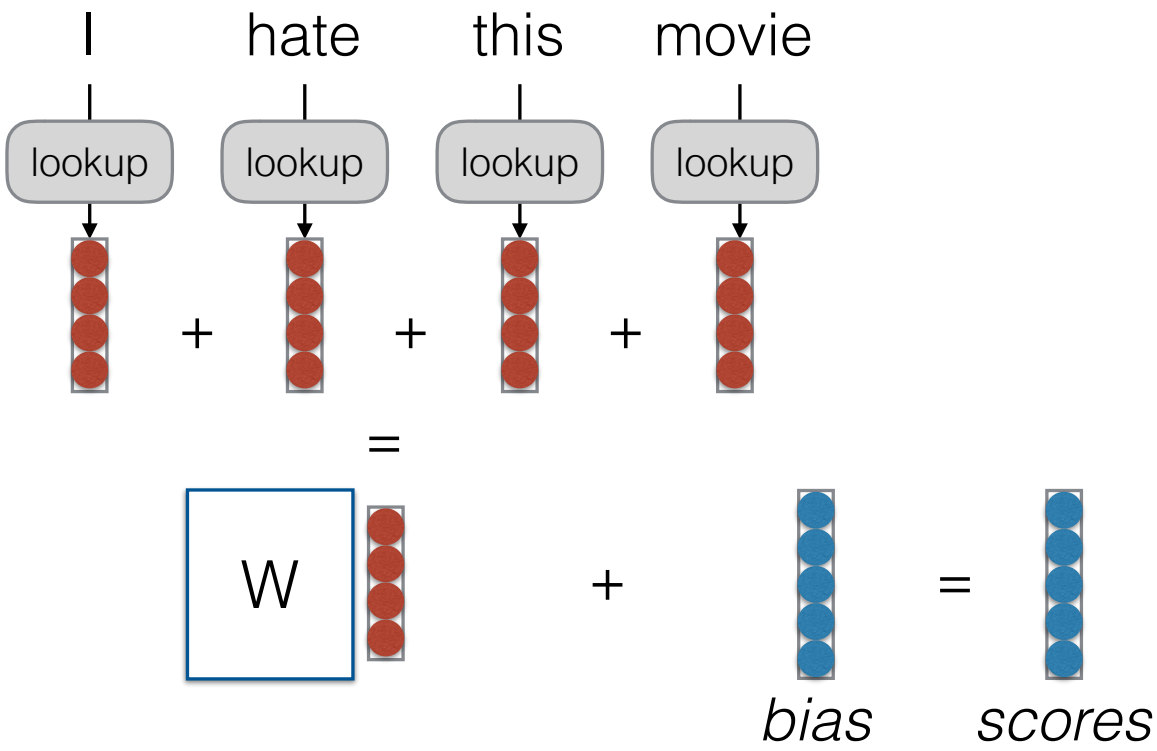
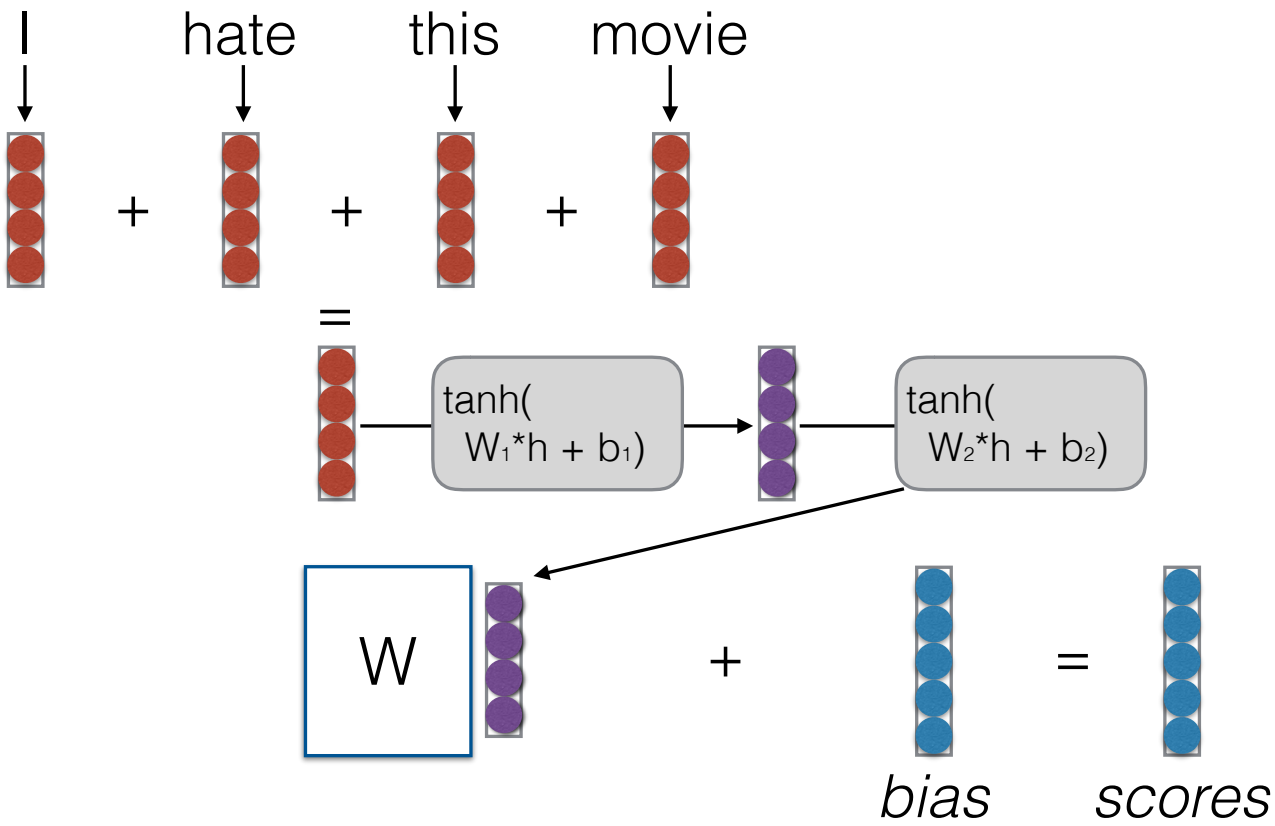
Continuous Bag of Words (CBOW)



A slide from Graham Neubig's CS11-747 Neural Networks for NLP in CMU2019

Deep CBOW



A slide from Graham Neubig's CS11-747 Neural Networks for NLP in CMU2019

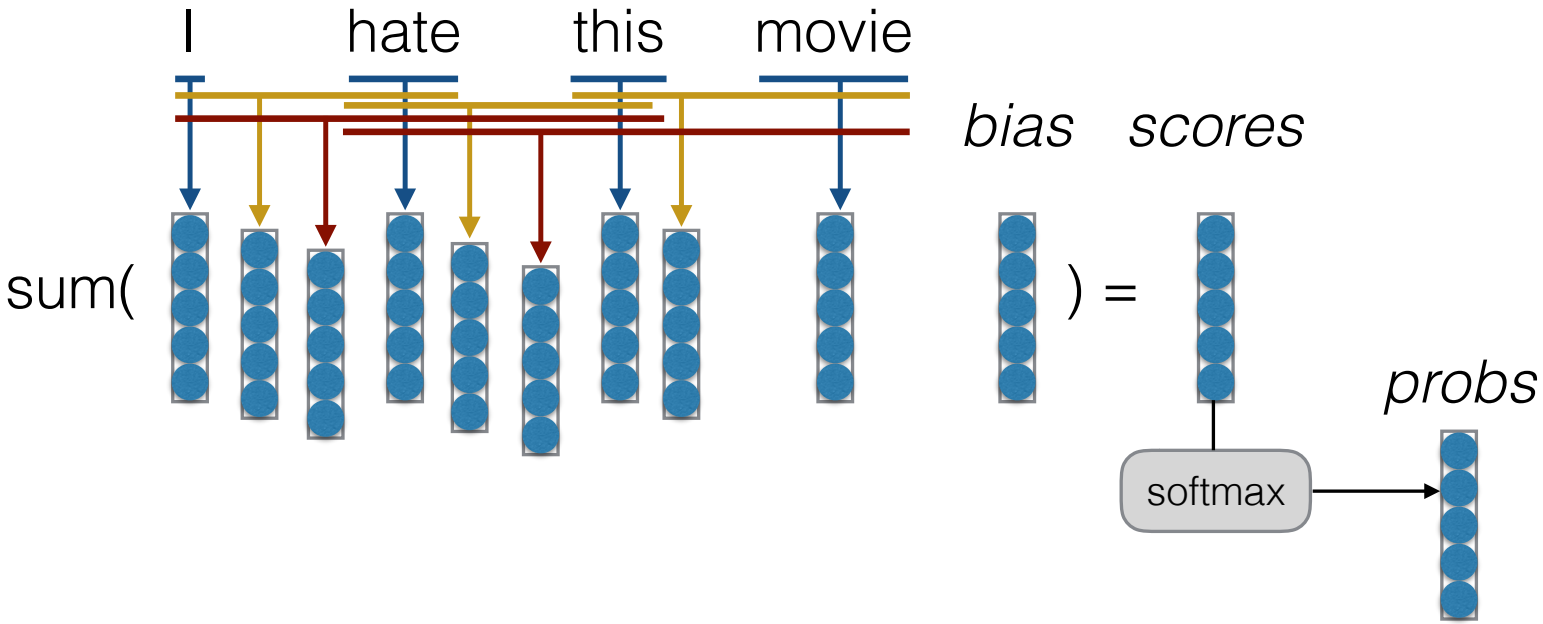
What do Our Vectors Represent?

- 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”

A slide from Graham Neubig’s CS11-747 Neural Networks for NLP in CMU2019

Content

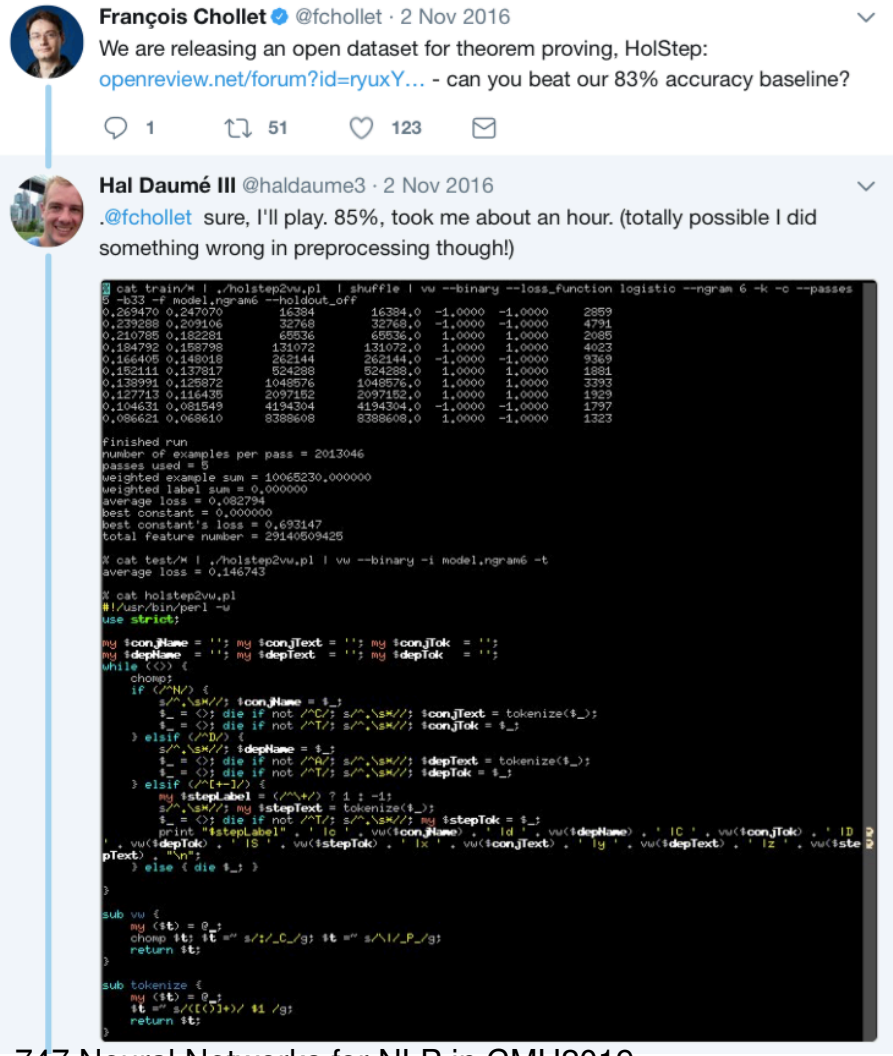
Bag of n-grams



A slide from Graham Neubig's CS11-747 Neural Networks for NLP in CMU2019

Why Bag of n-grams?

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



A slide from Graham Neubig’s CS11-747 Neural Networks for NLP in CMU2019

What Problems w/ Bag of n-grams?

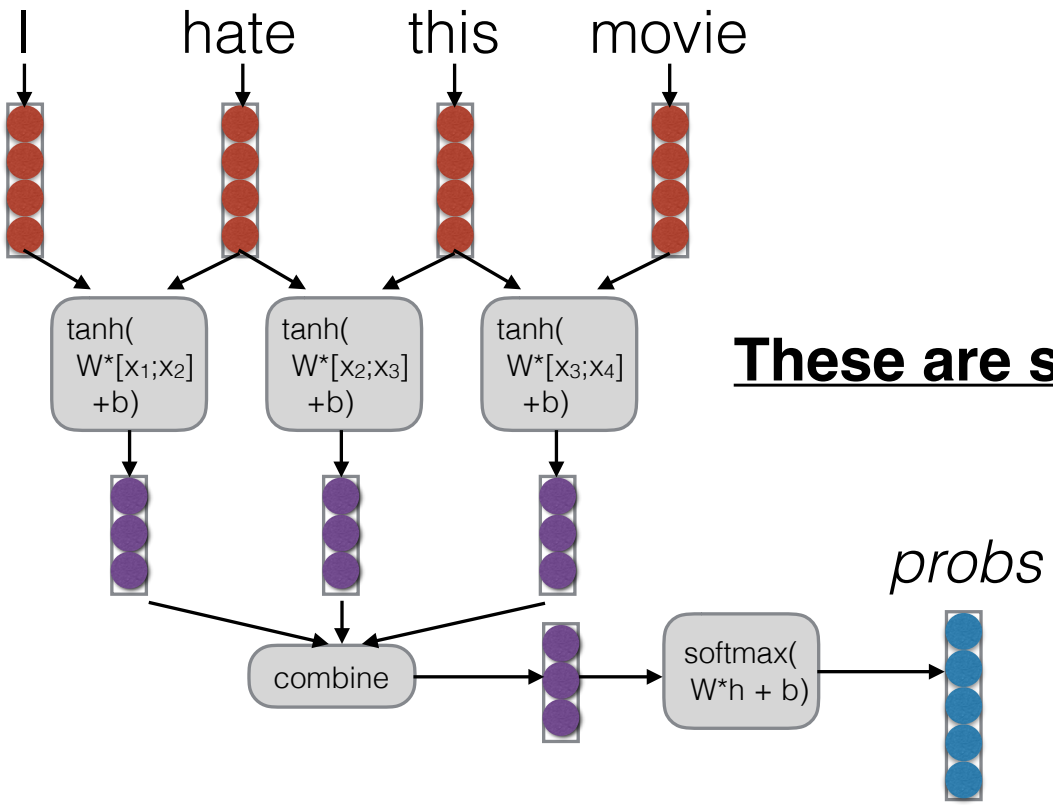
- Same as before: parameter explosion
- No sharing between similar words/n-grams

A slide from Graham Neubig's CS11-747 Neural Networks for NLP in CMU2019

Content

1-dimensional Convolutions / Time-delay Networks

(Waibel et al. 1989)

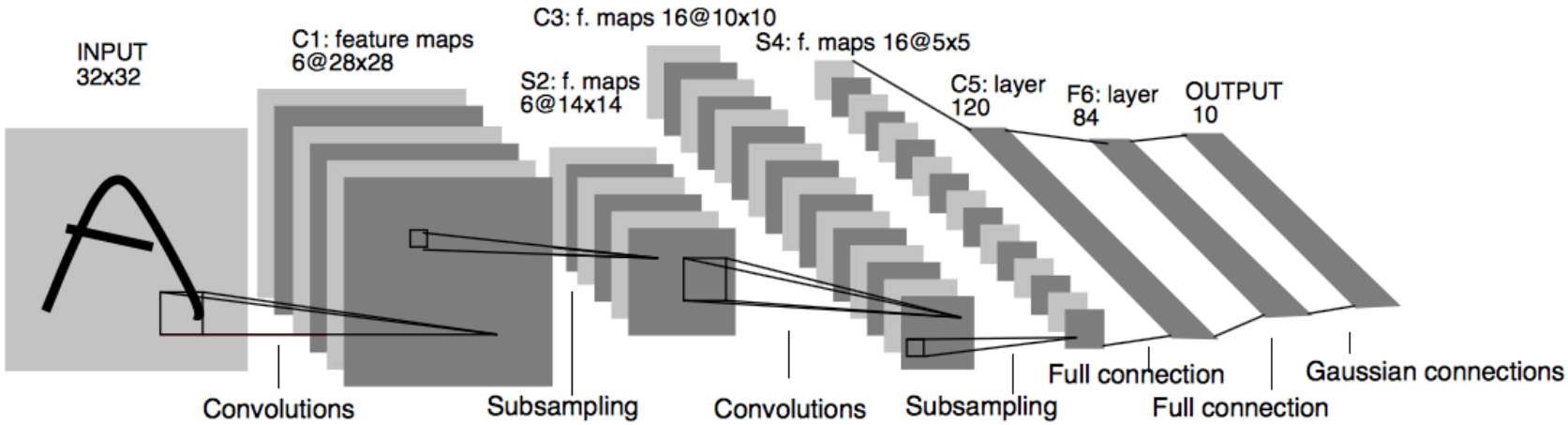


These are soft 2-grams!

A slide from Graham Neubig's CS11-747 Neural Networks for NLP in CMU2019

2-dimensional Convolutional Networks

(LeCun et al. 1997)



Parameter extraction performs a 2D sweep, not 1D

A slide from Graham Neubig's CS11-747 Neural Networks for NLP in CMU2019

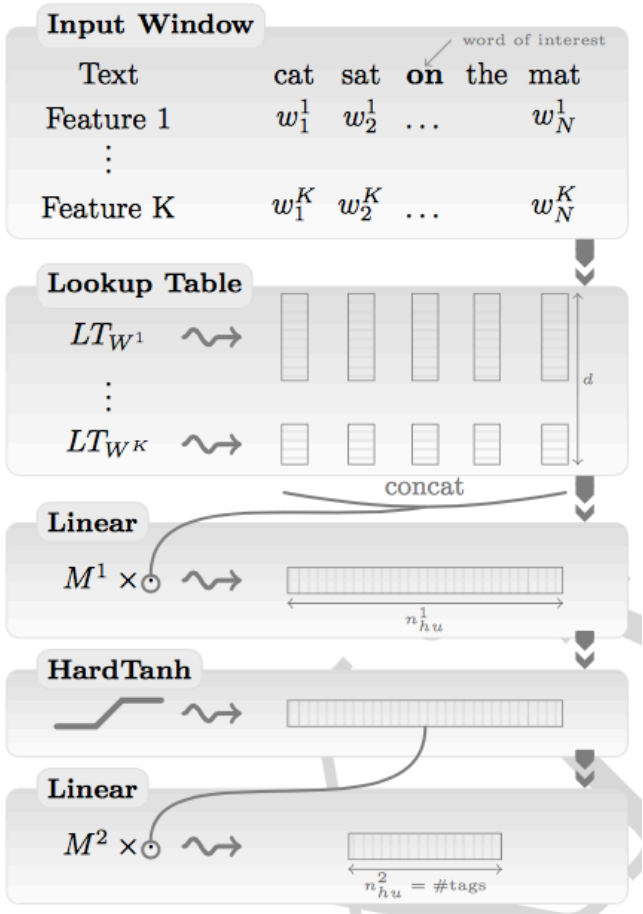
CNNs for Text
(Collobert and Weston 2011)

- 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

A slide from Graham Neubig's CS11-747 Neural Networks for NLP in CMU2019

CNNs for Tagging

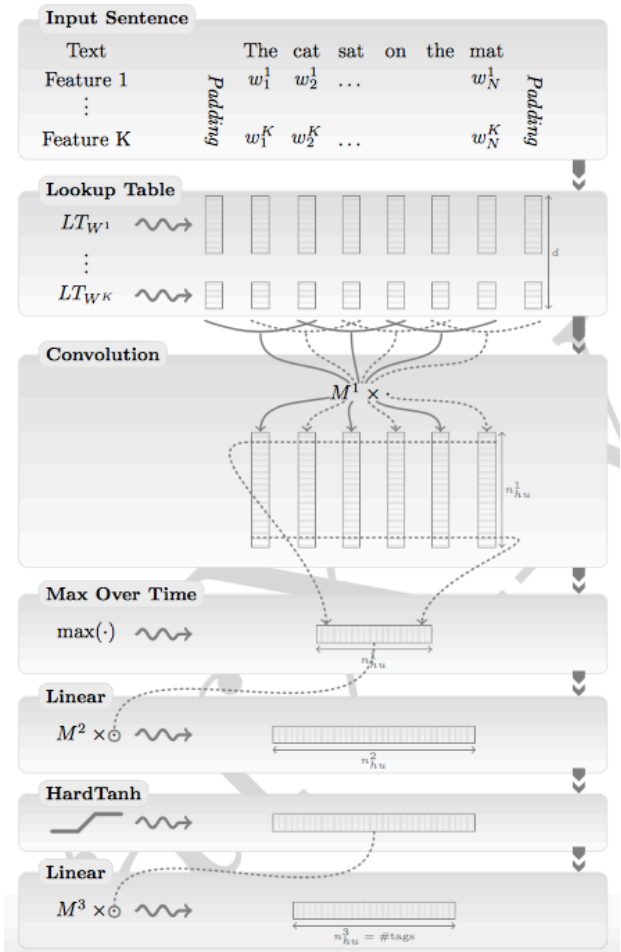
(Collobert and Weston 2011)



A slide from Graham Neubig's CS11-747 Neural Networks for NLP in CMU2019

CNNs for Sentence Modeling

(Collobert and Weston 2011)



A slide from Graham Neubig's CS11-747 Neural Networks for NLP in CMU2019

Standard conv2d Function

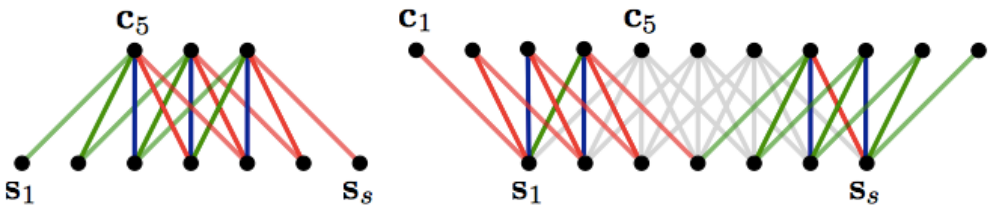
- 2D convolution function takes input + parameters
- **Input:** 3D tensor
  - rows (e.g. words), columns, features ("channels")
- **Parameters/Filters:** 4D tensor
  - rows, columns, input features, output features

A slide from Graham Neubig's CS11-747 Neural Networks for NLP in CMU2019

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"*)

Narrow →



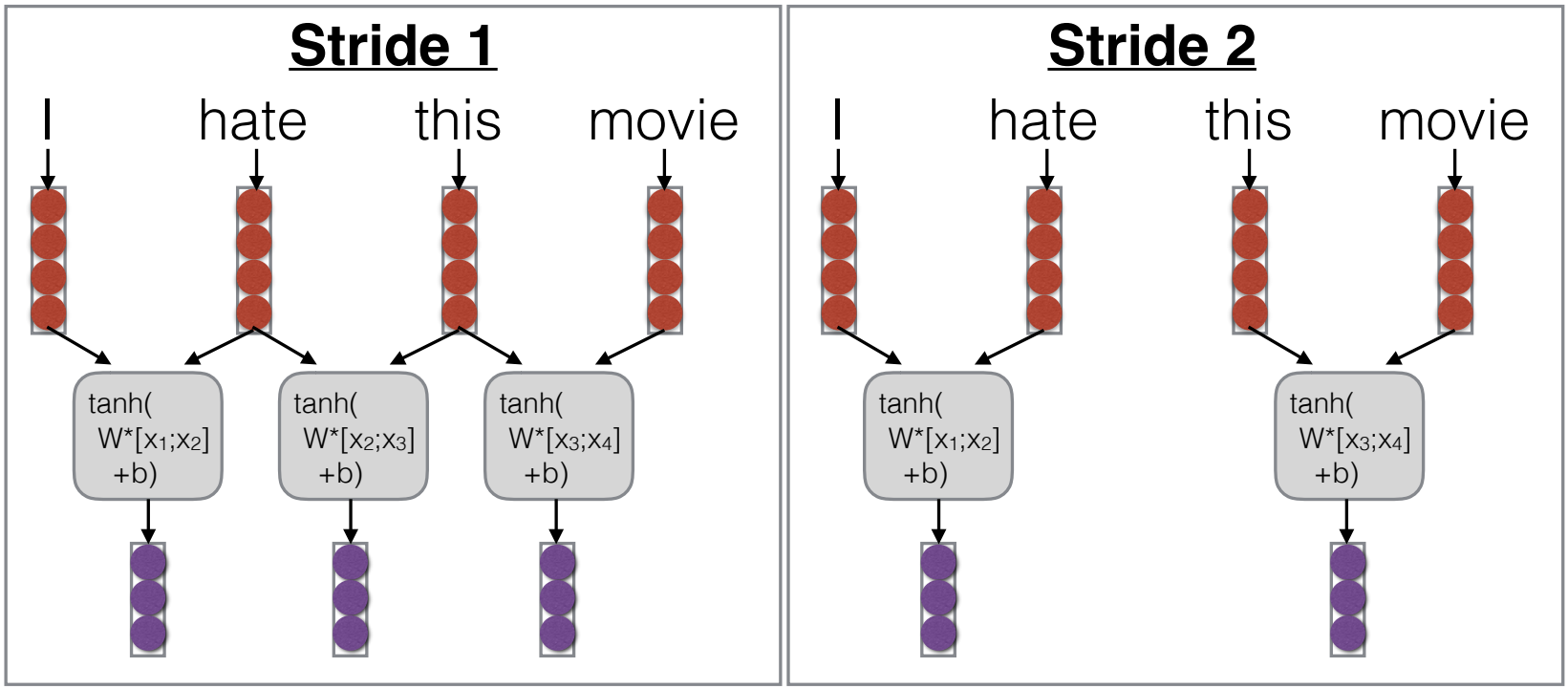
← Wide

Image: Kalchbrenner et al. 2014

A slide from Graham Neubig's CS11-747 Neural Networks for NLP in CMU2019

Striding

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



A slide from Graham Neubig's CS11-747 Neural Networks for NLP in CMU2019

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?"

A slide from Graham Neubig's CS11-747 Neural Networks for NLP in CMU2019

Content

Stacked Convolution

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

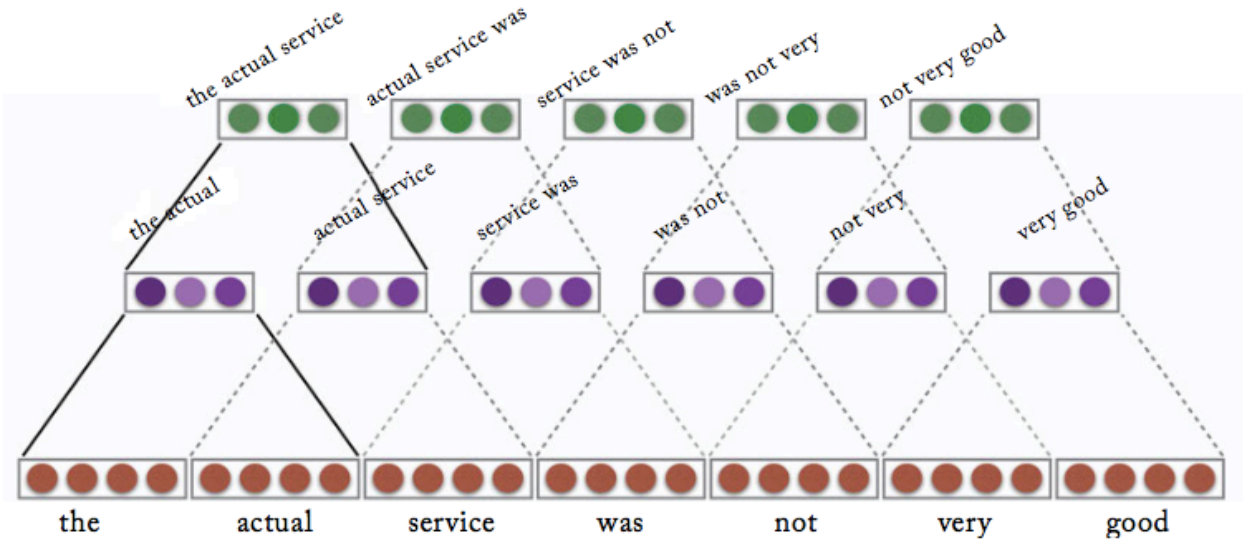


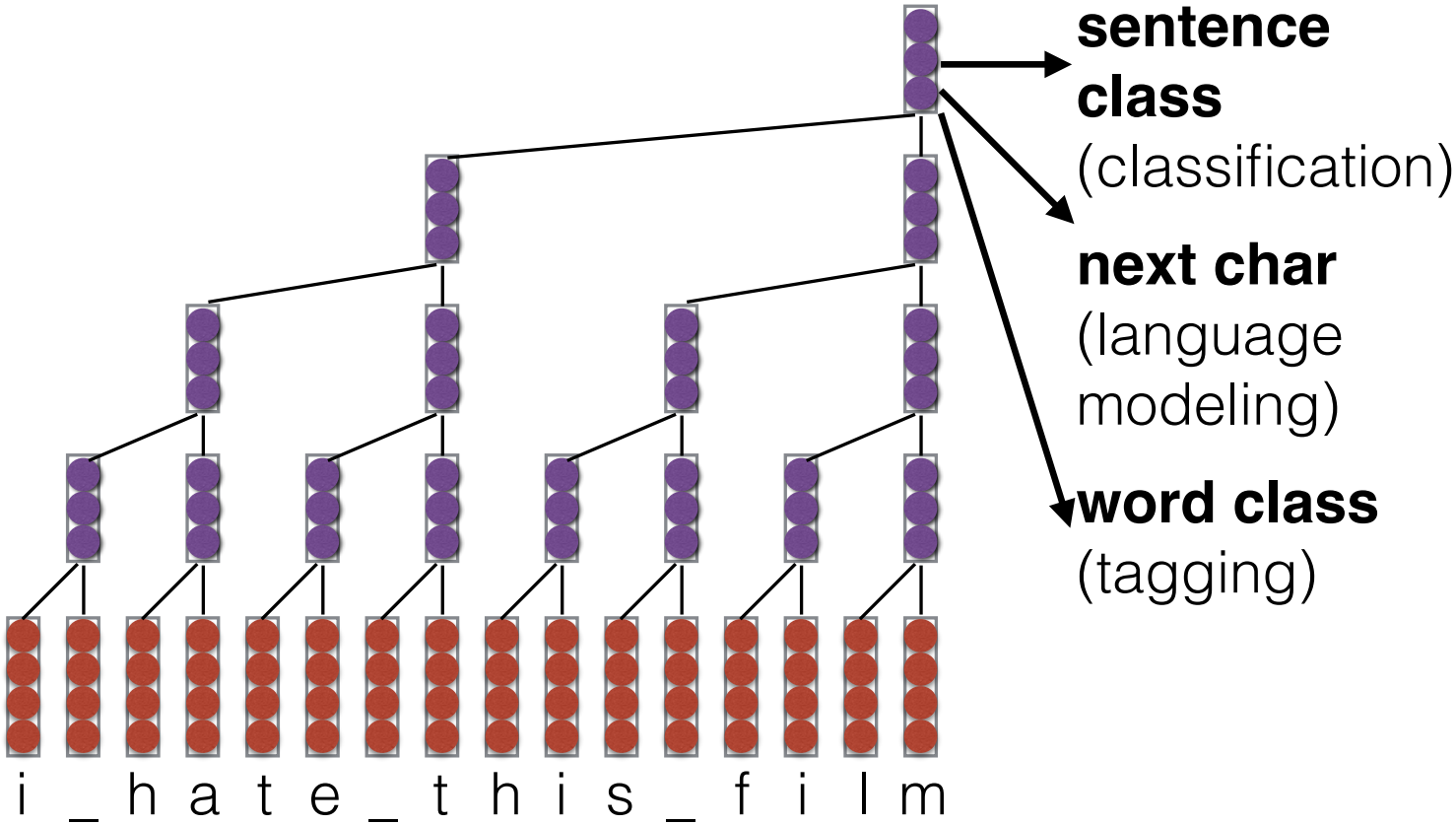
Image Credit: Goldberg Book

A slide from Graham Neubig's CS11-747 Neural Networks for NLP in CMU2019

Dilated Convolution

(e.g. Kalchbrenner et al. 2016)

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



A slide from Graham Neubig's CS11-747 Neural Networks for NLP in CMU2019

Why (Dilated) Convolution for Modeling Sentences?

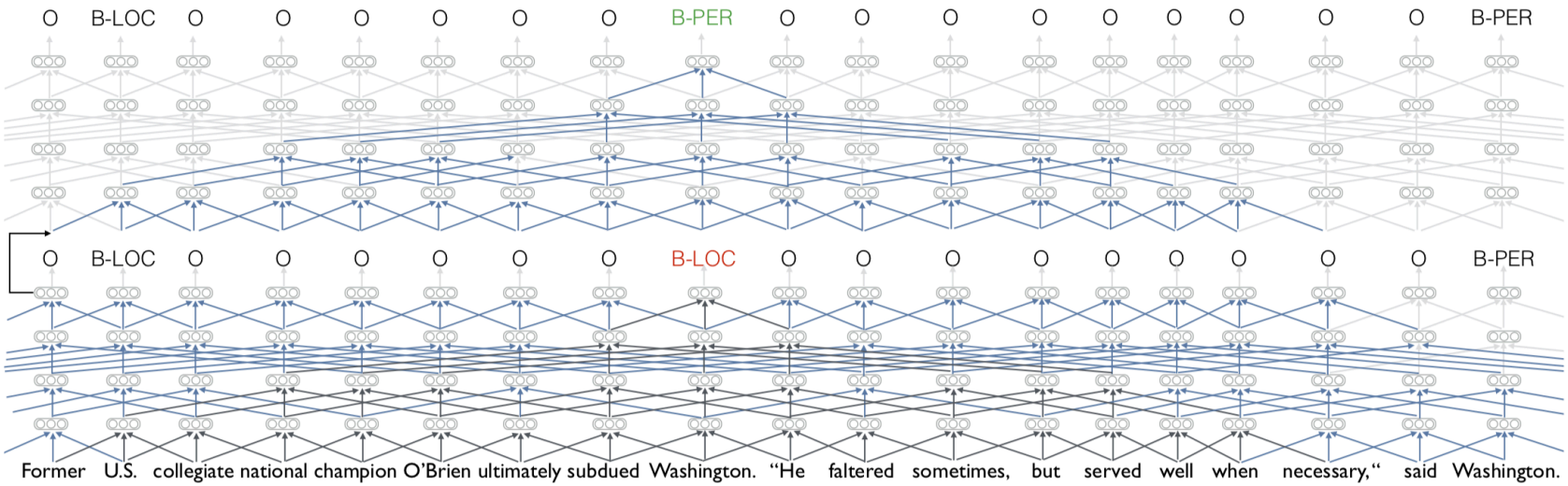
- In contrast to recurrent neural networks (next class)
- **+** Fewer steps from each word to the final representation: RNN O(N), Dilated CNN O(log N)
- **+** Easier to parallelize on GPU
- **-** Slightly less natural for arbitrary-length dependencies
- **-** A bit slower on CPU?

A slide from Graham Neubig's CS11-747 Neural Networks for NLP in CMU2019

Iterated Dilated Convolution

(Strubell+ 2017)

- Multiple iterations of the same stack of dilated convolutions



- Wider context, more parameter efficient

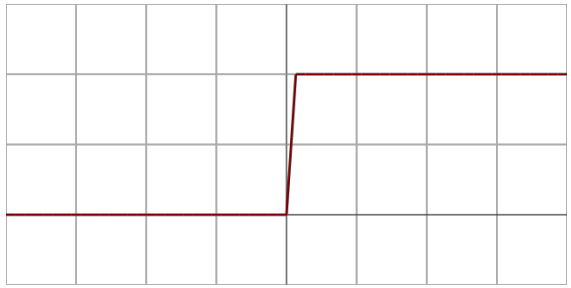
A slide from Graham Neubig's CS11-747 Neural Networks for NLP in CMU2019

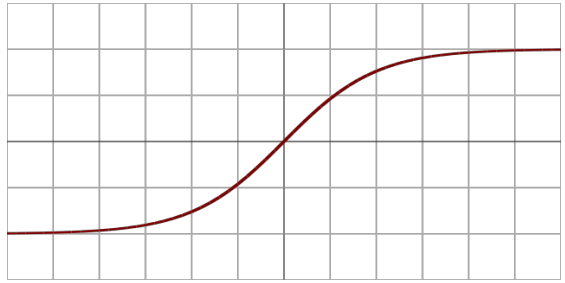
Content

Non-linear Functions

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

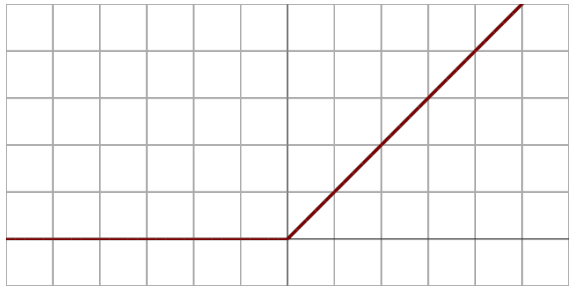
step

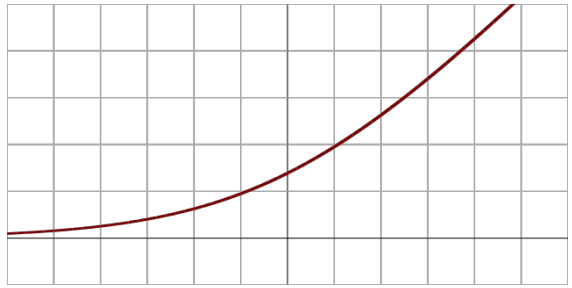




tanh

rectifier (RelU)





soft plus

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

Image: Wikipedia

A slide from Graham Neubig's CS11-747 Neural Networks for NLP in CMU2019

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.

A slide from Graham Neubig's CS11-747 Neural Networks for NLP in CMU2019

Content

A 1D convolution for text

tentative	0.2	0.1	−0.3	0.4
deal	0.5	0.2	−0.3	−0.1
reached	−0.1	−0.3	−0.2	0.4
to	0.3	−0.3	0.1	0.1
keep	0.2	−0.3	0.4	0.2
government	0.1	0.2	−0.1	−0.1
open	−0.4	−0.4	0.2	0.3

t,d,r	−1.0
d,r,t	−0.5
r,t,k	−3.6
t,k,g	−0.2
k,g,o	0.3

Apply a **filter** (or **kernel**) of size 3

3	1	2	−3
−1	2	1	−3
1	1	−1	1

10

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

1D convolution for text with padding

Ø	0.0	0.0	0.0	0.0
tentative	0.2	0.1	−0.3	0.4
deal	0.5	0.2	−0.3	−0.1
reached	−0.1	−0.3	−0.2	0.4
to	0.3	−0.3	0.1	0.1
keep	0.2	−0.3	0.4	0.2
government	0.1	0.2	−0.1	−0.1
open	−0.4	−0.4	0.2	0.3
Ø	0.0	0.0	0.0	0.0

Ø,t,d	−0.6
t,d,r	−1.0
d,r,t	−0.5
r,t,k	−3.6
t,k,g	−0.2
k,g,o	0.3
g,o,Ø	−0.5

Apply a **filter** (or **kernel**) of size 3

3	1	2	−3
−1	2	1	−3
1	1	−1	1

11

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

3 channel 1D convolution with padding = 1

Ø	0.0	0.0	0.0	0.0
tentative	0.2	0.1	−0.3	0.4
deal	0.5	0.2	−0.3	−0.1
reached	−0.1	−0.3	−0.2	0.4
to	0.3	−0.3	0.1	0.1
keep	0.2	−0.3	0.4	0.2
government	0.1	0.2	−0.1	−0.1
open	−0.4	−0.4	0.2	0.3
Ø	0.0	0.0	0.0	0.0

Ø,t,d	−0.6	0.2	1.4
t,d,r	−1.0	1.6	−1.0
d,r,t	−0.5	−0.1	0.8
r,t,k	−3.6	0.3	0.3
t,k,g	−0.2	0.1	1.2
k,g,o	0.3	0.6	0.9
g,o,Ø	−0.5	−0.9	0.1

Apply 3 **filters** of size 3

3	1	2	−3
−1	2	1	−3
1	1	−1	1

1	0	0	1
1	0	−1	−1
0	1	0	1

1	−1	2	−1
1	0	−1	3
0	2	2	1

Could also use (zero) padding = 2

Also called “wide convolution”

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

conv1d, padded with max pooling over time

Ø	0.0	0.0	0.0	0.0
tentative	0.2	0.1	−0.3	0.4
deal	0.5	0.2	−0.3	−0.1
reached	−0.1	−0.3	−0.2	0.4
to	0.3	−0.3	0.1	0.1
keep	0.2	−0.3	0.4	0.2
government	0.1	0.2	−0.1	−0.1
open	−0.4	−0.4	0.2	0.3
Ø	0.0	0.0	0.0	0.0

Ø,t,d	−0.6	0.2	1.4
t,d,r	−1.0	1.6	−1.0
d,r,t	−0.5	−0.1	0.8
r,t,k	−3.6	0.3	0.3
t,k,g	−0.2	0.1	1.2
k,g,o	0.3	0.6	0.9
g,o,Ø	−0.5	−0.9	0.1
max p	0.3	1.6	1.4

Apply 3 **filters** of size 3

3	1	2	−3
−1	2	1	−3
1	1	−1	1

1	0	0	1
1	0	−1	−1
0	1	0	1

1	−1	2	−1
1	0	−1	3
0	2	2	1

13

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

conv1d, padded with ave pooling over time

Ø	0.0	0.0	0.0	0.0
tentative	0.2	0.1	−0.3	0.4
deal	0.5	0.2	−0.3	−0.1
reached	−0.1	−0.3	−0.2	0.4
to	0.3	−0.3	0.1	0.1
keep	0.2	−0.3	0.4	0.2
government	0.1	0.2	−0.1	−0.1
open	−0.4	−0.4	0.2	0.3
Ø	0.0	0.0	0.0	0.0

Ø,t,d	−0.6	0.2	1.4
t,d,r	−1.0	1.6	−1.0
d,r,t	−0.5	−0.1	0.8
r,t,k	−3.6	0.3	0.3
t,k,g	−0.2	0.1	1.2
k,g,o	0.3	0.6	0.9
g,o,Ø	−0.5	−0.9	0.1
ave p	−0.87	0.26	0.53

Apply 3 **filters** of size 3

	3	1	2	−3
	−1	2	1	−3
14	1	1	−1	1

1	0	0	1
1	0	−1	−1
0	1	0	1

1	−1	2	−1
1	0	−1	3
0	2	2	1

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

Other less useful notions: stride = 2

Ø	0.0	0.0	0.0	0.0
tentative	0.2	0.1	−0.3	0.4
deal	0.5	0.2	−0.3	−0.1
reached	−0.1	−0.3	−0.2	0.4
to	0.3	−0.3	0.1	0.1
keep	0.2	−0.3	0.4	0.2
government	0.1	0.2	−0.1	−0.1
open	−0.4	−0.4	0.2	0.3
Ø	0.0	0.0	0.0	0.0

Ø,t,d	−0.6	0.2	1.4
d,r,t	−0.5	−0.1	0.8
t,k,g	−0.2	0.1	1.2
g,o,Ø	−0.5	−0.9	0.1

Apply 3 **filters** of size 3

	3	1	2	−3
	−1	2	1	−3
16	1	1	−1	1

1	0	0	1
1	0	−1	−1
0	1	0	1

1	−1	2	−1
1	0	−1	3
0	2	2	1

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

Less useful: local max pool, stride = 2

Ø	0.0	0.0	0.0	0.0
tentative	0.2	0.1	−0.3	0.4
deal	0.5	0.2	−0.3	−0.1
reached	−0.1	−0.3	−0.2	0.4
to	0.3	−0.3	0.1	0.1
keep	0.2	−0.3	0.4	0.2
government	0.1	0.2	−0.1	−0.1
open	−0.4	−0.4	0.2	0.3
Ø	0.0	0.0	0.0	0.0

Apply 3 **filters** of size 3

3	1	2	−3
−1	2	1	−3
1	1	−1	1

1	0	0	1
1	0	−1	−1
0	1	0	1

1	−1	2	−1
1	0	−1	3
0	2	2	1

Ø,t,d	−0.6	0.2	1.4
t,d,r	−1.0	1.6	−1.0
d,r,t	−0.5	−0.1	0.8
r,t,k	−3.6	0.3	0.3
t,k,g	−0.2	0.1	1.2
k,g,o	0.3	0.6	0.9
g,o,Ø	−0.5	−0.9	0.1
Ø	−Inf	−Inf	−Inf

Ø,t,d,r	−0.6	1.6	1.4
d,r,t,k	−0.5	0.3	0.8
t,k,g,o	0.3	0.6	1.2
g,o,Ø,Ø	−0.5	−0.9	0.1

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

conv1d, *k*-max pooling over time, *k* = 2

Ø	**0.0**	**0.0**	**0.0**	**0.0**
tentative	0.2	0.1	−0.3	0.4
deal	0.5	0.2	−0.3	−0.1
reached	−0.1	−0.3	−0.2	0.4
to	0.3	−0.3	0.1	0.1
keep	0.2	−0.3	0.4	0.2
government	0.1	0.2	−0.1	−0.1
open	−0.4	−0.4	0.2	0.3
Ø	**0.0**	**0.0**	**0.0**	**0.0**

Ø,t,d	−0.6	0.2	1.4
t,d,r	−1.0	1.6	−1.0
d,r,t	−0.5	−0.1	0.8
r,t,k	−3.6	0.3	0.3
t,k,g	−0.2	0.1	1.2
k,g,o	0.3	0.6	0.9
g,o,Ø	−0.5	−0.9	0.1

2-max p	−0.2	1.6	1.4
	0.3	0.6	1.2

Apply 3 **filters** of size 3

3	1	2	−3
−1	2	1	−3
1	1	−1	1

1	0	0	1
1	0	−1	−1
0	1	0	1

1	−1	2	−1
1	0	−1	3
0	2	2	1

18

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

Other somewhat useful notions: dilation = 2

Ø	0.0	0.0	0.0	0.0
tentative	0.2	0.1	−0.3	0.4
deal	0.5	0.2	−0.3	−0.1
reached	−0.1	−0.3	−0.2	0.4
to	0.3	−0.3	0.1	0.1
keep	0.2	−0.3	0.4	0.2
government	0.1	0.2	−0.1	−0.1
open	−0.4	−0.4	0.2	0.3
Ø	0.0	0.0	0.0	0.0

Apply 3 **filters** of size 3

3	1	2	−3
−1	2	1	−3
1	1	−1	1

1	0	0	1
1	0	−1	−1
0	1	0	1

1	−1	2	−1
1	0	−1	3
0	2	2	1

Ø,t,d	−0.6	0.2	1.4
t,d,r	−1.0	1.6	−1.0
d,r,t	−0.5	−0.1	0.8
r,t,k	−3.6	0.3	0.3
t,k,g	−0.2	0.1	1.2
k,g,o	0.3	0.6	0.9
g,o,Ø	−0.5	−0.9	0.1

1,3,5	0.3	0.0
2,4,6		
3,5,7		

2	3	1
1	−1	−1
3	1	0

1	3	1
1	−1	−1
3	1	−1

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

Content

3. Single Layer CNN for Sentence Classification

- Yoon Kim (2014): Convolutional Neural Networks for Sentence Classification. EMNLP 2014. https://arxiv.org/pdf/1408.5882.pdf Code: https://arxiv.org/pdf/1408.5882.pdf [Theano!, etc.]
- A variant of convolutional NNs of Collobert, Weston et al. (2011)

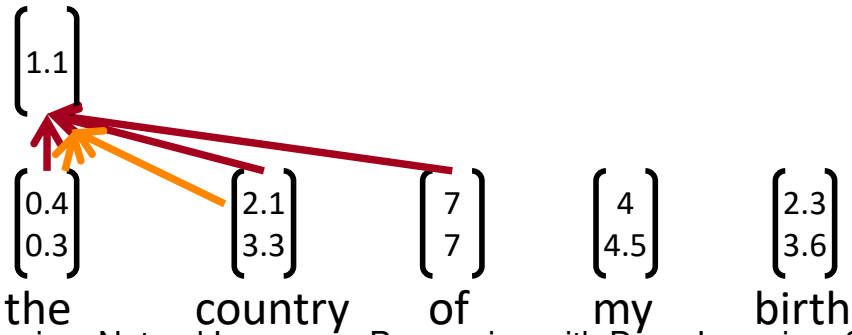
- Goal: Sentence classification:
  - Mainly positive or negative sentiment of a sentence
  - Other tasks like:
    - Subjective or objective language sentence
    - Question classification: about person, location, number, …

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

20

Single Layer CNN for Sentence Classification

- A simple use of one convolutional layer and **pooling**
- Word vectors: $\mathbf{x}_i \in \mathbb{R}^k$
- Sentence: $\mathbf{x}_{1:n} = \mathbf{x}_1 \oplus x_2 \oplus \cdots \oplus \mathbf{x}_n$ (vectors concatenated)
- Concatenation of words in range: $\mathbf{x}_{i:i+j}$ (symmetric more common)
- Convolutional filter: $\mathbf{w} \in \mathbb{R}^{hk}$ (over window of *h* words)
- Note, filter is a vector!
- Filter could be of size 2, 3, or 4:



Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

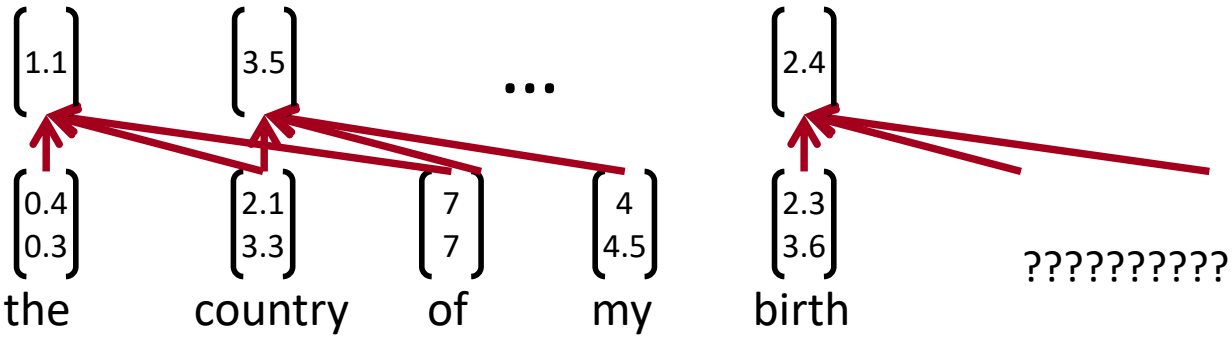
21

Single layer CNN

- Filter **w** is applied to all possible windows (concatenated vectors)
- To compute feature (one *channel*) for CNN layer:

$$c_i = f\left(\mathbf{w}^T \mathbf{x}_{i:i+h-1} + b\right)$$

- Sentence: $\mathbf{x}_{1:n} = \mathbf{x}_1 \oplus \mathbf{x}_2 \oplus \ldots \oplus \mathbf{x}_n$
- All possible windows of length *h*: $\{\mathbf{x}_{1:h}, \mathbf{x}_{2:h+1}, \ldots, \mathbf{x}_{n-h+1:n}\}$
- Result is a feature map: $\mathbf{c} = [c_1, c_2, \ldots, c_{n-h+1}] \in \mathbb{R}^{n-h+1}$



Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

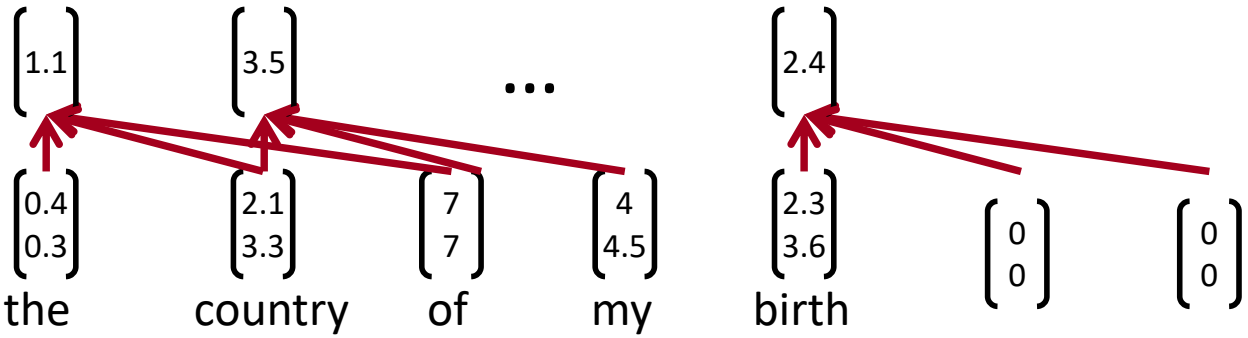
22

Single layer CNN

- Filter **w** is applied to all possible windows (concatenated vectors)
- To compute feature (one *channel*) for CNN layer:

$$c_i = f\left(\mathbf{w}^T \mathbf{x}_{i:i+h-1} + b\right)$$

- Sentence: $\mathbf{x}_{1:n} = \mathbf{x}_1 \oplus \mathbf{x}_2 \oplus \ldots \oplus \mathbf{x}_n$
- All possible windows of length *h*: $\{\mathbf{x}_{1:h}, \mathbf{x}_{2:h+1}, \ldots, \mathbf{x}_{n-h+1:n}\}$
- Result is a feature map: $\mathbf{c} = [c_1, c_2, \ldots, c_{n-h+1}] \in \mathbb{R}^{n-h+1}$



Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

Pooling and channels

- Pooling: max-over-time pooling layer
- Idea: capture most important activation (maximum over time)
- From feature map $\mathbf{c} = [c_1, c_2, \ldots, c_{n-h+1}] \in \mathbb{R}^{n-h+1}$
- Pooled single number: $\hat{c} = \max\{\mathbf{c}\}$

- Use multiple filter weights **w**
- Useful to have different window sizes *h*
- Because of max pooling $\hat{c} = \max\{\mathbf{c}\}$, length of **c** irrelevant

$$\mathbf{c} = [c_1, c_2, \ldots, c_{n-h+1}] \in \mathbb{R}^{n-h+1}$$

- So we could have some filters that look at unigrams, bigrams, tri-grams, 4-grams, etc.

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

24

Multi-channel input idea

- Initialize with pre-trained word vectors (word2vec or Glove)
- Start with two copies
- Backprop into only one set, keep other “static”
- Both channel sets are added to c_i before max-pooling

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

25

Classification after one CNN layer

- First one convolution, followed by one max-pooling
- To obtain final feature vector: $\mathbf{z} = [\hat{c}_1, \ldots, \hat{c}_m]$ (assuming *m* filters **w**)
  - Used 100 feature maps each of sizes 3, 4, 5
- Simple final softmax layer $y = softmax\left(W^{(S)}z + b\right)$

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

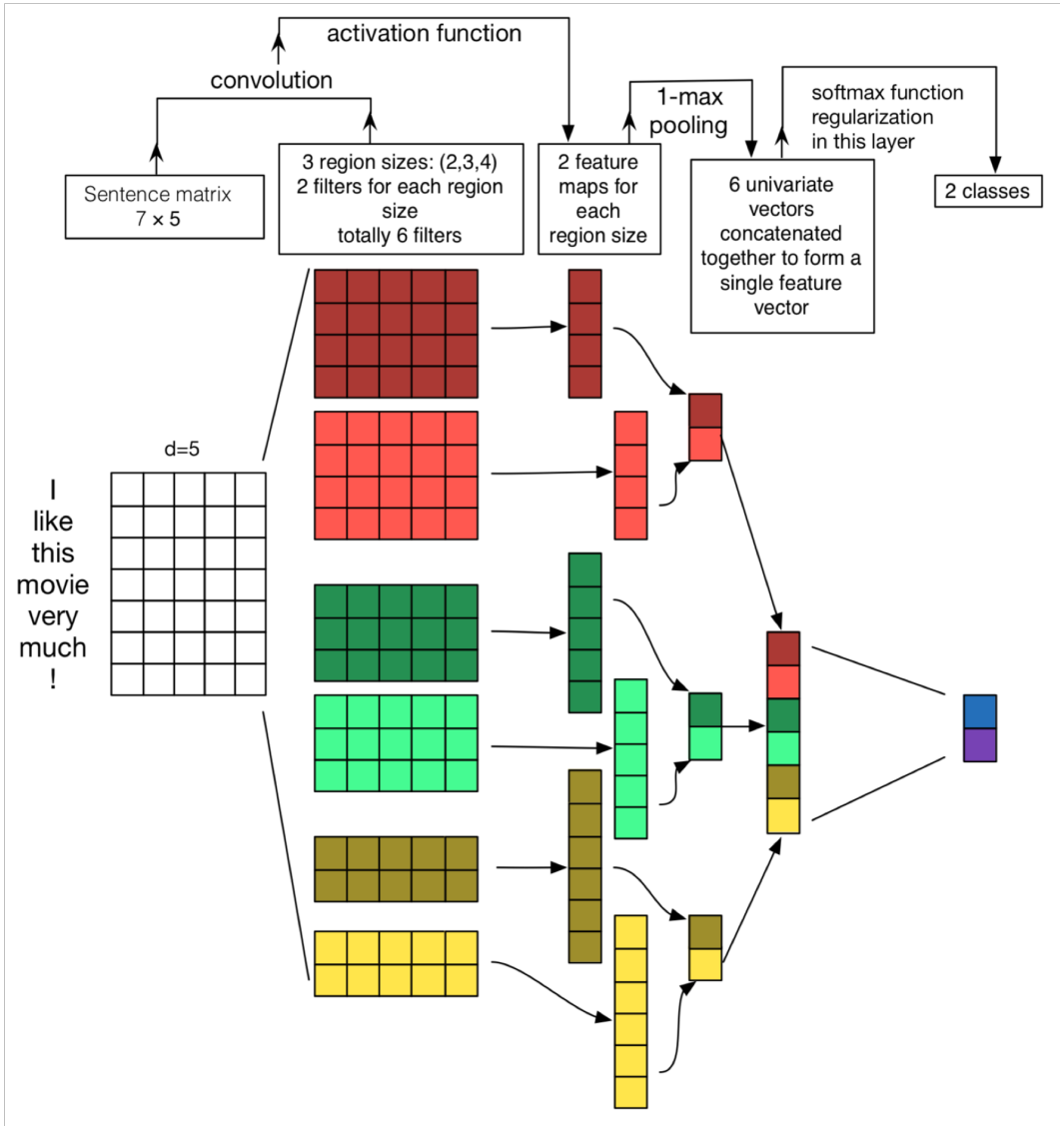
26

From:

Zhang and Wallace (2015) A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification

https://arxiv.org/pdf/1510.03820.pdf

(follow on paper, not famous, but a nice picture)



Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

Regularization

- Use **Dropout**: Create masking vector *r* of Bernoulli random variables with probability *p* (a hyperparameter) of being 1
- Delete features during training:

$$y = softmax\left(W^{(S)}(r \circ z) + b\right)$$

- Reasoning: Prevents co-adaptation (overfitting to seeing specific feature constellations) (Srivastava, Hinton, et al. 2014)
- At test time, no dropout, scale final vector by probability *p*

$$\hat{W}^{(S)} = pW^{(S)}$$

- **Also:** Constrain l_2 norms of weight vectors of each class (row in softmax weight $W^{(S)}$) to fixed number *s* (also a hyperparameter)
- If $\|W_{c\cdot}^{(S)}\| > s$, then rescale it so that: $\|W_{c\cdot}^{(S)}\| = s$
- Not very common

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

28

All hyperparameters in Kim (2014)

- Find hyperparameters based on dev set
- Nonlinearity: ReLU
- Window filter sizes h = 3, 4, 5
- Each filter size has 100 feature maps
- Dropout p = 0.5
  - Kim (2014) reports **2–4%** accuracy improvement from dropout
- L2 constraint *s* for rows of softmax, *s* = 3
- Mini batch size for SGD training: 50
- Word vectors: pre-trained with word2vec, *k* = 300

- During training, keep checking performance on dev set and pick highest accuracy weights for final evaluation

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

29

Experiments

Model	MR	SST-1	SST-2	Subj	TREC	CR	MPQA
CNN-rand	76.1	45.0	82.7	89.6	91.2	79.8	83.4
CNN-static	81.0	45.5	86.8	93.0	92.8	84.7	**89.6**
CNN-non-static	**81.5**	48.0	87.2	93.4	93.6	84.3	89.5
CNN-multichannel	81.1	47.4	**88.1**	93.2	92.2	**85.0**	89.4
RAE (Socher et al., 2011)	77.7	43.2	82.4	–	–	–	86.4
MV-RNN (Socher et al., 2012)	79.0	44.4	82.9	–	–	–	–
RNTN (Socher et al., 2013)	–	45.7	85.4	–	–	–	–
DCNN (Kalchbrenner et al., 2014)	–	48.5	86.8	–	93.0	–	–
Paragraph-Vec (Le and Mikolov, 2014)	–	**48.7**	87.8	–	–	–	–
CCAE (Hermann and Blunsom, 2013)	77.8	–	–	–	–	–	87.2
Sent-Parser (Dong et al., 2014)	79.5	–	–	–	–	–	86.3
NBSVM (Wang and Manning, 2012)	79.4	–	–	93.2	–	81.8	86.3
MNB (Wang and Manning, 2012)	79.0	–	–	**93.6**	–	80.0	86.3
G-Dropout (Wang and Manning, 2013)	79.0	–	–	93.4	–	82.1	86.1
F-Dropout (Wang and Manning, 2013)	79.1	–	–	**93.6**	–	81.9	86.3
Tree-CRF (Nakagawa et al., 2010)	77.3	–	–	–	–	81.4	86.1
CRF-PR (Yang and Cardie, 2014)	–	–	–	–	–	82.7	–
SVM_S (Silva et al., 2011)	–	–	–	–	**95.0**	–	–

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

30

Content

1. Artificial neural networks (ANNs)
2. Multilayer perceptron (MLP)
3. Convolutional neural networks (CNNs)
4. Convolutional Networks for Text Classification