classification

(Kim 2014)



- Different widths of filters for the input
- Dropout on the penultimate layer
- Pre-trained or fine-tuned word vectors
- State-of-the-art or competitive results on sentence classification (at the time)

Questions?