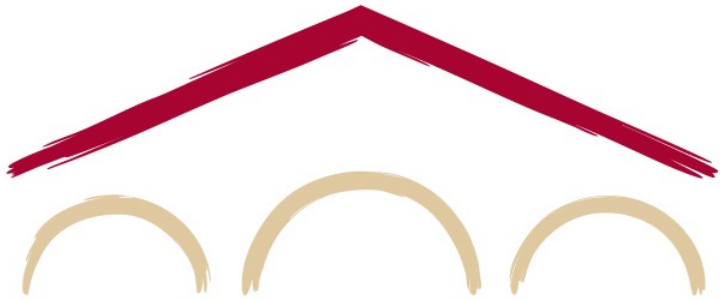
Natural Language Processing with Deep Learning CS224N/Ling284



Christopher Manning

Lecture 16: ConvNets for NLP and Tree Recursive Neural Networks

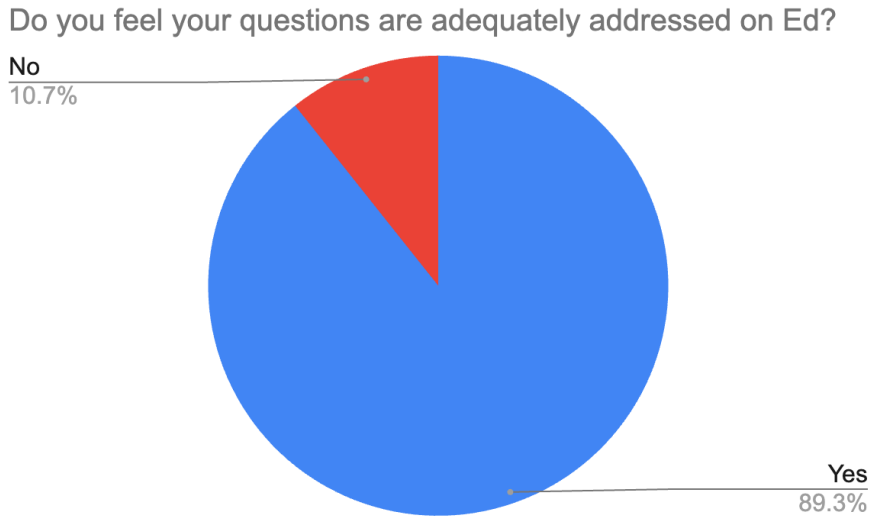
Lecture Plan

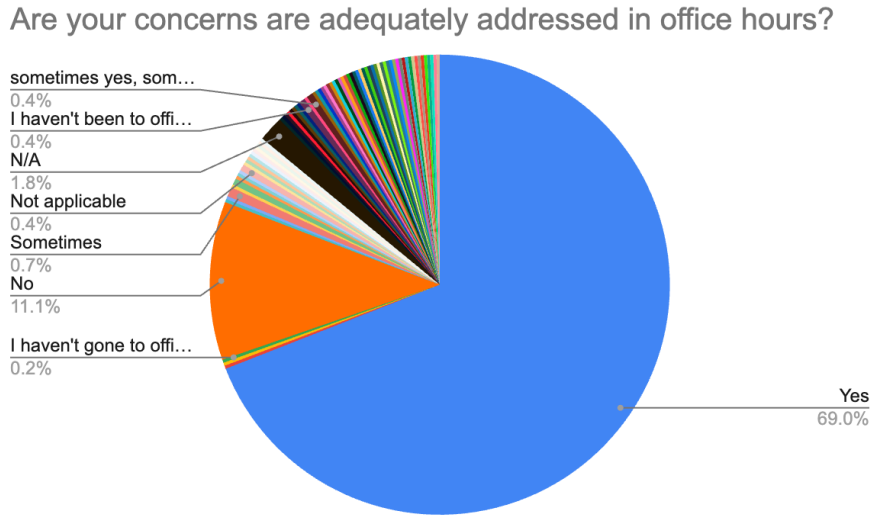
1. Course organization updates (5 mins)
2. Intro to CNNs (25 mins)
3. Simple CNN for Sentence Classification: Yoon (2014) (10 mins)
4. CNN potpourri (5 mins)
5. Deep CNN for Sentence Classification: Conneau et al. (2017) (10 mins)
6. Tree Recursive Neural Nets, briefly (15 mins)
7. Recursive Neural Tensor Networks and Sentiment Analysis (15 mins)

1. Course Organization Updates

Mid-quarter survey

- Fantastic lectures and really interesting content
- I enjoy the lectures a lot. Also I like the coding part of the problem sets.
- I love how the lectures focus on theory and assignments on implementation - I feel learning both is hugely beneficial.
-
- Maybe an exam to help reinforce concepts from class
- There could be more timely responses from course staff
- I think that some of the more math-heavy lectures are better explained with a whiteboard/pen and paper. The PowerPoints are well made but it can sometimes be hard to understand the individual steps in a process.
- I would love to see more content about more recent models like state space models





Course Organization Updates

Final Project: The key remaining thing to do

- Final Project Milestone was due yesterday, Wed May 22!
- Make an effort to get feedback in person from your mentor as well in office hours!
- **Final project poster session: Mon Jun 10, 11:00am–3:00pm: You need to be there***
  - Alumni Center, McCaw Hall and Ford Gardens
  - Groundbreaking research! Prizes! **Food!**

Invited speakers

- We had Nathan Lambert in the previous lecture
- Next Tuesday, May 28 is Adina Williams on Safety
- **Attendance is expected for on-campus students**; otherwise: “reaction paragraph”

Course Organization Updates

GPUs: Cloud Compute for projects

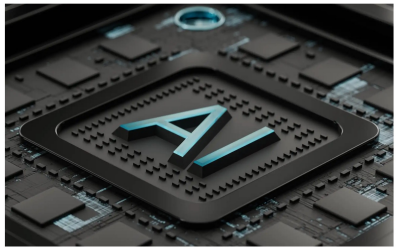
- You're welcome to use **Google Colab**, but it provides limited, inconsistent GPU access
  - We recommend paying $10/month for Colab Pro, which gives better GPU access
  - We can't reimburse you for that.
- We encourage you to use the **GCP** credits we got for the class and/or API access through **Together AI**, if appropriate for your project
- You're also welcome to try Kaggle Notebooks
  - A vanilla Jupyter notebook, not as fancy as Colab, but better GPU access
- Some groups have done well using Modal
  - Some free hours, then need to pay

AI chip shortages continue, but there may be an end in sight

News Analysis

May 07, 2024 • 7 mins

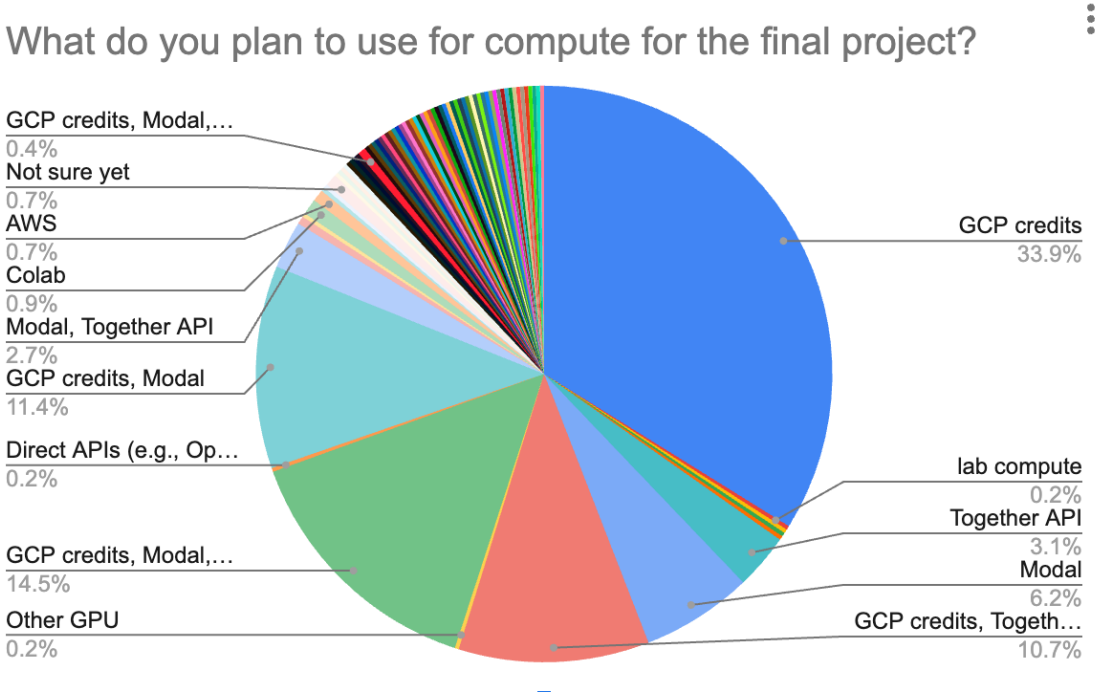
CPUs and Processors | Generative AI | Technology Industry



CREDIT: SHUTTERSTOCK/JAVIER PARDINA

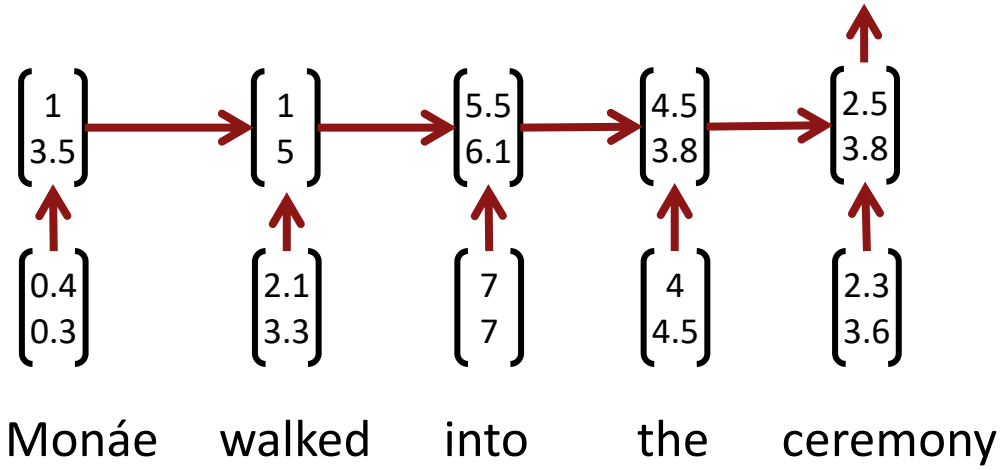
While GPUs are in high demand, they still need high-performance memory chips for AI apps. The market is tight for both — for now.

As the adoption of generative artificial intelligence (genAI) continues to soar, the infrastructure to support that growth is currently running into a supply and demand bottleneck.



2. From RNNs to Convolutional Neural Nets

- Recurrent neural nets cannot capture phrases without prefix context
- Often capture too much of last words in final vector



- E.g., softmax for word prediction is usually calculated based on the last step

From RNNs to Convolutional Neural Nets

- Main Convolutional Neural Net (CNN/ConvNet) idea:
  - What if we compute vectors for every possible word subsequence of a certain length?

- Example: “tentative deal reached to keep government open” computes vectors for:
  - tentative deal reached, deal reached to, reached to keep, to keep government, keep government open

- Regardless of whether subsequence is grammatical or a natural linguistic constituent
  - Not very linguistically or cognitively plausible

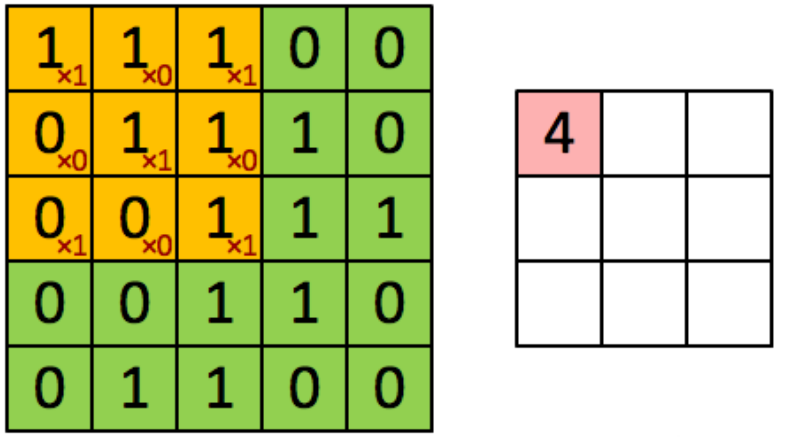
- Then group them afterwards (more soon)

What is a convolution anyway?

- 1d discrete convolution generally: $(f * g)[n] = \sum_{m=-M}^{M} f[n-m]g[m].$

- Convolution is classically used to extract features from images
  - Models position-invariant identification
  - Longer version in cs231n!

- 2d example →
- Yellow color and red numbers show filter (=kernel) weights
- Green shows input
- Pink shows output



From Stanford UFLDL wiki

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

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

3	1	2	−3
−1	2	1	−3
1	1	−1	1

t,d,r	−1.0	0.0	0.50
d,r,t	−0.5	0.5	0.38
r,t,k	−3.6	-2.6	0.93
t,k,g	−0.2	0.8	0.31
k,g,o	0.3	1.3	0.21

+ bias

→ non-linearity

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

Could also use (zero) padding = 2

Also called “wide convolution”

3 channel 1D convolution with padding = 1 and 3 filters

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

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

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
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

In PyTorch

```
batch_size = 16
word_embed_size = 4
seq_len = 7
input = torch.randn(batch_size, word_embed_size, seq_len)
conv1 = Conv1d(in_channels=word_embed_size, out_channels=3,
                        kernel_size=3)  # can add: padding=1
hidden1 = conv1(input)
hidden2 = torch.max(hidden1, dim=2)  # max pool
```

Other (maybe 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
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

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

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	0.3	1.6	1.4
	−0.2	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

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

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		

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

2	3	1
1	−1	−1
3	1	0

1	3	1
1	−1	−1
3	1	−1

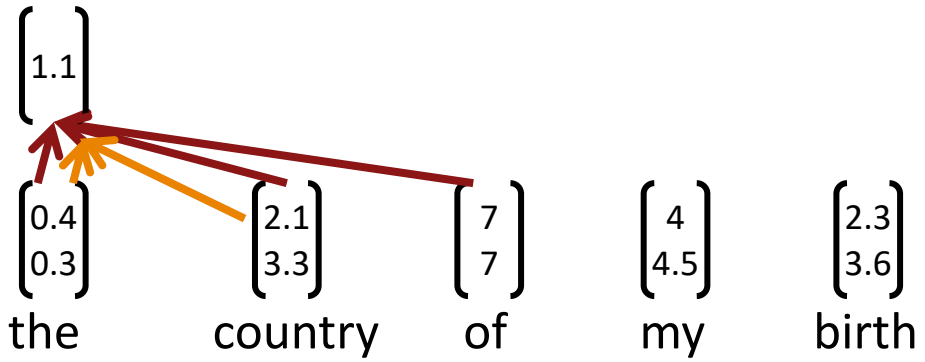
3. Single Layer CNN for Sentence Classification

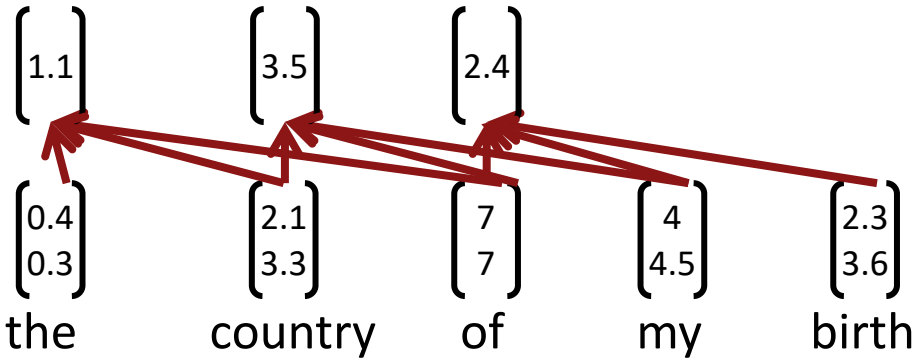
- Yoon Kim (2014): Convolutional Neural Networks for Sentence Classification. EMNLP 2014. https://arxiv.org/pdf/1408.5882.pdf

- 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, …

Single Layer CNN for Sentence Classification

- A simple use of one convolutional layer and **max 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 are concatenated)
- Filter applied to concatenation of words in range: $\mathbf{x}_{i:i+j}$ (symmetric more common)
- Convolutional filter $\mathbf{w} \in \mathbb{R}^{hk}$ applied to all possible windows $\{\mathbf{x}_{1:h}, \mathbf{x}_{2:h+1}, \ldots, \mathbf{x}_{n-h+1:n}\}$
  - Filter is done as a long vector over window of *h* words
  - Filter could be of size *h* = 2, 3, or 4 words
- To compute feature (one *channel*) for CNN layer: $c_i = f(\mathbf{w}^T \mathbf{x}_{i:i+h-1} + b)$
- Result is a feature map: $\mathbf{c} = [c_1, c_2, \ldots, c_{n-h+1}] \in \mathbb{R}^{n-h+1}$





Pooling, channels, and classification

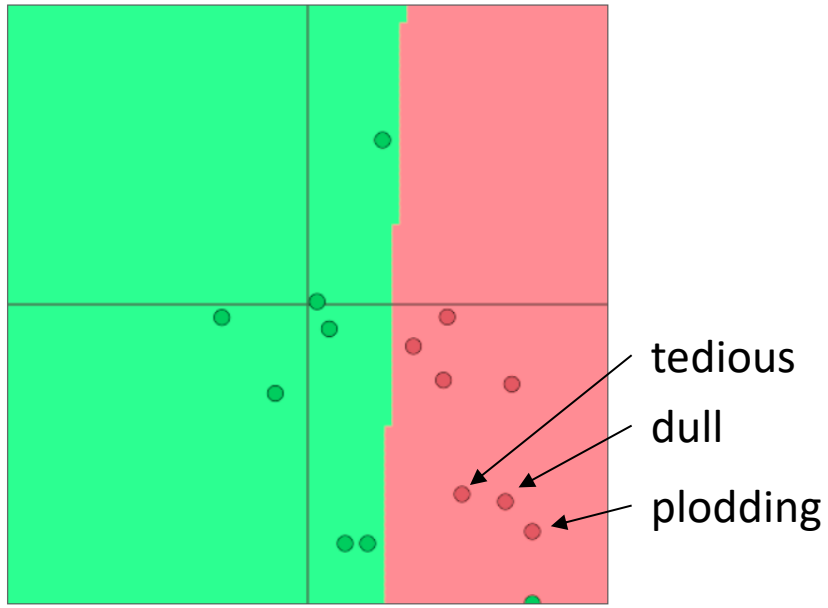
- Pooling: max-over-time pooling layer
- Idea: capture most important activation (maximum over time)
- Use multiple filter weights **w** (i.e., multiple channels)
- From feature map $\mathbf{c} = [c_1, c_2, \ldots, c_{n-h+1}] \in \mathbb{R}^{n-h+1}$
- Pooled single number: $\hat{c} = \max\{\mathbf{c}\}$
  - Because of max pooling $\hat{c} = \max\{\mathbf{c}\}$, length of **c** can be variable

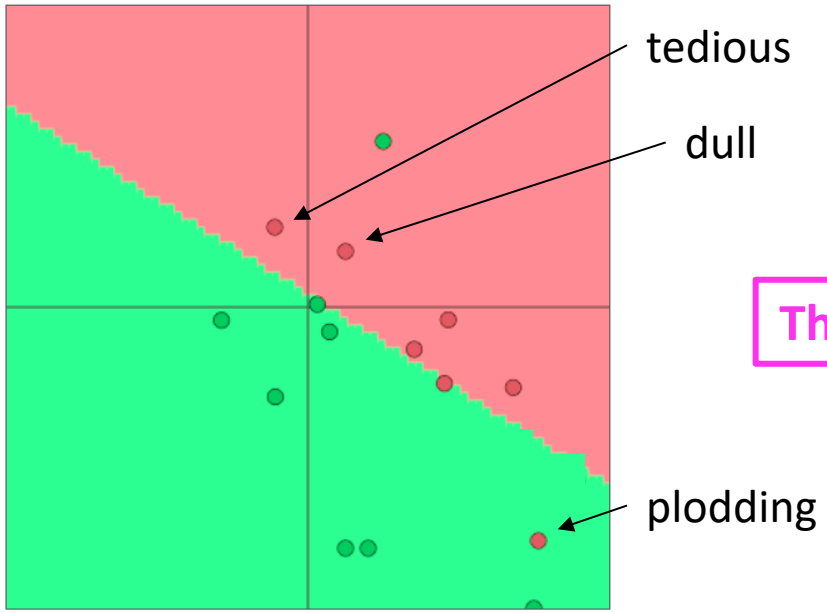
- One convolution layer, followed by one max-pooling
  - To obtain final feature vector: (assuming *m* filters **w**) $\mathbf{z} = [\hat{c}_1, \ldots, \hat{c}_m]$
  - Used 100 feature maps each of sizes 3, 4, 5

- Simple final softmax layer : $y = softmax\left(W^{(S)} z + b\right)$

A pitfall when fine-tuning word vectors

- **Setting:** We are training a model for movie review sentiment building on word vectors
- In the training data we have “tedious”, “dull”; in the testing data we have “plodding”
- The pre-trained word vectors have all three similar:
- **Question: What happens when we update the word vectors?**
- **Answer:** Words in the training data move around; other words stay where they were





This can be bad!

A solution: Channel doubling multi-channel input idea

- Initialize model with pre-trained word vectors (e.g., word2vec or Glove)

- Start with two copies

- Backprop into only one set, keep other "static"
  - Fine-tuning should be useful for improving word vectors for task
  - But there is a problem that words in pre-training (and maybe runtime data) but not in training data will not move. So, it also makes sense to leave all word vectors where they are and to only update the parameters above the word vectors
  - Having two copies is an attempt to get the best of both worlds

- Both channel sets are added to c_i before max-pooling

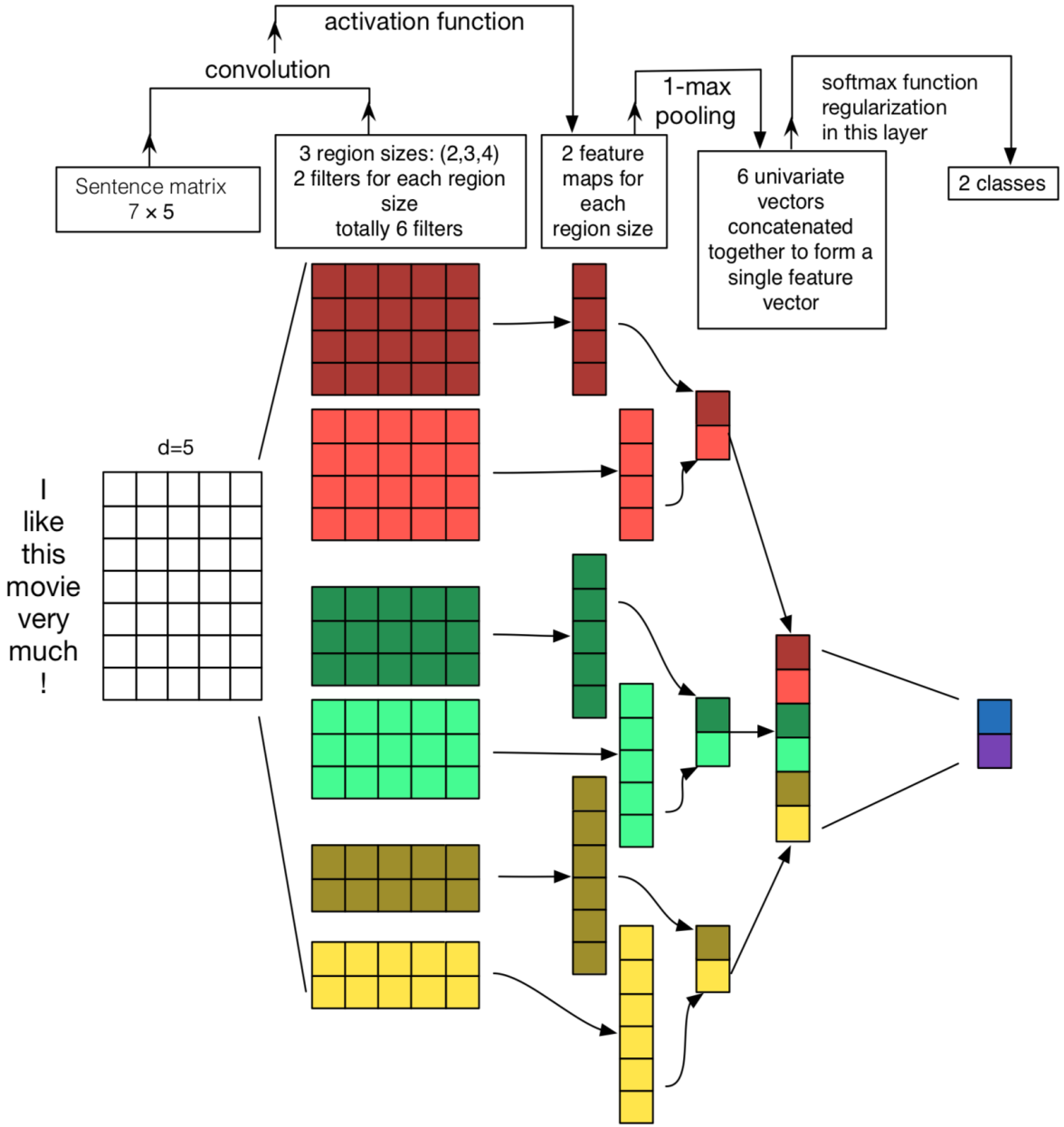
Kim (2014)

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)



Experiments on text classification

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**	–	–

Be careful of fine-points in comparisons!

- Kim (2014) uses dropout, reporting that it gives 2–4 % accuracy improvement!
- But several compared-to systems came earlier and hence didn't use dropout (from 2012/2014) and would probably gain equally from it

- Still seen as remarkable results from a simple architecture!

- Differences from window architecture we described in an early lecture:
  - Many filters and pooling

4. Model comparison: Our growing toolkit

- **Bag of Vectors**: Surprisingly good baseline for simple classification problems
  - Especially if followed by a few ReLU layers! (See paper: Deep Averaging Networks)
- **Window Model**: Good for single word classification for problems that do not need wide context. E.g., POS, NER
- **CNNs:** good for classification, need zero padding for shorter phrases, somewhat implausible/hard to interpret, **easy to parallelize on GPUs;** efficient and versatile
- **Recurrent Neural Networks**: Cognitively plausible (reading from left to right), not best for classification (if just use last state), much slower than CNNs, good for sequence tagging and classification, good for language models, better with attention
- **Transformers:** Great for language models, great for sentence calculations; in general, still the best thing since sliced bread for all NLP problems
  - "Vision Transformers" are taking over in vision but some papers argue that CNNs and transformers have complementary advantages, and you can usefully use both

Batch Normalization (BatchNorm)

[Ioffe and Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv:1502.03167.]

- Often used in CNNs
- Transform the convolution output of **a batch** by scaling the activations to have zero mean and unit variance
  - Again, like the familiar Z-transform of statistics
  - Related to LayerNorm, which is standard in Transformers, but crucially different:
    - LayerNorm calculates statistics across all feature dimensions for each instance independently
    - BatchNorm normalizes across all elements and items in a batch for each feature independently
- Use of BatchNorm also makes models **much** less sensitive to parameter initialization, since outputs are automatically rescaled
  - It also tends to make tuning of learning rates simpler
- PyTorch: nn.BatchNorm1d

Size 1 Convolutions

[Lin, Chen, and Yan. 2013. Network in network. arXiv:1312.4400.]

- **Does this concept make sense?!? Yes.**
- Size 1 convolutions (“1x1”), a.k.a. Network-in-network (NiN) connections, are convolutional kernels with kernel_size=1
- A size 1 convolution gives you a fully connected linear layer across channels!
- It can be used to map from many channels to fewer channels
- Size 1 convolutions add additional neural network layers with very few additional parameters
  - Unlike Fully Connected (FC) layer across data item which adds **tons** of parameters
  - This is similar to the per-position feed-forward layers in transformers

5. Very Deep Convolutional Networks for Text Classification

- Conneau, Schwenk, Lecun, Barrault. EACL 2017.
- Starting point: sequence models (LSTMs) had been very dominant in NLP
  - Also CNNs, Attention, etc., but all the models were basically not very deep – not like the deep models in Vision
- What happens when we build a vision-like system for NLP?
- Model works up from the character level
  - Desire for “NLP from scratch” [raw signal]

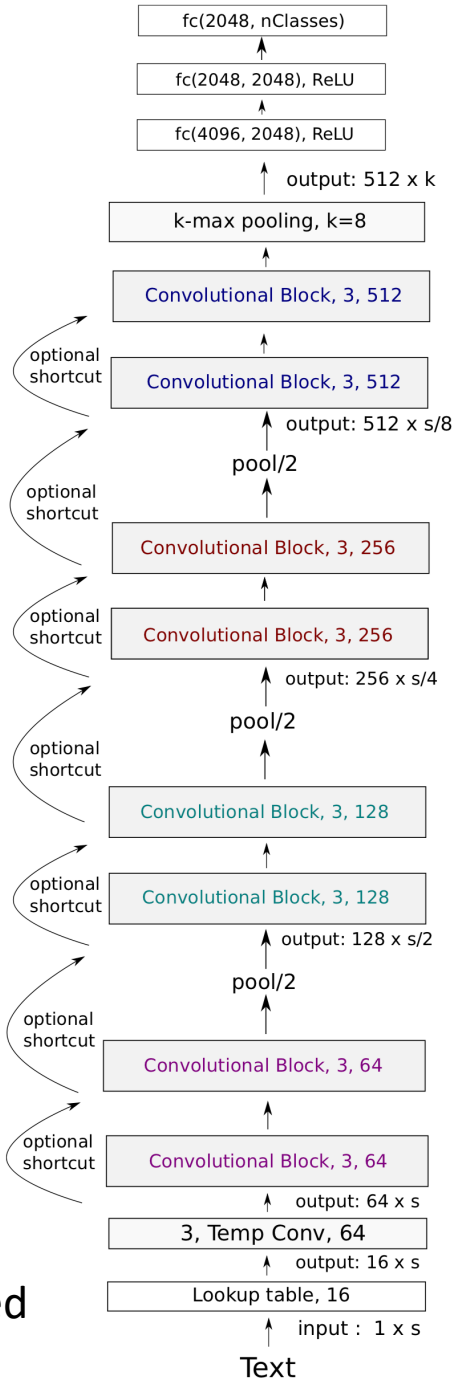
VD-CNN architecture

The system very much looks like a vision system in its design, similar to VGGnet or ResNet

It looks unlike then typical Deep Learning NLP systems

- It looks a bit more like a Transformer?

Result is constant size, since text is truncated or padded

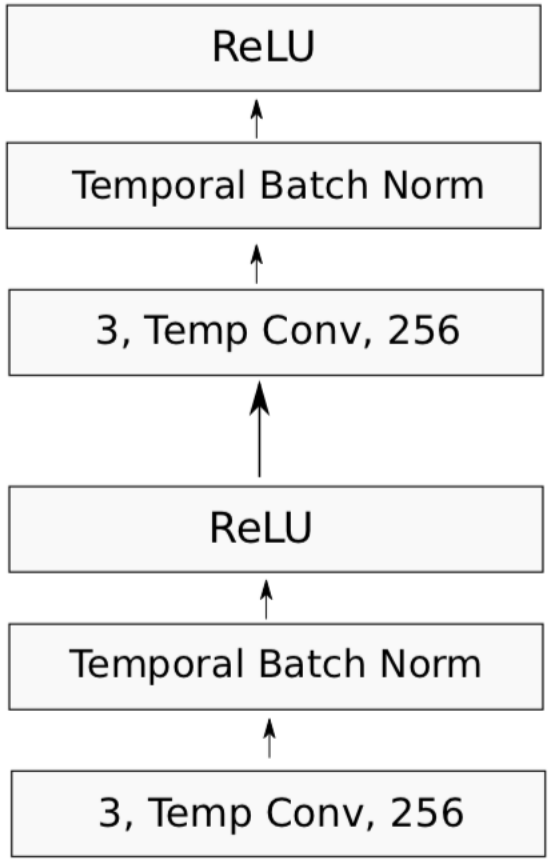


Local pooling at each stage halves temporal resolution and doubles number of features

s = 1024 chars; 16d embed

Convolutional block in VD-CNN

- Each convolutional block is two convolutional layers, each followed by batch norm and a ReLU nonlinearity
- Convolutions of size 3
- Pad to preserve (or halve when local pooling) dimension



Experiments

- Use large text classification datasets
  - Much bigger than the small datasets used in the Yoon Kim (2014) paper

Data set	#Train	#Test	#Classes	Classification Task
AG's news	120k	7.6k	4	English news categorization
Sogou news	450k	60k	5	Chinese news categorization
DBPedia	560k	70k	14	Ontology classification
Yelp Review Polarity	560k	38k	2	Sentiment analysis
Yelp Review Full	650k	50k	5	Sentiment analysis
Yahoo! Answers	1 400k	60k	10	Topic classification
Amazon Review Full	3 000k	650k	5	Sentiment analysis
Amazon Review Polarity	3 600k	400k	2	Sentiment analysis

Experiments

Corpus:	AG	Sogou	DBP.	Yelp P.	Yelp F.	Yah. A.	Amz. F.	Amz. P.
Method	n-TFIDF	n-TFIDF	n-TFIDF	ngrams	Conv	Conv+RNN	Conv	Conv
Author	[Zhang]	[Zhang]	[Zhang]	[Zhang]	[Zhang]	[Xiao]	[Zhang]	[Zhang]
Error	7.64	2.81	1.31	4.36	37.95*	28.26	40.43*	4.93*
[Yang]	-	-	-	-	-	24.2	36.4	-

Table 4: Best published results from previous work. Zhang et al. (2015) best results use a Thesaurus data augmentation technique (marked with an *). Yang et al. (2016)'s hierarchical methods is particularly

Depth	Pooling	AG	Sogou	DBP.	Yelp P.	Yelp F.	Yah. A.	Amz. F.	Amz. P.
9	Convolution	10.17	4.22	1.64	5.01	37.63	28.10	38.52	4.94
9	KMaxPooling	9.83	3.58	1.56	5.27	38.04	28.24	39.19	5.69
9	MaxPooling	9.17	3.70	1.35	4.88	36.73	27.60	37.95	4.70
17	Convolution	9.29	3.94	1.42	4.96	36.10	27.35	37.50	4.53
17	KMaxPooling	9.39	3.51	1.61	5.05	37.41	28.25	38.81	5.43
17	MaxPooling	8.88	3.54	1.40	4.50	36.07	27.51	37.39	4.41
29	Convolution	9.36	3.61	1.36	4.35	**35.28**	27.17	37.58	**4.28**
29	KMaxPooling	**8.67**	**3.18**	1.41	4.63	37.00	27.16	38.39	4.94
29	MaxPooling	8.73	3.36	**1.29**	**4.28**	35.74	**26.57**	**37.00**	4.31

Table 5: Testing error of our models on the 8 data sets. No data preprocessing or augmentation is used.

6. TreeRNNs: Recursion in human language

SCIENCE'S COMPASS • REVIEW

REVIEW: NEUROSCIENCE

The Faculty of Language: What Is It, Who Has It, and How Did It Evolve?

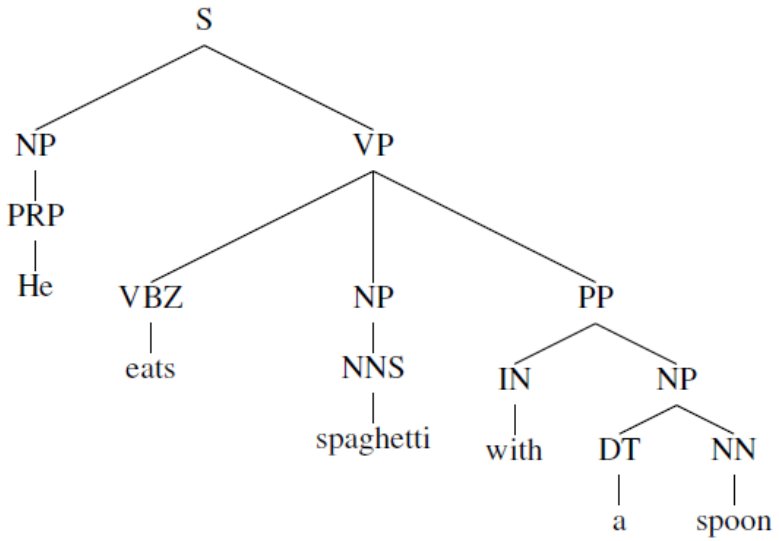
Marc D. Hauser,[1]* Noam Chomsky,[2] W. Tecumseh Fitch[1]

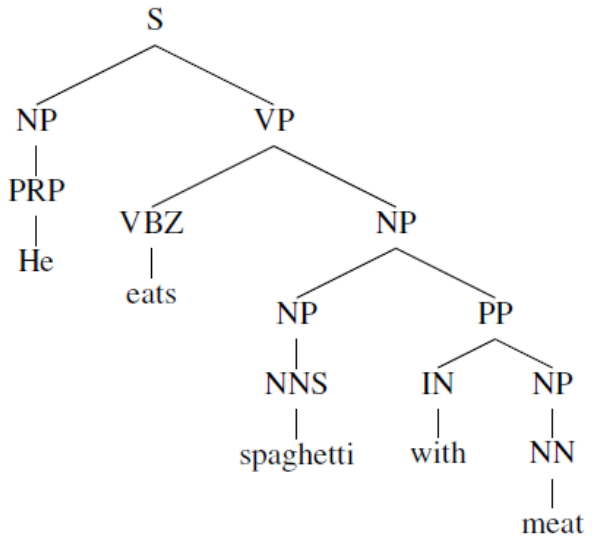
We argue that an understanding of the faculty of language requires substantial interdisciplinary cooperation. We suggest how current developments in linguistics can be profitably wedded to work in evolutionary biology, anthropology, psychology, and neuroscience. We submit that a distinction should be made between the faculty of language in the broad sense (FLB) and in the narrow sense (FLN). FLB includes a sensory-motor system, a conceptual-intentional system, and the computational mechanisms for recursion, providing the capacity to generate an infinite range of expressions from a finite set of elements. We hypothesize that FLN only includes recursion and is the only uniquely human component of the faculty of language. We further argue that FLN may have evolved for reasons other than language, hence comparative studies might look for evidence of such computations outside of the domain of communication (for example, number, navigation, and social relations).

If a martian graced our planet, it would be struck by one remarkable similarity among Earth's living creatures and a key difference. Concerning similarity, it would note that all

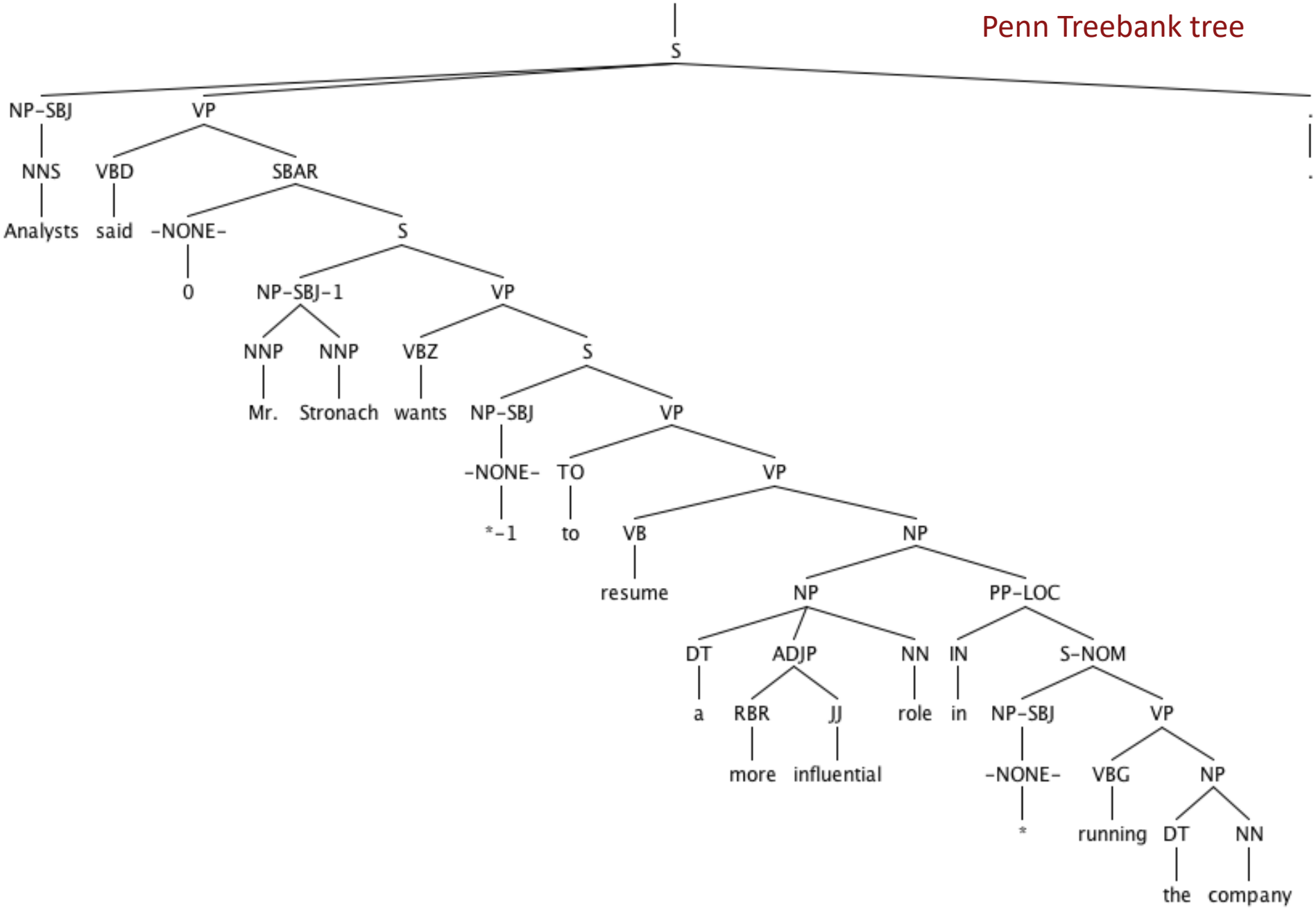
Are languages recursive?

- Cognitively somewhat debatable (need to head to infinity)
- But: recursive structure is natural/right for describing language
  - *[The person standing next to [the man from [the company that purchased [the firm that you used to work at]]]]*
  - noun phrase containing a noun phrase containing a noun phrase
- It's a very powerful prior for language structure





Penn Treebank tree

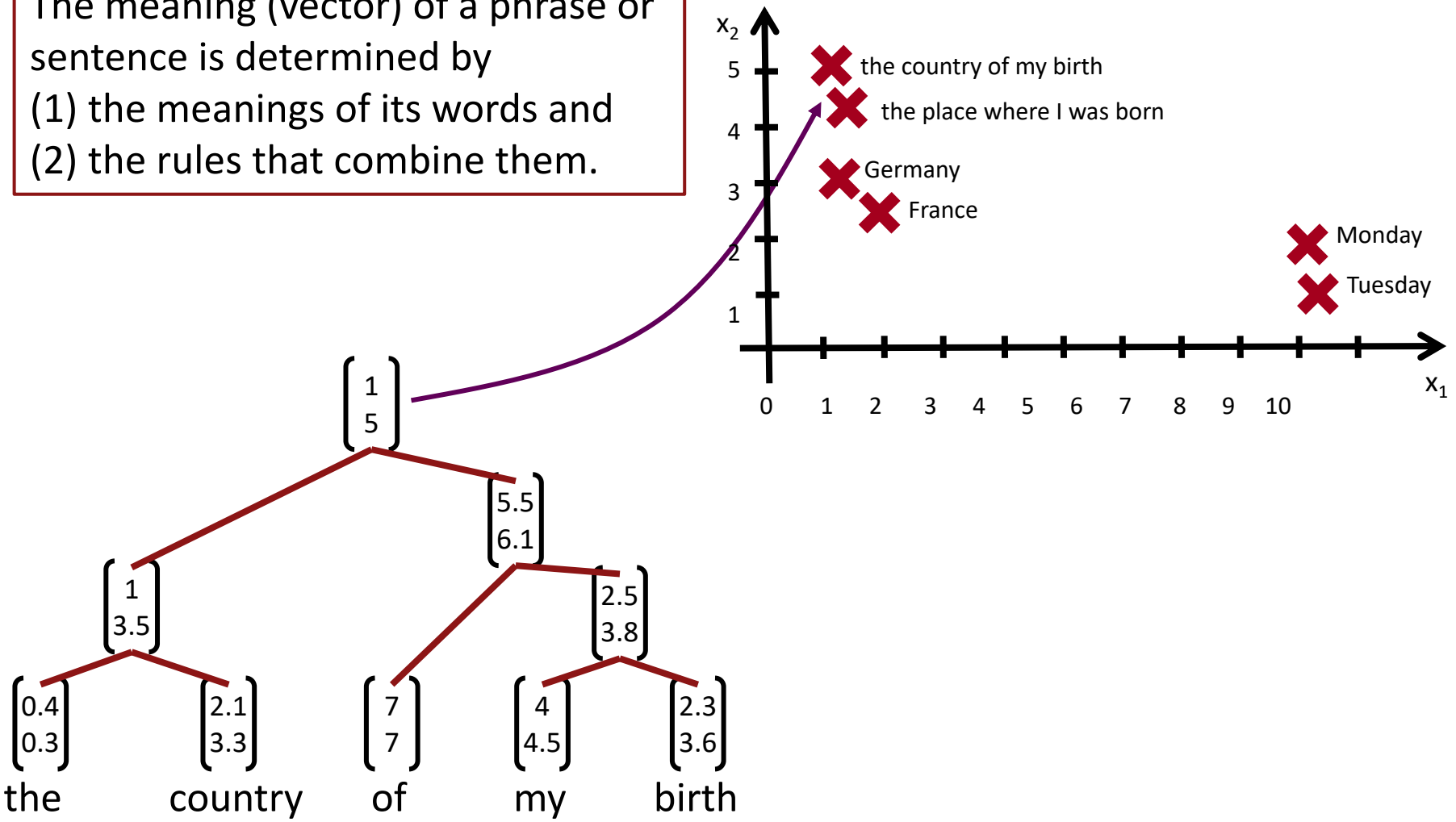


How should we map phrases into a vector space?

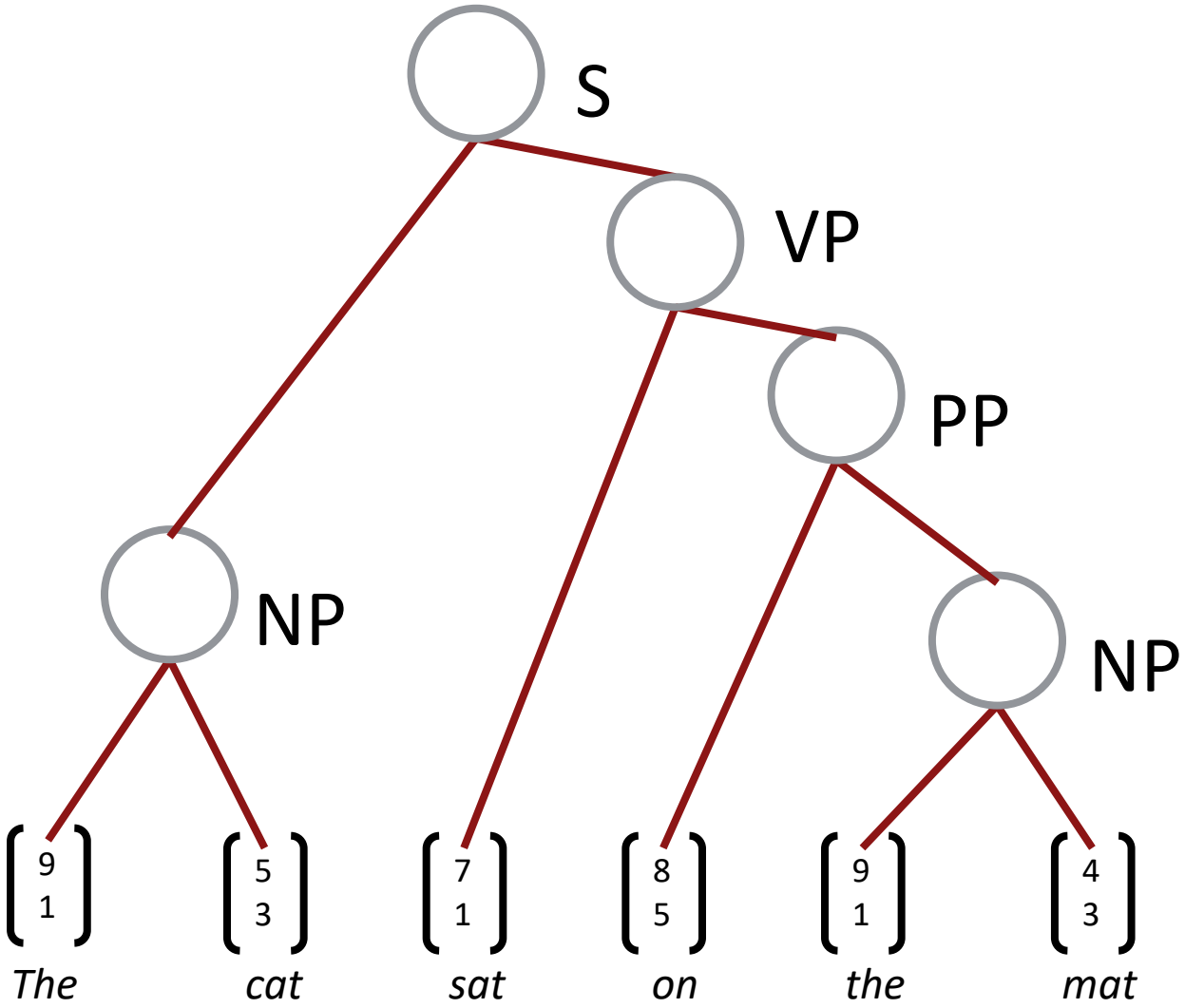
Socher, Manning, and Ng. ICML, 2011

Use principle of compositionality

The meaning (vector) of a phrase or sentence is determined by
(1) the meanings of its words and
(2) the rules that combine them.

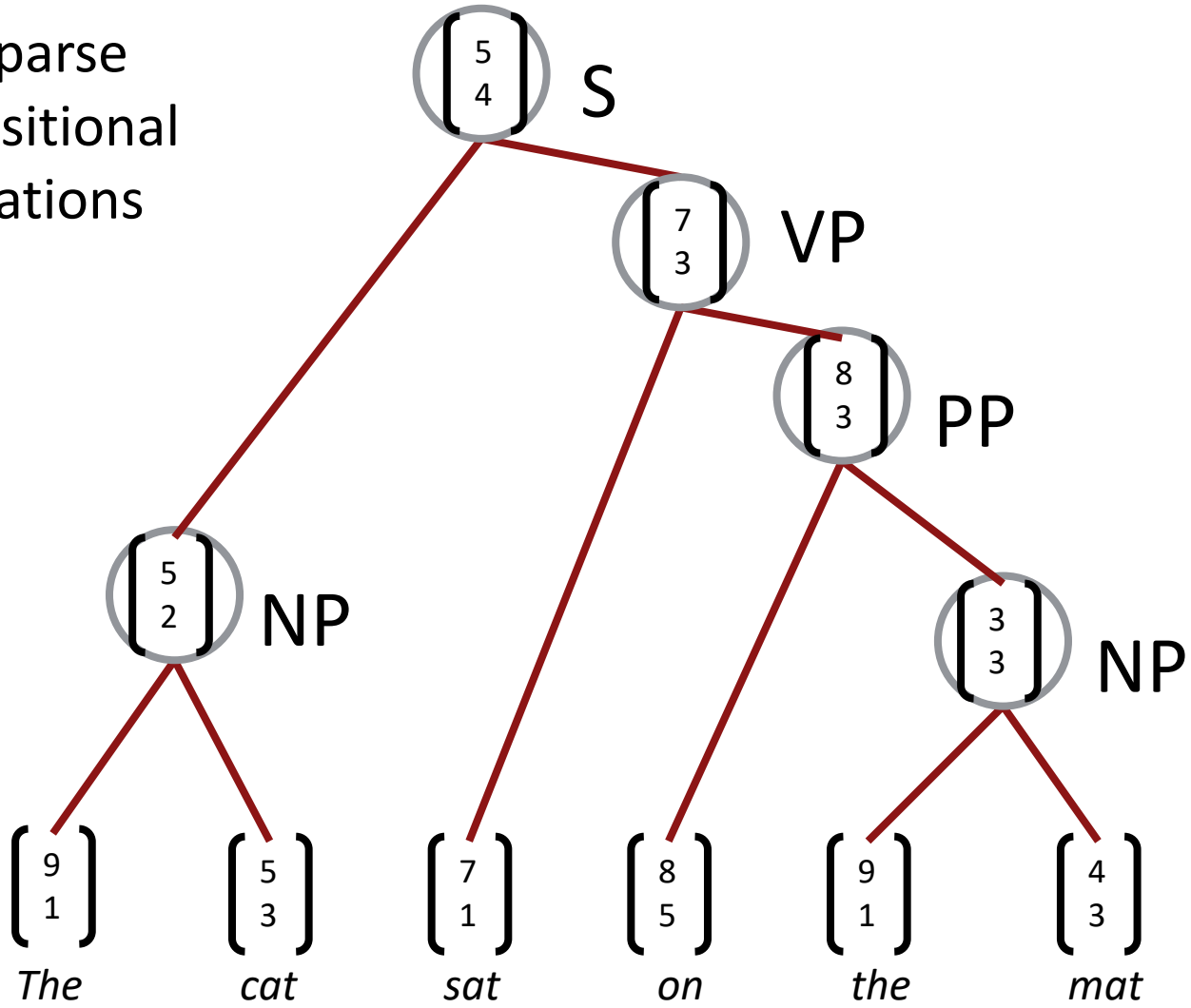


Constituency Sentence Parsing: What we want



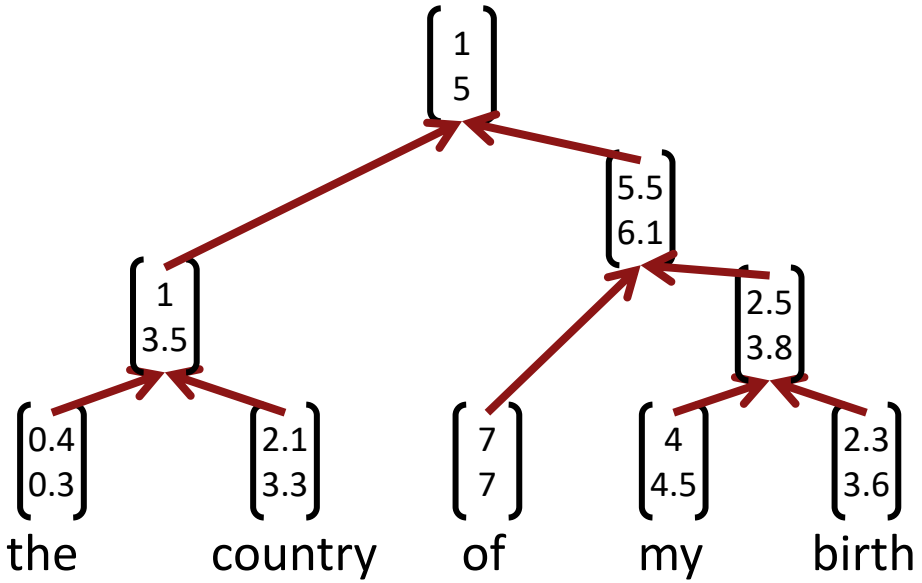
Learn Structure and Representation

Models in this section can jointly learn parse trees and compositional vector representations

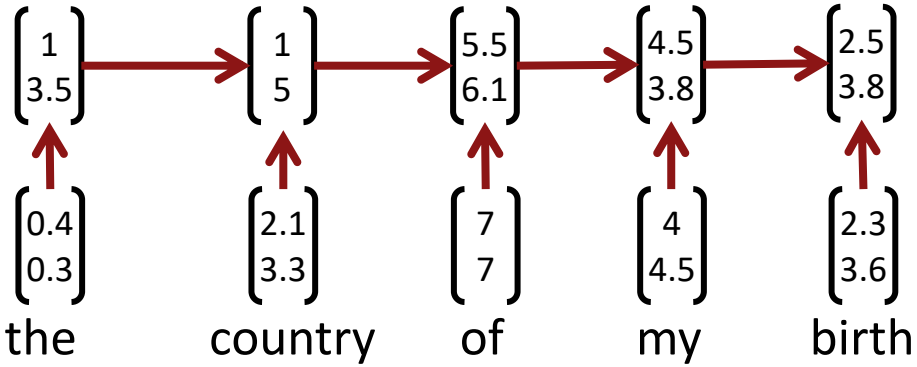


Recursive vs. recurrent neural networks

- Recursive neural nets provide representations for linguistic phrases
- But they require a tree structure



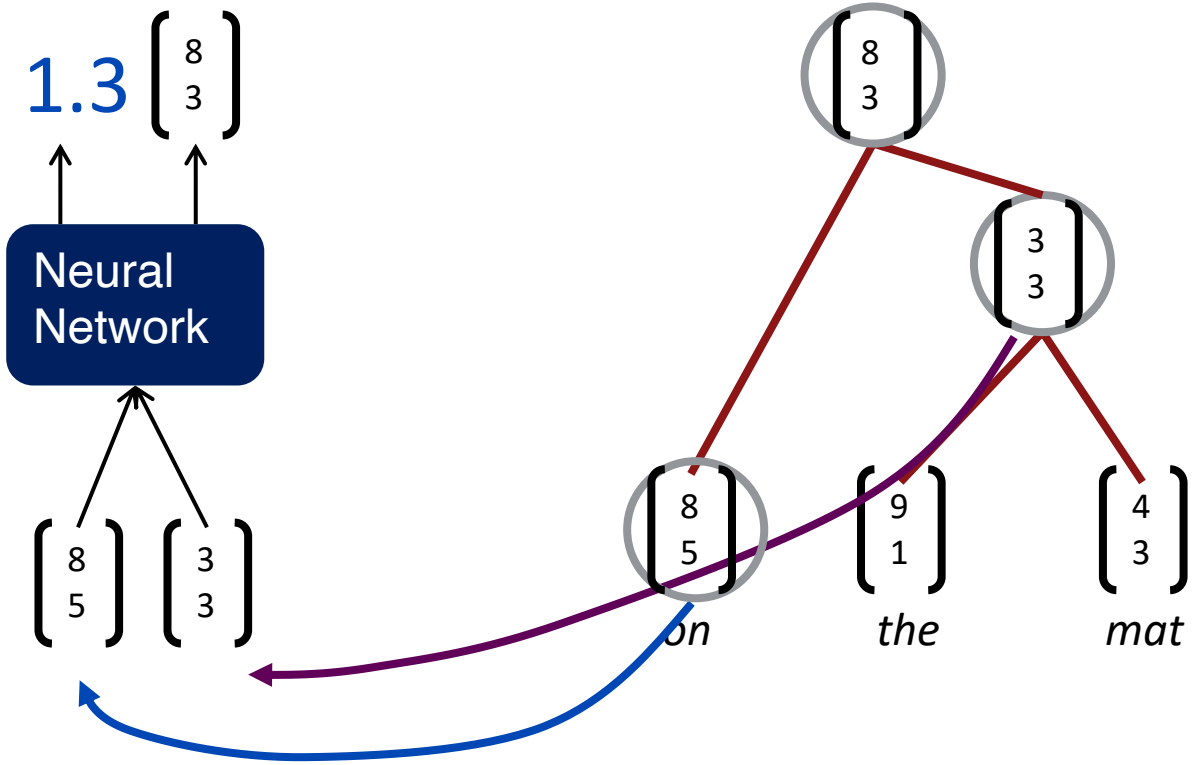
- Recurrent neural nets cannot capture phrases without prefix context
- They often capture too much of last words in “phrase” vector



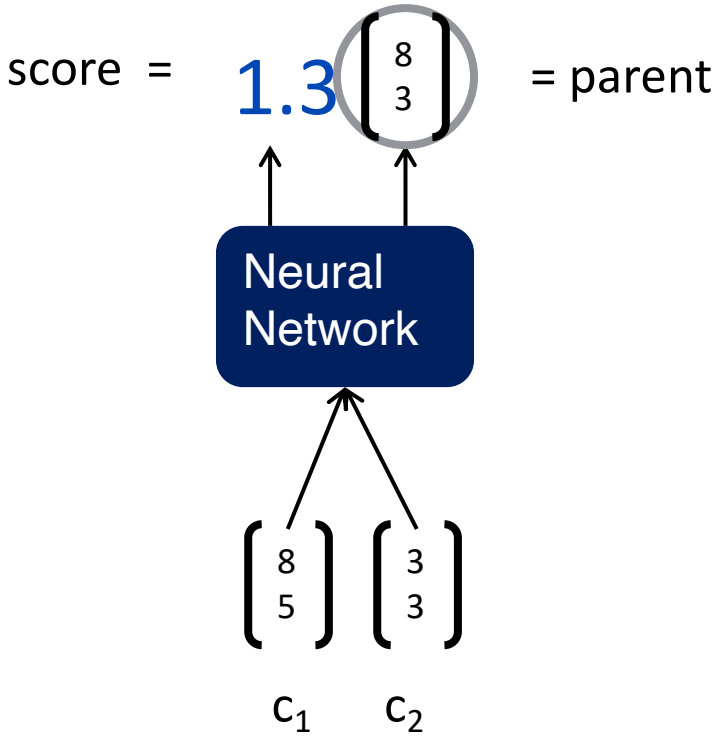
Recursive Neural Networks for Structure Prediction

Inputs: two candidate children's representations
Outputs:

1. The semantic representation if the two nodes are merged.
2. Score of how plausible the new node would be.



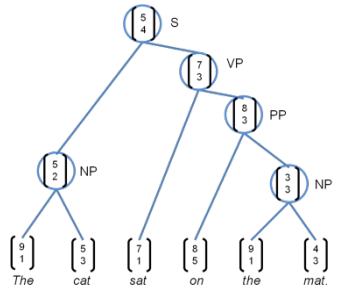
Simple Tree Recursive Neural Network Definition



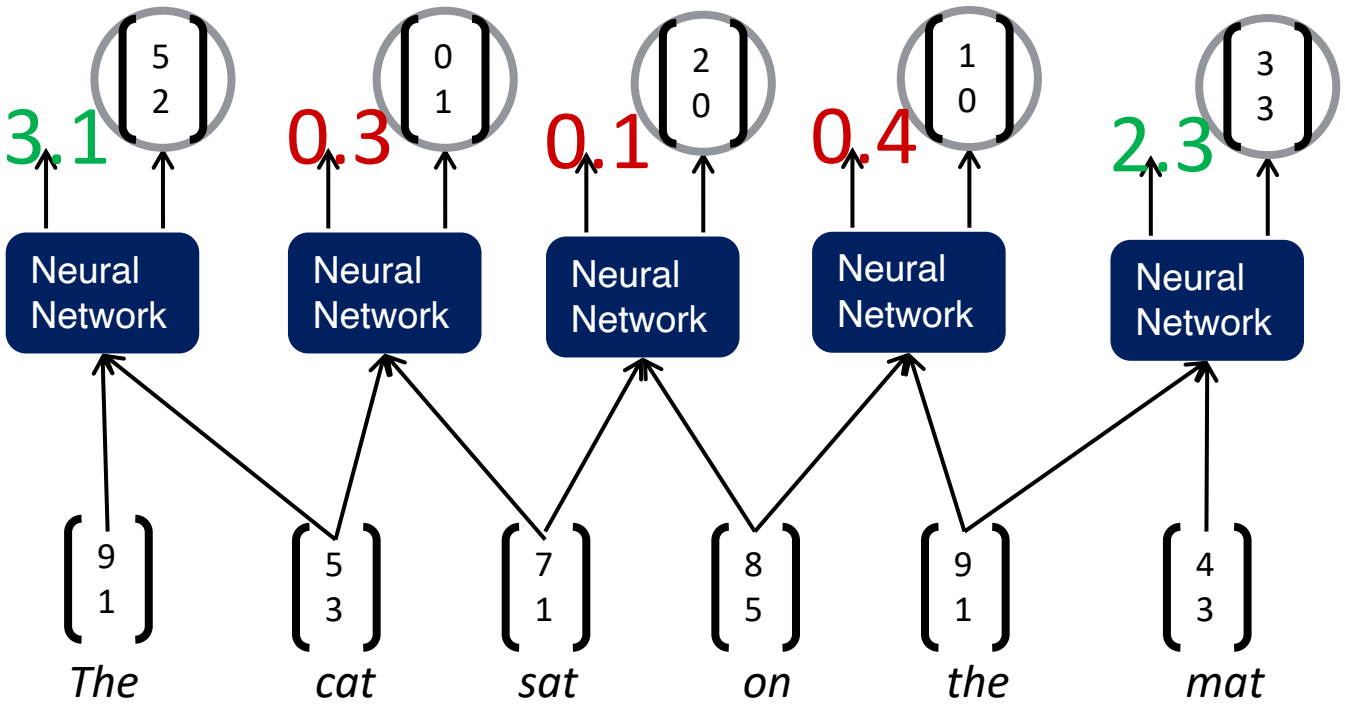
score = $U^T p$

$$p = \tanh\left(W \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} + b\right),$$

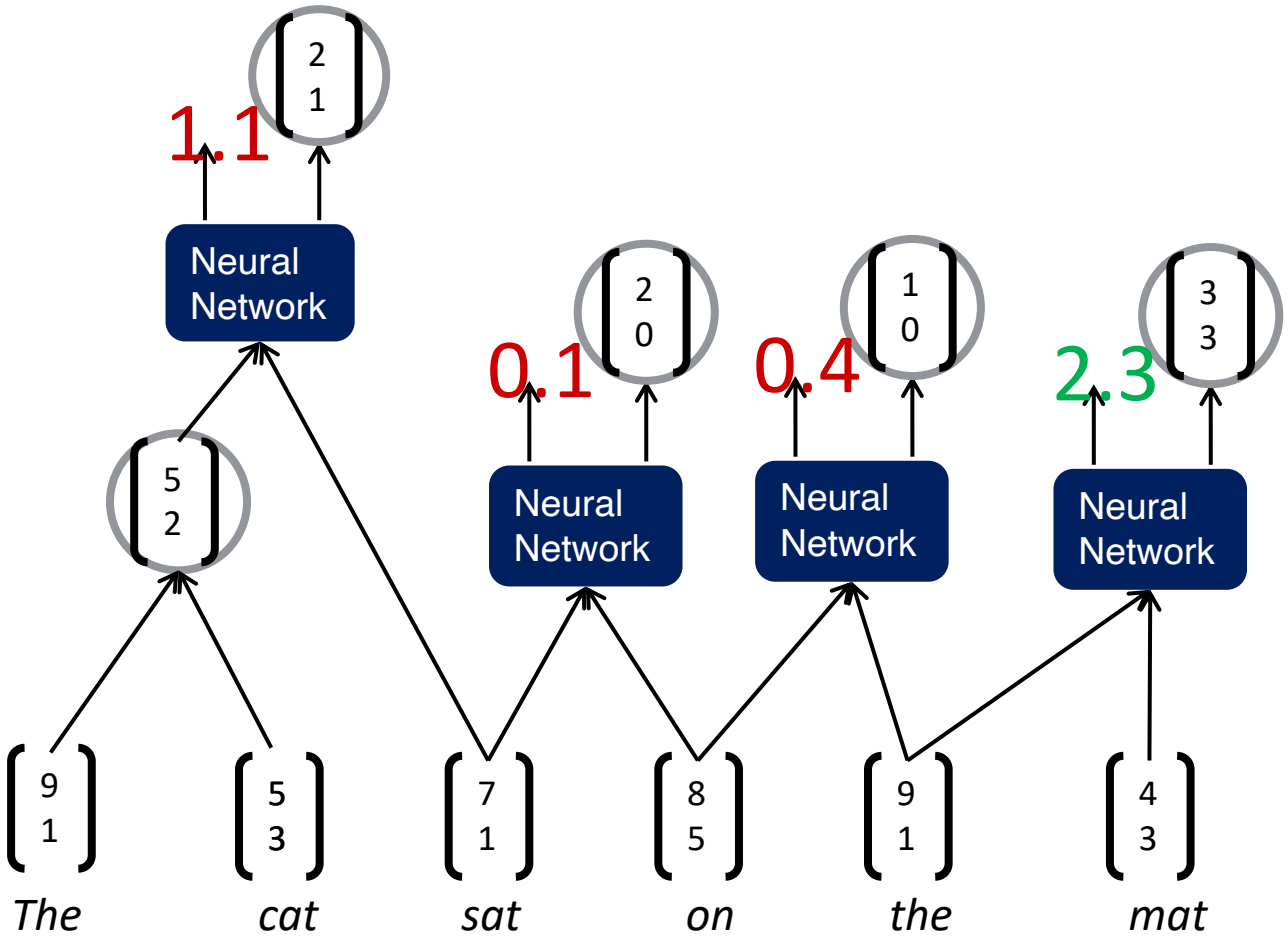
Same W parameters at all nodes of the tree



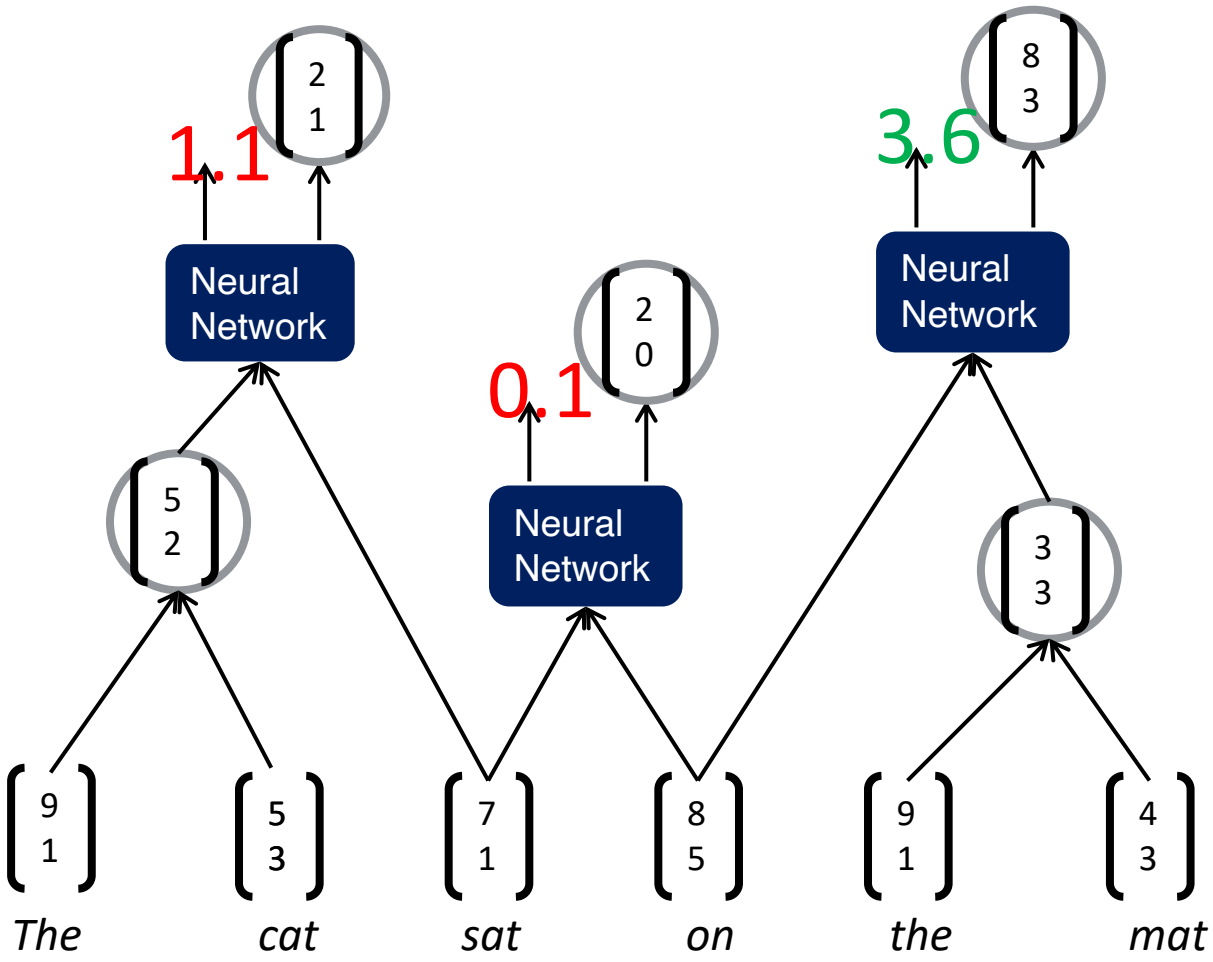
Parsing a sentence with an RNN (greedily)



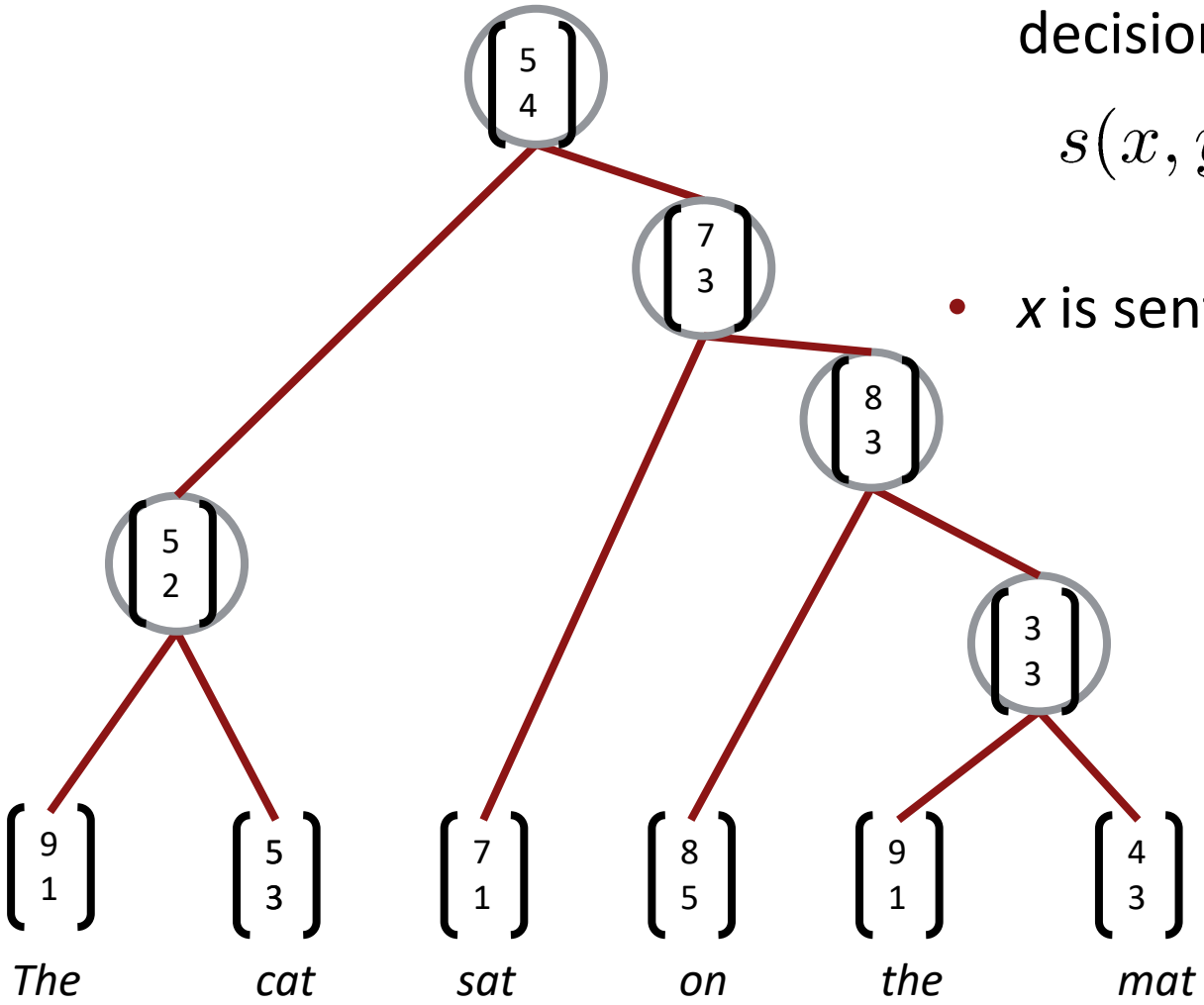
Parsing a sentence



Parsing a sentence



Parsing a sentence



- The score of a tree is computed by the sum of the parsing decision scores at each node:

$$s(x, y) = \sum_{n \in nodes(y)} s_n$$

- *x* is sentence; *y* is parse tree

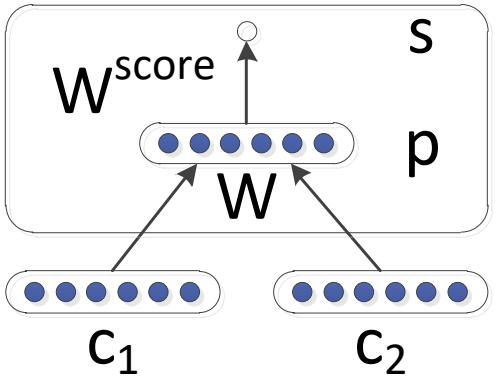
Discussion: Simple TreeRNN

- We got some decent results with a single layer TreeRNN like this!
  - [Socher, Manning, and Ng. ICML, 2011] got a best paper award!

- A single weight matrix TreeRNN could capture some things but not more complex, higher order composition and parsing long sentences

- There is no real interaction between the input words

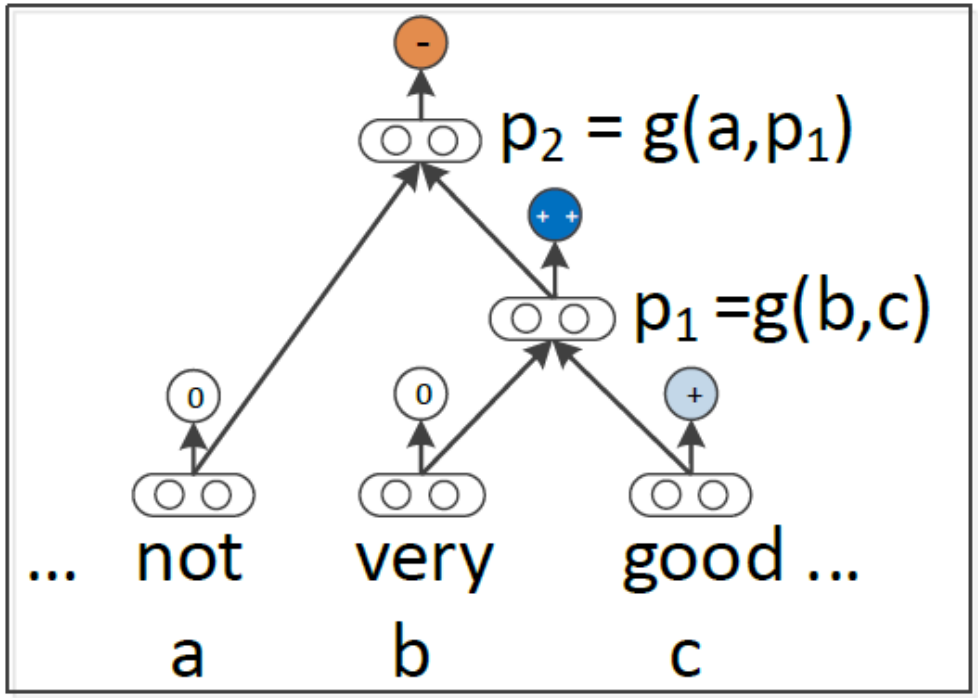
- And the composition function is the same for all syntactic categories, punctuation, etc.

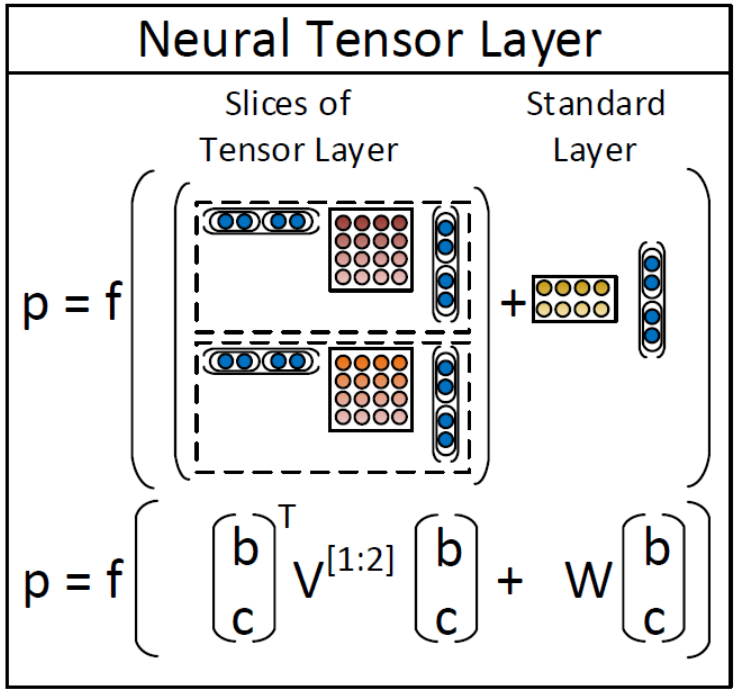


7. Recursive Neural Tensor Networks

Socher, Perelygin, Wu, Chuang, Manning, Ng, and Potts 2013

- Allows two word or phrase vectors to interact multiplicatively





- Not today, but see also Tai, Socher, Manning [2015]: TreeLSTMs
  - Work even better

Beyond the bag of words: Sentiment detection

Is the tone of a piece of text positive, negative, or neutral?

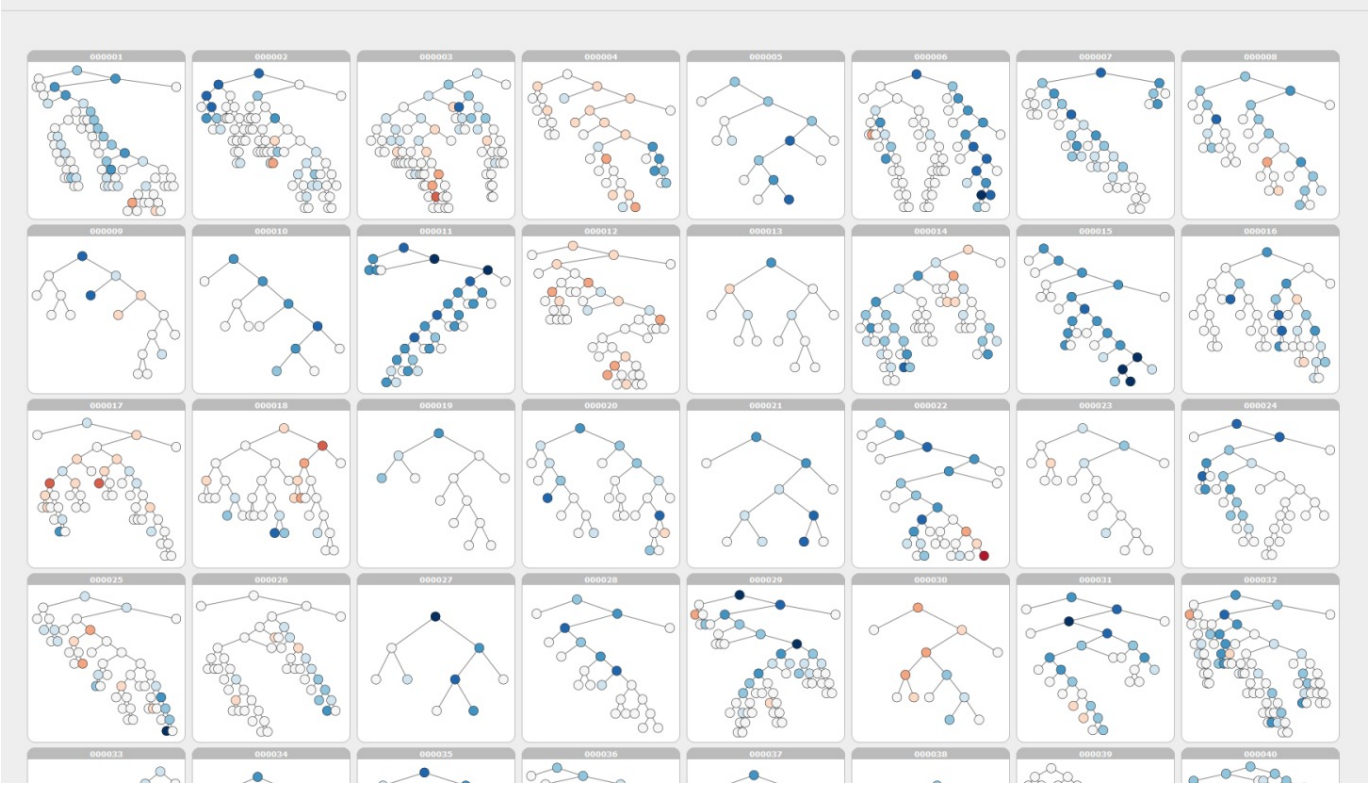
- Sentiment is that sentiment is “easy”
- Detection accuracy for longer documents ~90%, BUT

… … loved … … … … … great … … … … … … impressed … … … … … … marvelous … … … …

With this cast, and this subject matter, the movie should have been funnier and more entertaining.

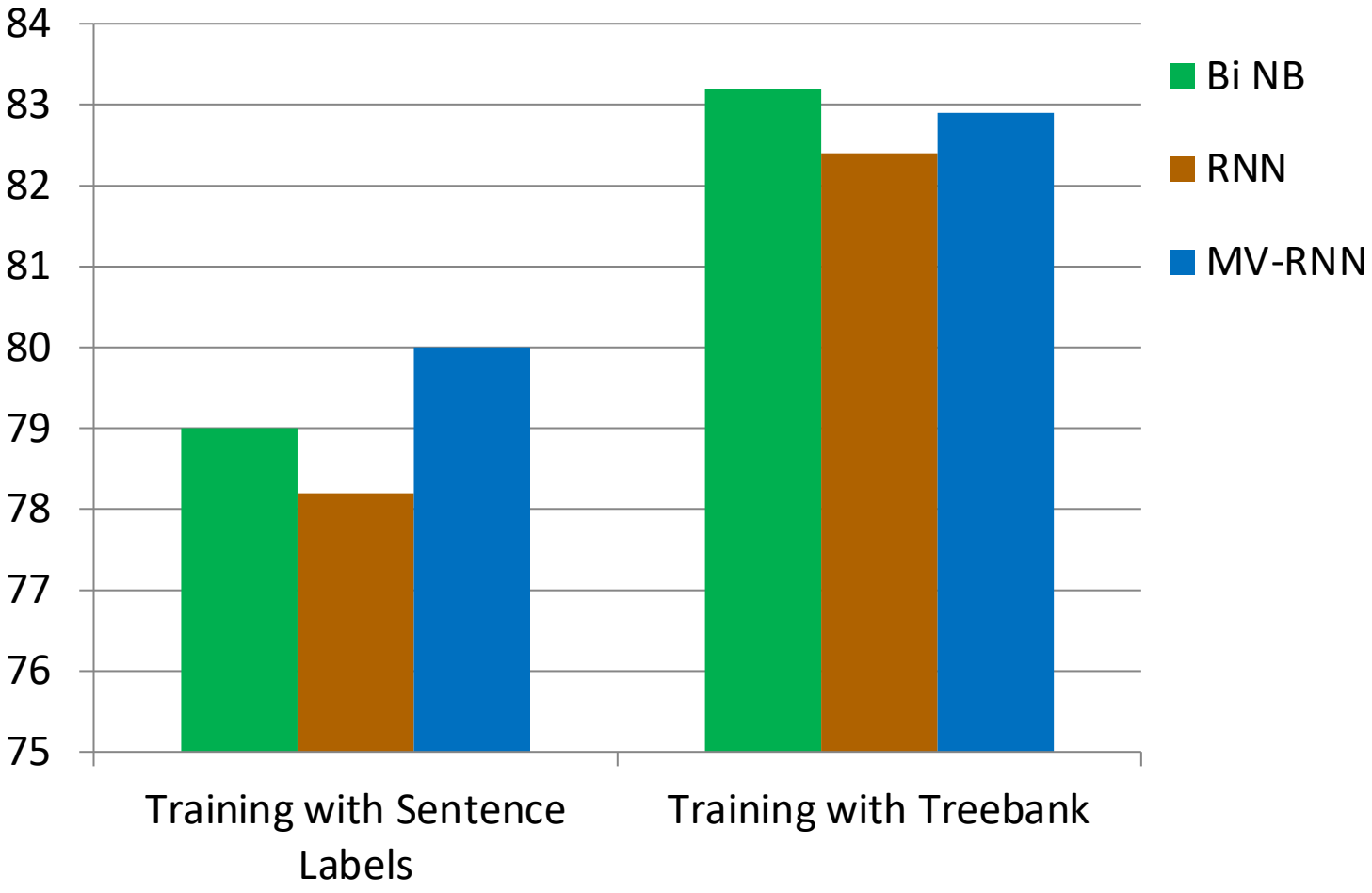
Stanford Sentiment Treebank

- 215,154 phrases labeled in 11,855 sentences
- Can actually train and test compositions



http://nlp.stanford.edu:8080/sentiment/

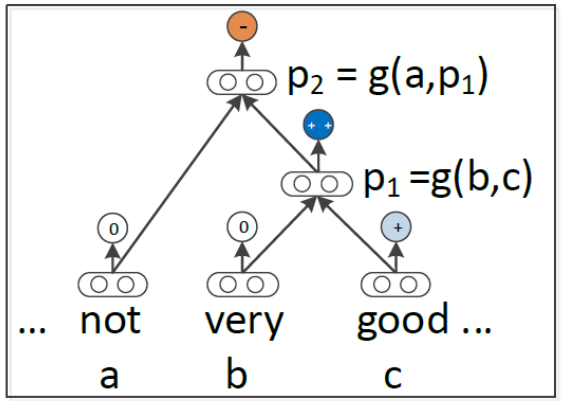
Better Dataset Helped All Models

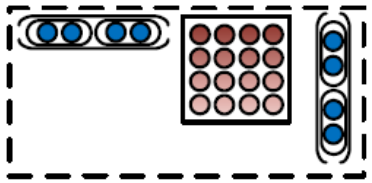


- Hard negation cases are still mostly incorrect
- We also need a more powerful model!

Recursive Neural Tensor Network

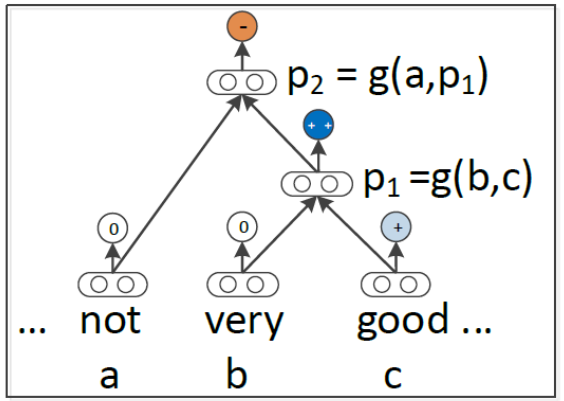
Idea: Allow both additive and mediated multiplicative interactions of vectors

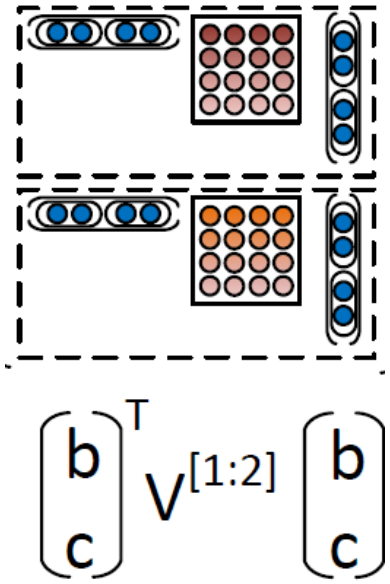




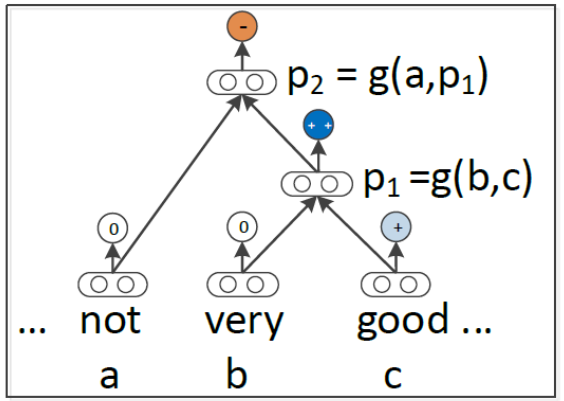
$$\begin{bmatrix} b \\ c \end{bmatrix}^T V \begin{bmatrix} b \\ c \end{bmatrix}$$

Recursive Neural Tensor Network





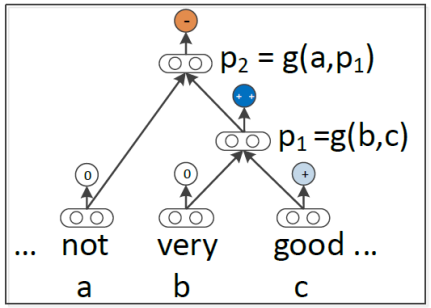
Recursive Neural Tensor Network



$$\begin{bmatrix} b \\ c \end{bmatrix}^T V^{[1:2]} \begin{bmatrix} b \\ c \end{bmatrix} + W \begin{bmatrix} b \\ c \end{bmatrix}$$

Recursive Neural Tensor Network

- Use resulting vectors in tree as input to a classifier like logistic regression
- Train all weights jointly with gradient descent

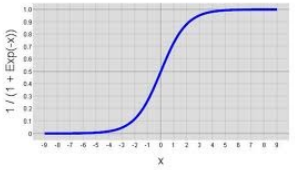


Neural Tensor Layer

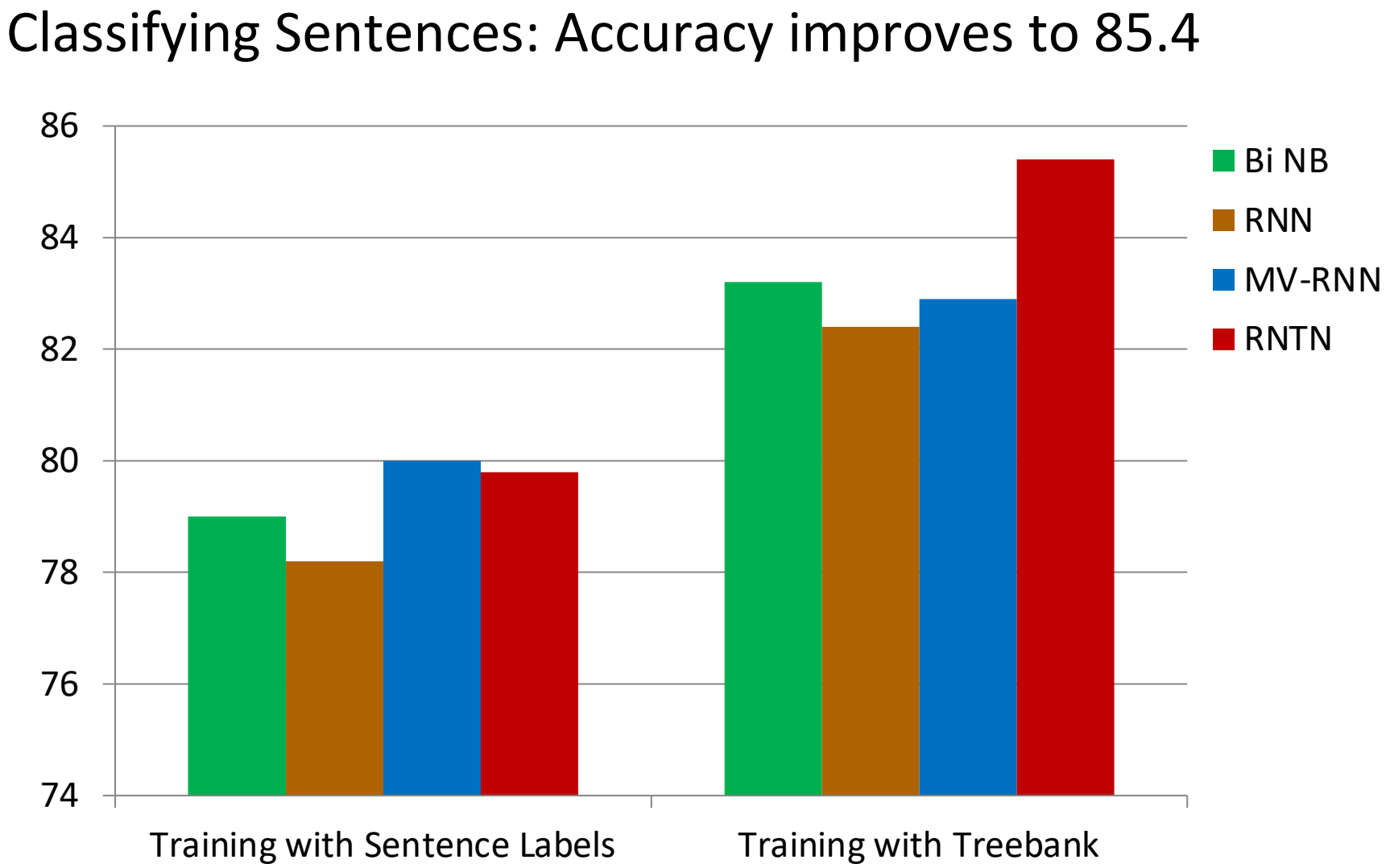
Slices of Tensor Layer

Standard Layer

$$p = f\left(\begin{bmatrix} b \\ c \end{bmatrix}^T V^{[1:2]} \begin{bmatrix} b \\ c \end{bmatrix} + W \begin{bmatrix} b \\ c \end{bmatrix} \right)$$

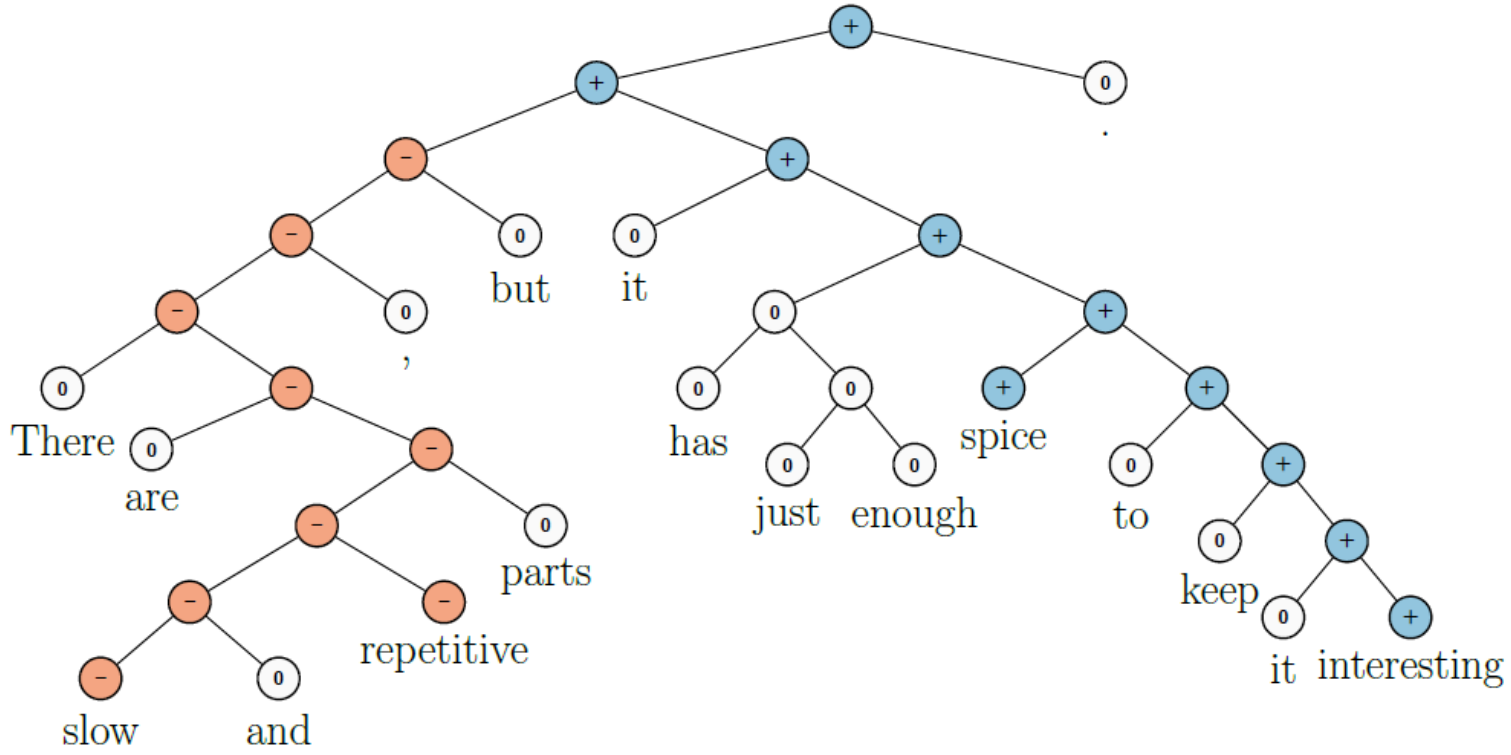


Positive/Negative Results on Treebank



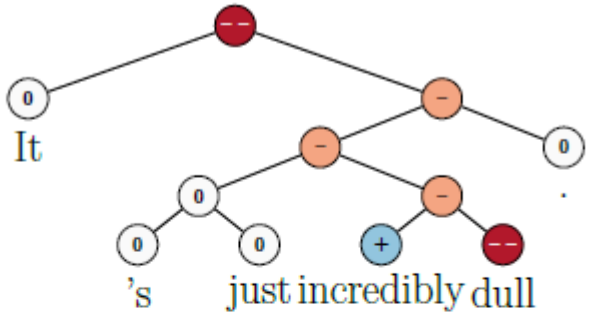
Experimental Results on Treebank

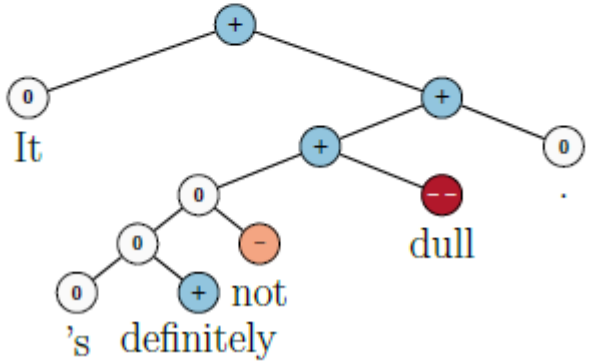
- RNTN can capture constructions like *X but Y*
- RNTN accuracy of 72%, compared to MV-RNN (65%), biword NB (58%) and RNN (54%)

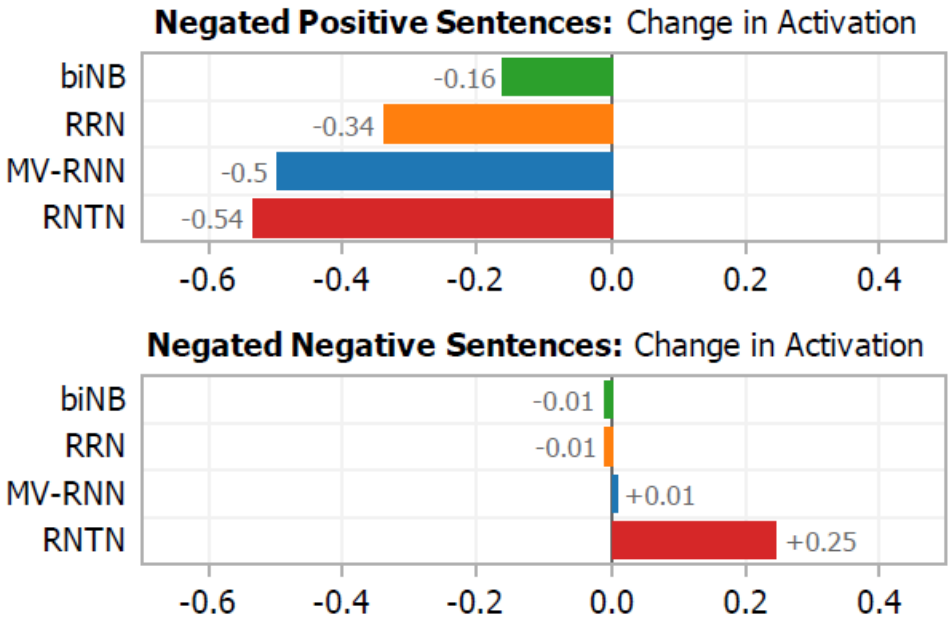


Negation Results

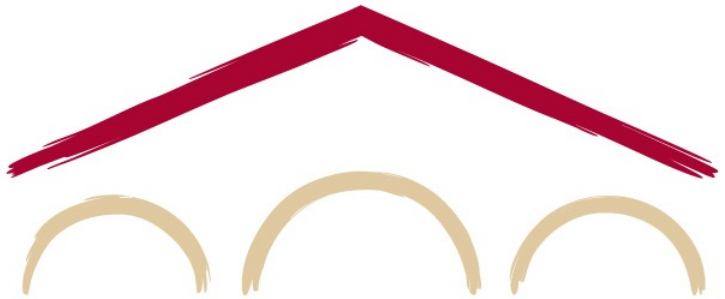
When negating negatives, positive activation should increase!







Natural Language Processing with Deep Learning CS224N/Ling284



Christopher Manning

Lecture 16: ConvNets for NLP and Tree Recursive Neural Networks