Neural networks for natural language understanding

Sam Bowman

Department of Linguistics and NLP Group

Stanford University

with Chris Potts, Chris Manning

Today

Promising signs that neural network models can learn to handle semantic tasks:

- Sentiment and semantic similarity (e.g., Tai et al. 2015)
- Paraphrase detection (Socher et al. 2011)
- Machine translation (e.g., Sutskever et al. 2014, Bahdanau et al. 2014)

How do these models work?

How well can they handle anything we'd recognize as meaning?

Today

How do these models work?

- Survey: Deep learning models for NLU

How well can they handle anything we'd recognize as meaning?

- A measure of success: natural language inference
- Three experiments on artificial data
- Frontiers: What about real language?

NNs for sentence meaning

Input: Word vectors

not *that* *bad*

Output: Sentence vectors

not *that* *bad* → not that bad

Training: Supervised classification over sentence vectors

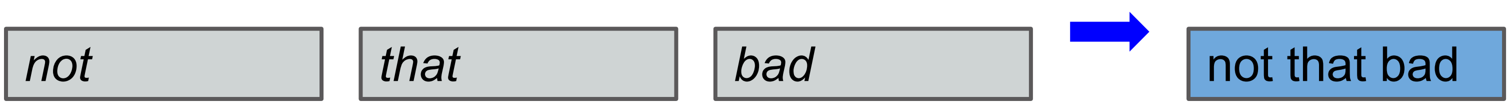
not *that* *bad* → not that bad

Prediction: 2/5

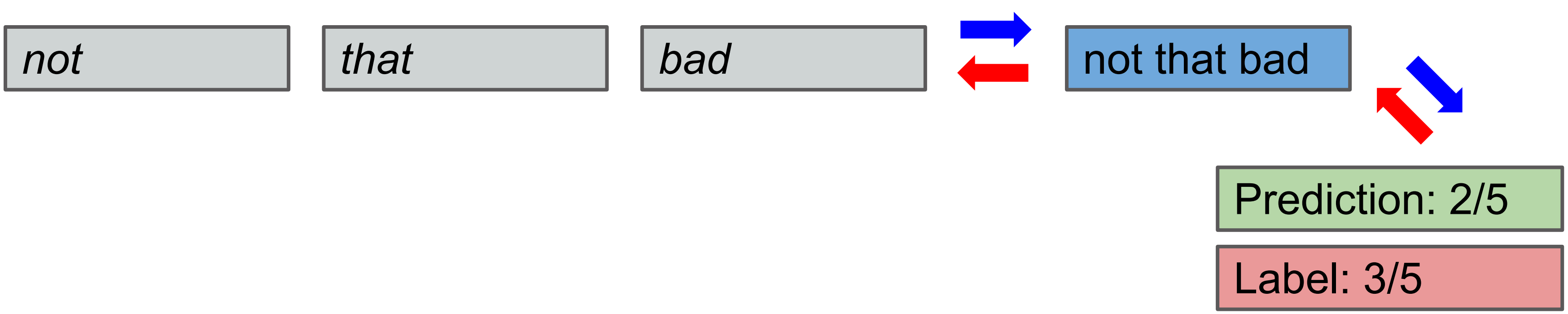
NNs for sentence meaning

Input: Word vectors

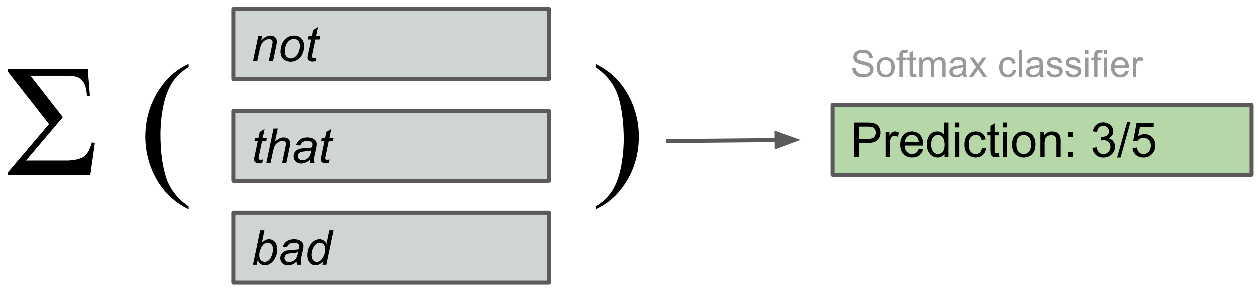
Output: Sentence vectors



Training: Supervised classification over sentence vectors

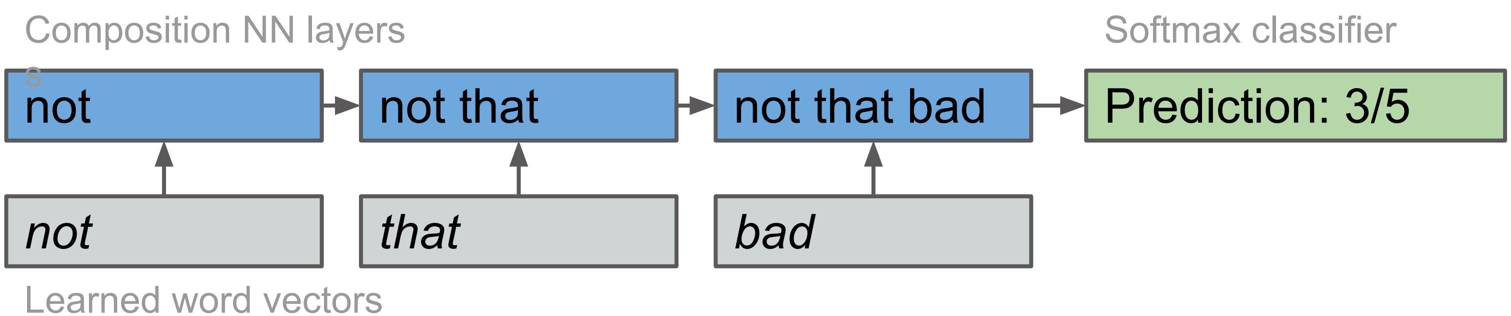


Baseline: Sum of words



- Words and constituents are 25-500d vectors.
- Optimize with SGD
- Gradients from backprop

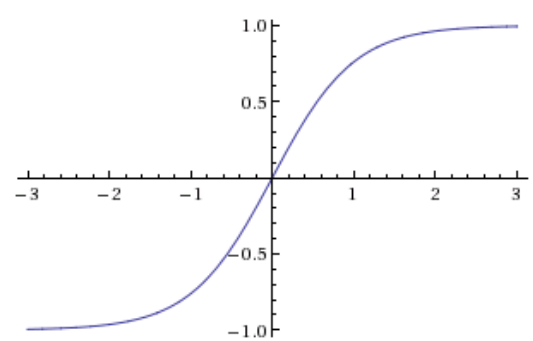
Recurrent NNs for text



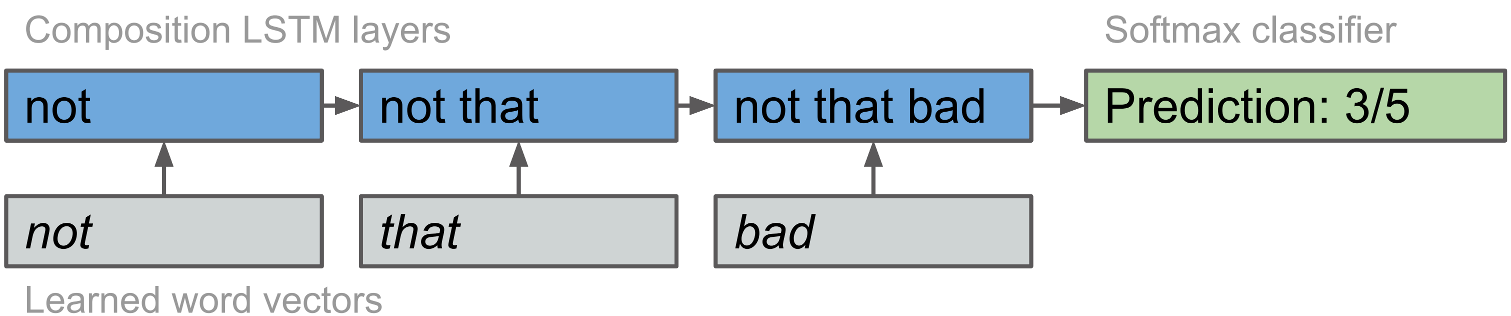
- Words and constituents are 25-500d vectors.
- One learned composition function:
  $\mathbf{y} = f(\mathbf{M}[\mathbf{x}\ \mathbf{y}_{\mathbf{prev}}] + \mathbf{b})$
- Optimize with SGD
- Gradients from backprop (through time)

f(x) = tanh(x)

...usually



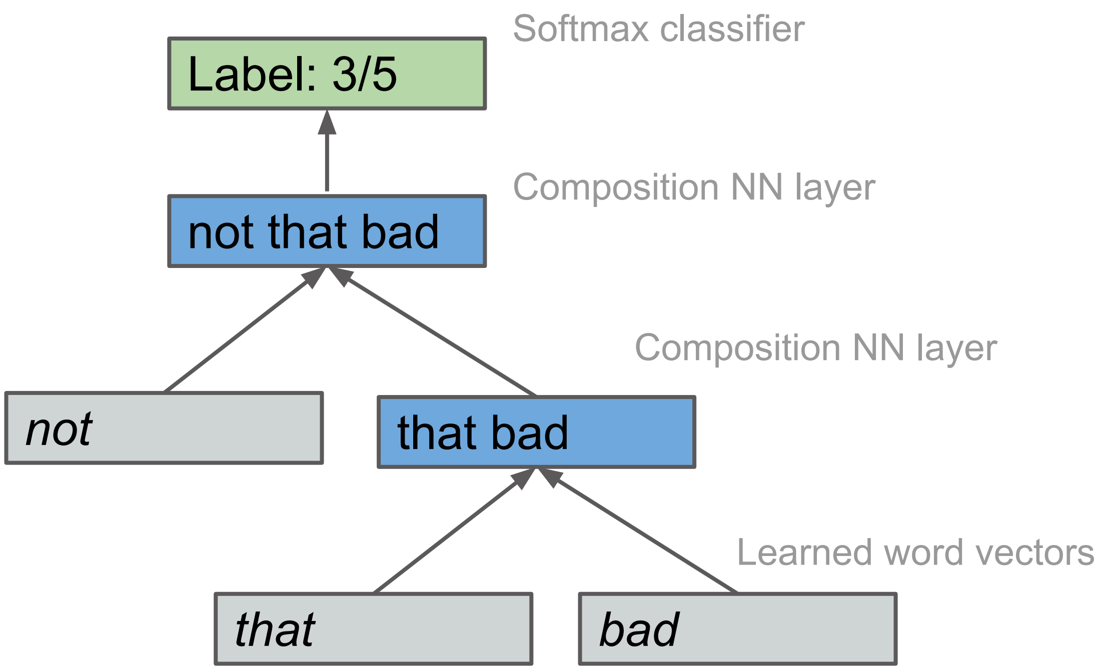
Recurrent NNs for text



- LSTM (Long Short Term Memory) models are RNNs with a more complex learned activation function meant to do a better job of preserving information across long sequences.

Hochreiter and Schmidhuber 1997

My focus: TreeRNNs*

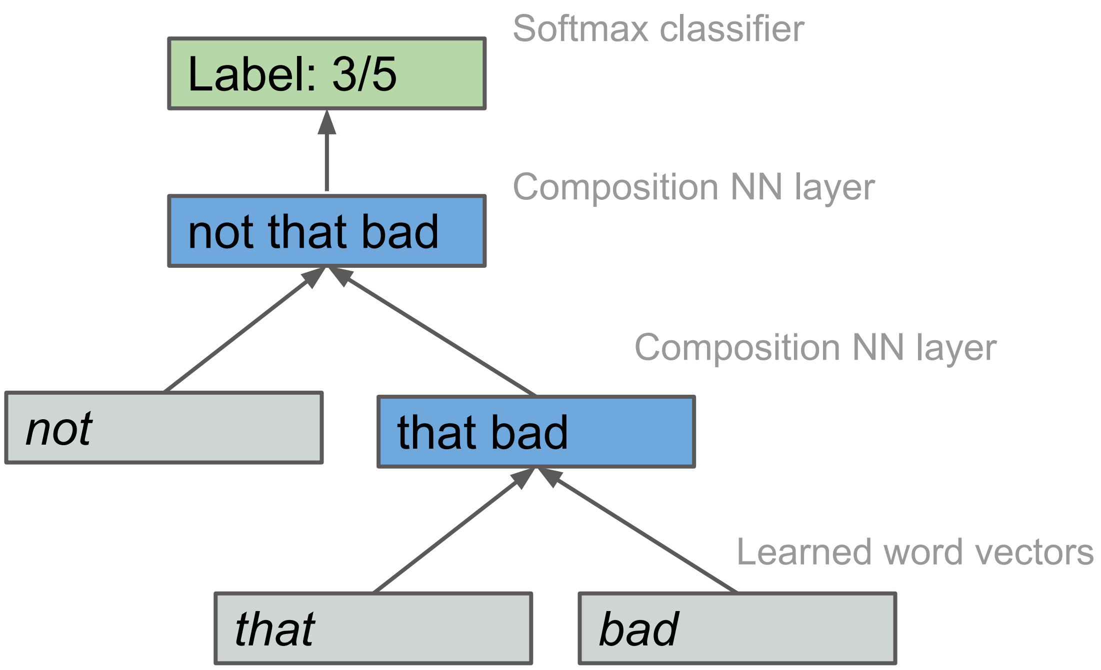


- Sequence of operations follows parse tree
- Different sentence? Different tree structure.

*RNN = **Recurrent** NN and TreeRNN = Tree-structured **Recursive** NN. Aagh.

Socher et al. 2011

My focus: TreeRNNs



- Basic TreeRNN uses the same kind of learned function as an RNN:

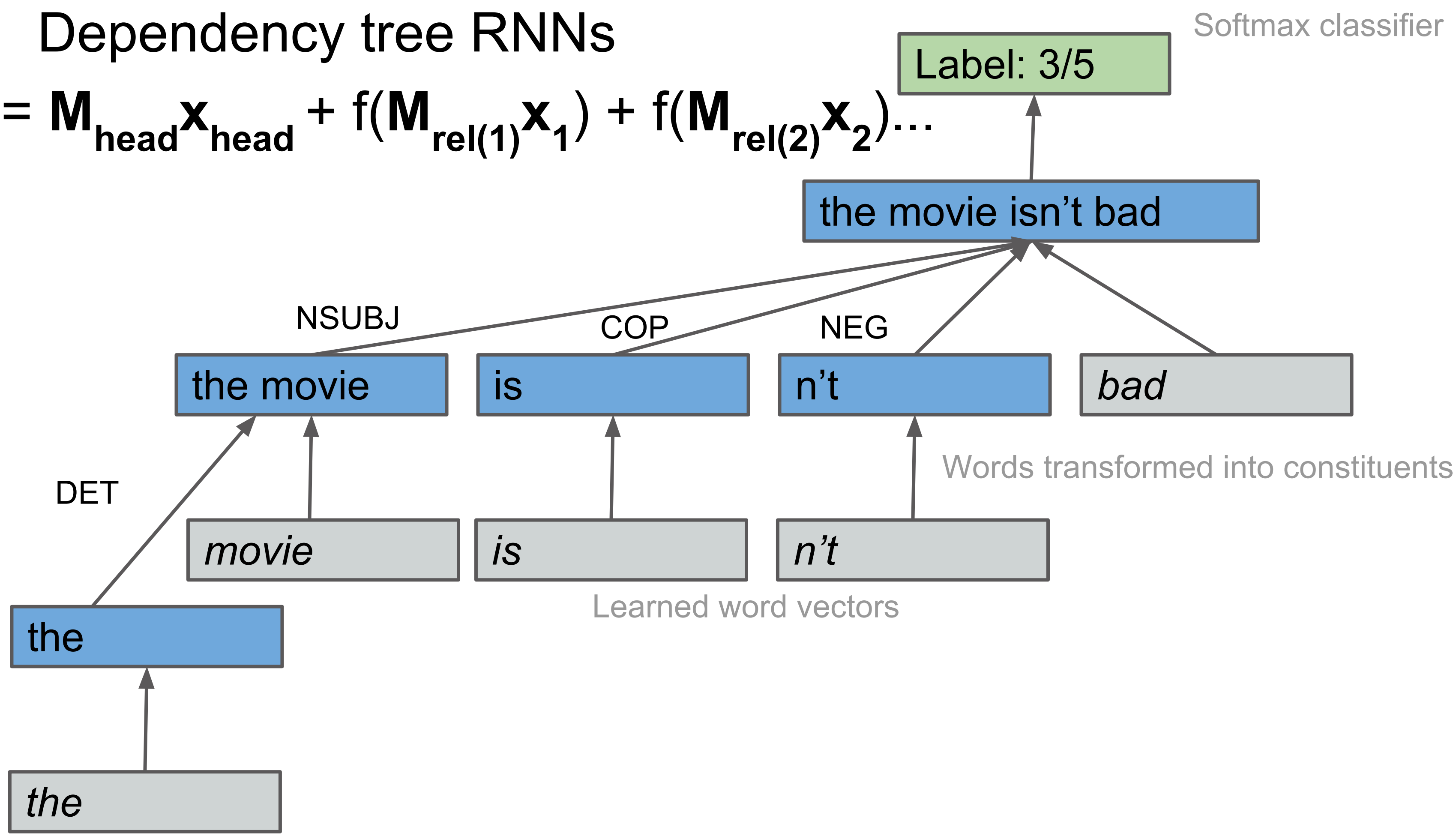
  $$\mathbf{y} = f(\mathbf{M}[\mathbf{x}_l\ \mathbf{x}_r] + \mathbf{b})$$

Socher et al. 2011

Variants: Dependency TreeRNNs

- Dependency tree RNNs

$$\mathbf{y} = \mathbf{M}_{\mathbf{head}}\mathbf{x}_{\mathbf{head}} + f(\mathbf{M}_{\mathbf{rel(1)}}\mathbf{x}_{\mathbf{1}}) + f(\mathbf{M}_{\mathbf{rel(2)}}\mathbf{x}_{\mathbf{2}})\ldots$$



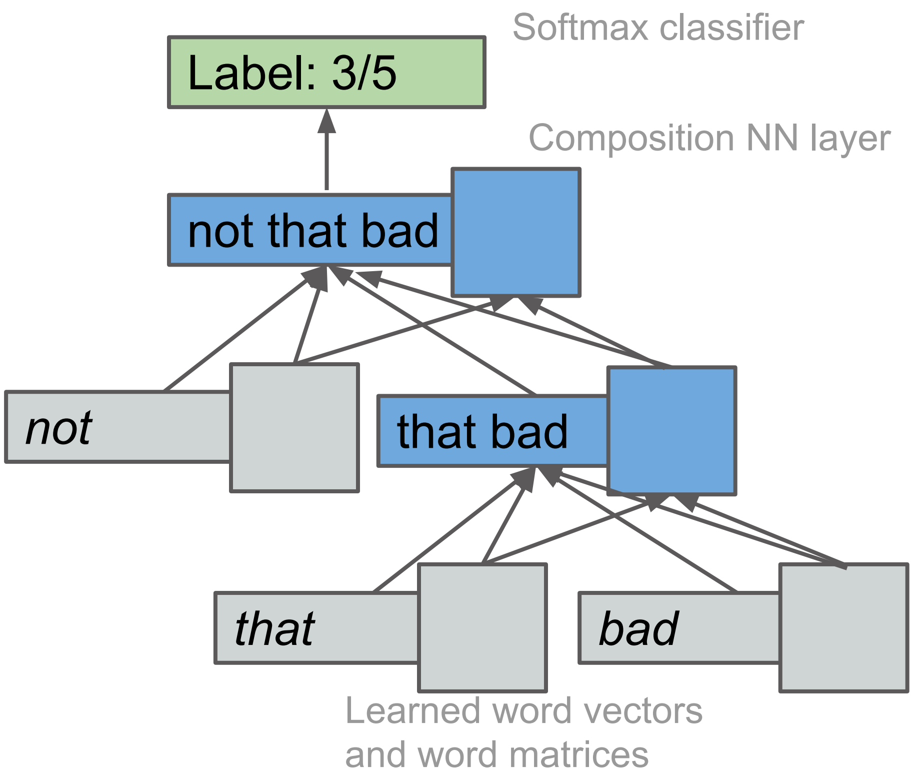
Socher et al. 2014

Variants: Matrix-vector TreeRNN

- Matrix-vector RNN composition functions:

  $\mathbf{y} = f(\mathbf{M}_v[\mathbf{Ba}\ \mathbf{Ab}])$

  $\mathbf{Y} = \mathbf{M}_m[\mathbf{A}\ \mathbf{B}]$

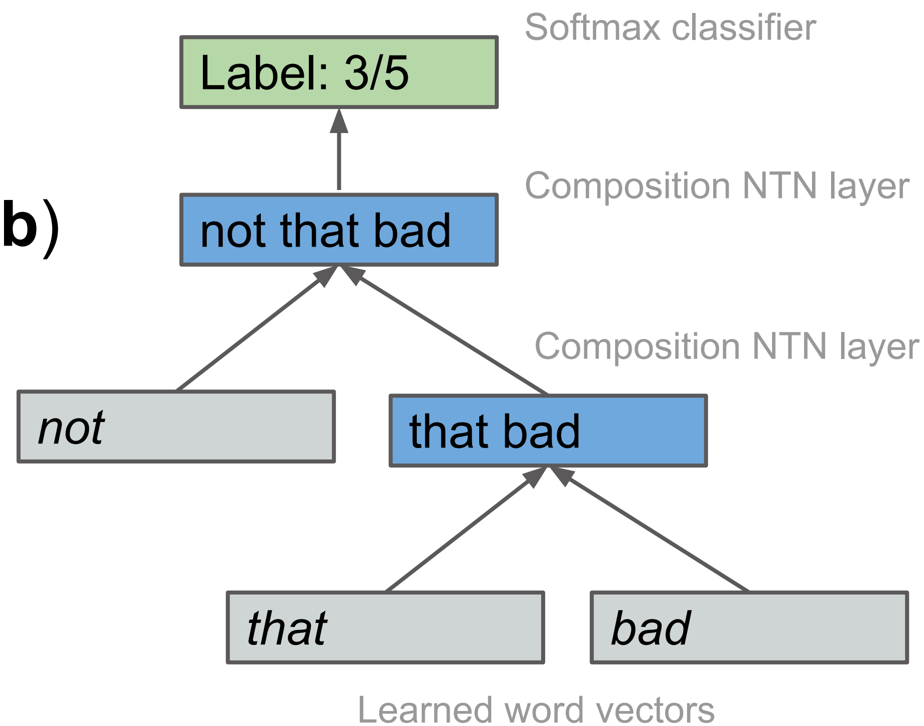


Socher et al. 2012

Variants: TreeRNTN

- Recursive neural **tensor** network composition function:

$$\mathbf{y} = f(\mathbf{x}_l \mathbf{M}^{[1\ldots N]} \mathbf{x}_r + \mathbf{M}[\mathbf{x}_l\ \mathbf{x}_r] + \mathbf{b})$$



Chen et al. 2013, Socher et al. 2013

Other NNs for sentence meaning

And more:

- Tree autoencoders (Socher et al 2011)
- TreeLSTMs (Tai et al 2015)
- Convolutional NNs for text (Kalchbrenner et al. 2014)
- Character-level convolution (Zhang and LeCun 2015)

  ...

The big question

How well are supervised neural network models able to learn representations of sentence meaning?

The big question

How well are supervised neural network models able to learn representations of sentence meaning?

The big question

How well are supervised neural network models able to learn representations of sentence meaning?

Don't ask what meanings are. Ask what they do, and find something that does that.

-David Lewis, paraphrased

The task: Natural language inference

James Byron Dean refused to move without blue jeans

{**entails**, contradicts, neither}

James Dean didn't dance without pants

The task: Natural language inference

Claim: Simple task to define, but engages the full complexity of compositional semantics:

- Lexical entailment
- Quantification
- Coreference
- Lexical/scope ambiguity
- Commonsense knowledge
- Propositional attitudes
- Modality
- Factivity and implicativity

...

Lexical relations

Experimental approach: Train on relational statements generated from some formal system, test on other such relational statements.

The model needs to:

- Learn the relations between individual words. (lexical relations)

Formulating a learning problem

Training data:

dance	***entails***	*move*
waltz	***neutral***	*tango*
tango	***entails***	*dance*
sleep	***contradicts***	*dance*
waltz	***entails***	*dance*

Memorization (training set):

dance ***???*** *move*

waltz ***???*** *tango*

Generalization (test set):

sleep ***???*** *waltz*

tango ***???*** *move*

MacCartney's natural logic

An implementable logic for natural language inference without logical forms. (MacCartney and Manning '09)

- Sound logical interpretation (Icard and Moss '13)

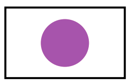
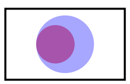
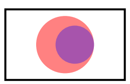
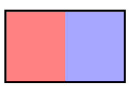
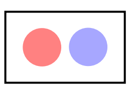
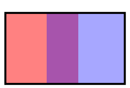
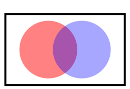
P	*James Dean*	*refused to*			*move*	*without*	*blue*	*jeans*
H	*James Byron Dean*		*did*	*n't*	*dance*	*without*		*pants*
edit index	1	2	3	4	5	6	7	8
edit type	SUB	DEL	INS	INS	SUB	MAT	DEL	SUB
lex feats	strsim= 0.67	implic: –/o	cat:aux	cat:neg	hypo			hyper
lex entrel	=	|	=	^	⊐	=	⊏	⊏
projec-tivity	↑	↑	↑	↑	↓	↓	↑	↑
atomic entrel	=	|	=	^	⊏	=	⊏	⊏

inversion

Natural logic: relations

Seven possible relations between phrases/sentences:

Slide from Bill MacCartney

	$x \equiv y$	equivalence	*couch* ≡ *sofa*
	$x \sqsubset y$	forward entailment (strict)	*crow* ⊏ *bird*
	$x \sqsupset y$	reverse entailment (strict)	*European* ⊐ *French*
	$x \wedge y$	negation (exhaustive exclusion)	*human* ^ *nonhuman*
	$x \mid y$	alternation (non-exhaustive exclusion)	*cat* | *dog*
	$x \smile y$	cover (exhaustive non-exclusion)	*animal* ‿ *nonhuman*
	$x \# y$	independence	*hungry* # *hippo*

Natural logic: relation joins

	≡	⊏	⊐	^	|	‿	#
≡	≡	⊏	⊐	^	|	‿	#
⊏	⊏	⊏	·	|	|	·	·
⊐	⊐	·	⊐	‿	·	‿	·
^	^	‿	|	≡	⊐	⊏	#
|	|	·	|	⊏	·	⊏	·
‿	‿	‿	·	⊐	⊐	·	·
#	#	·	·	#	·	·	·

MacCartney's join table: $a\ R\ b \wedge b\ R'\ c \vdash a\ \{R \bowtie R'\}\ c$

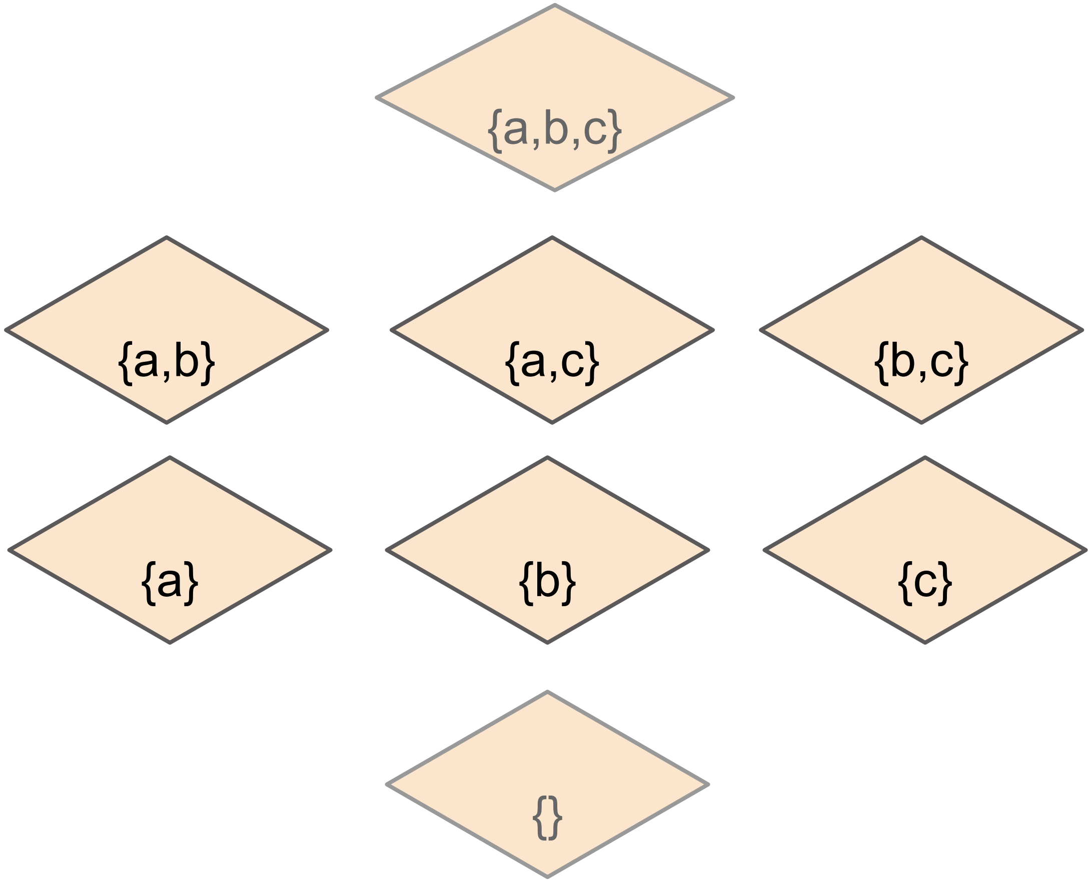
{animal ⊐ cat, cat ⊐ kitten} ⊢ animal ⊐ kitten
{cat ⊏ animal, animal ^ non-animal} ⊢ cat | non-animal

Natural logic: relation joins

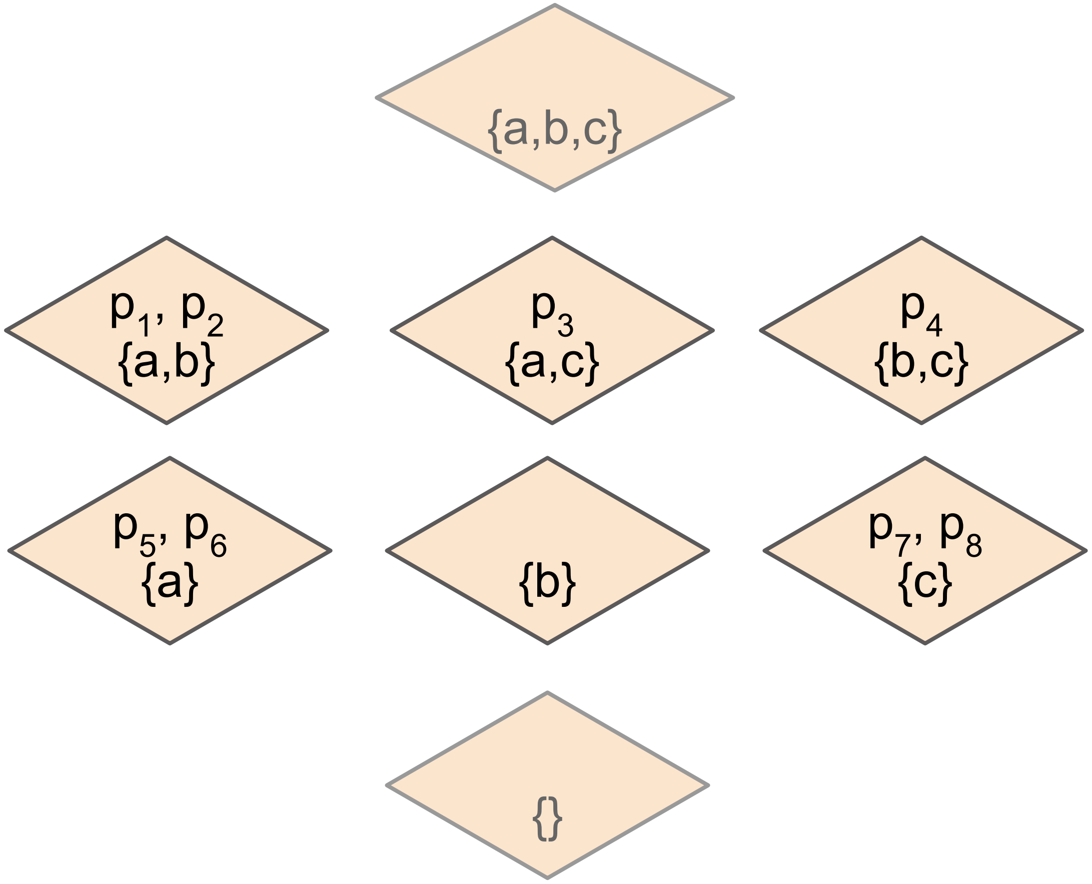
	$\equiv$	$\sqsubset$	$\sqsupset$	$\wedge$	$\mid$	$\smile$	$\#$
$\equiv$	$\equiv$	$\sqsubset$	$\sqsupset$	$\wedge$	$\mid$	$\smile$	$\#$
$\sqsubset$	$\sqsubset$	$\sqsubset$	$\cdot$	$\mid$	$\mid$	$\cdot$	$\cdot$
$\sqsupset$	$\sqsupset$	$\cdot$	$\sqsupset$	$\smile$	$\cdot$	$\smile$	$\cdot$
$\wedge$	$\wedge$	$\smile$	$\mid$	$\equiv$	$\sqsupset$	$\sqsubset$	$\#$
$\mid$	$\mid$	$\cdot$	$\mid$	$\sqsubset$	$\cdot$	$\sqsubset$	$\cdot$
$\smile$	$\smile$	$\smile$	$\cdot$	$\sqsupset$	$\sqsupset$	$\cdot$	$\cdot$
$\#$	$\#$	$\cdot$	$\cdot$	$\#$	$\cdot$	$\cdot$	$\cdot$

Can our NNs learn to make these inferences over pairs of embedding vectors?

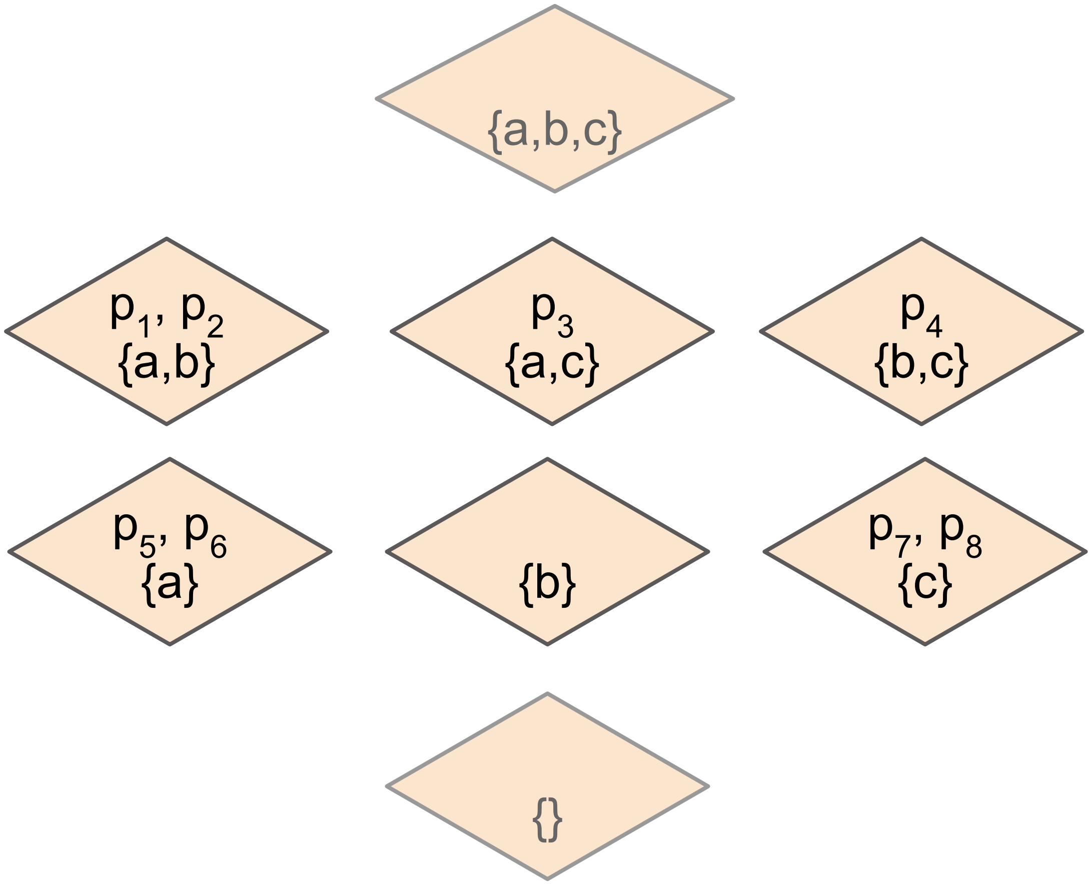
Lexical relations: data generation



Lexical relations: data generation



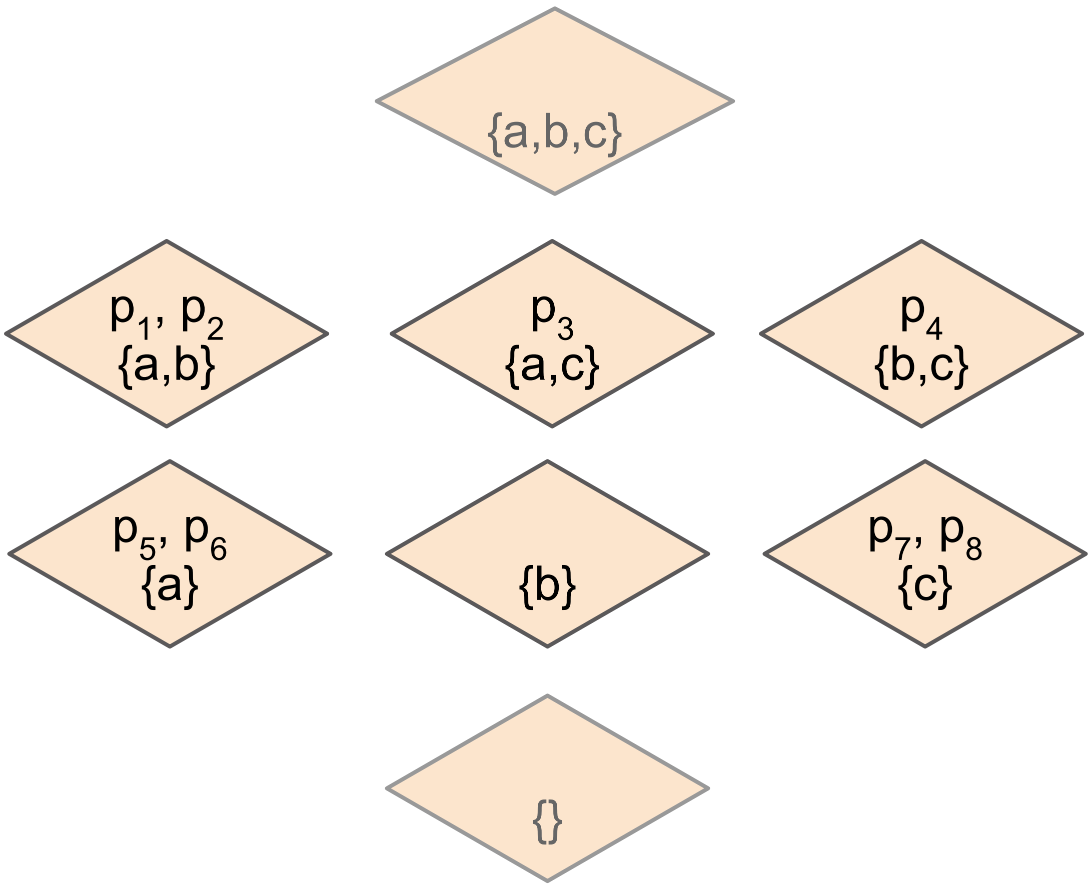
Lexical relations: data generation



Extracted relations:

$p_1 \equiv p_2$
$p_1 \wedge p_7$
$p_1 \sqsupset p_5$
$p_4 \sqsupset p_8$
$p_2 \sqsupset p_5$
$p_5 \equiv p_6$
$p_5 \mid p_7$
$p_7 \sqsubset p_4$
$p_7 \wedge p_1$
$p_8 \sqsupset p_1$

Lexical relations: data generation



Train:

$p_1 \equiv p_2$

$p_1 \sqsupset p_5$

$p_4 \sqsupset p_8$

$p_5 \mid p_7$

$p_7 \wedge p_1$

Test:

$p_1 \wedge p_7$

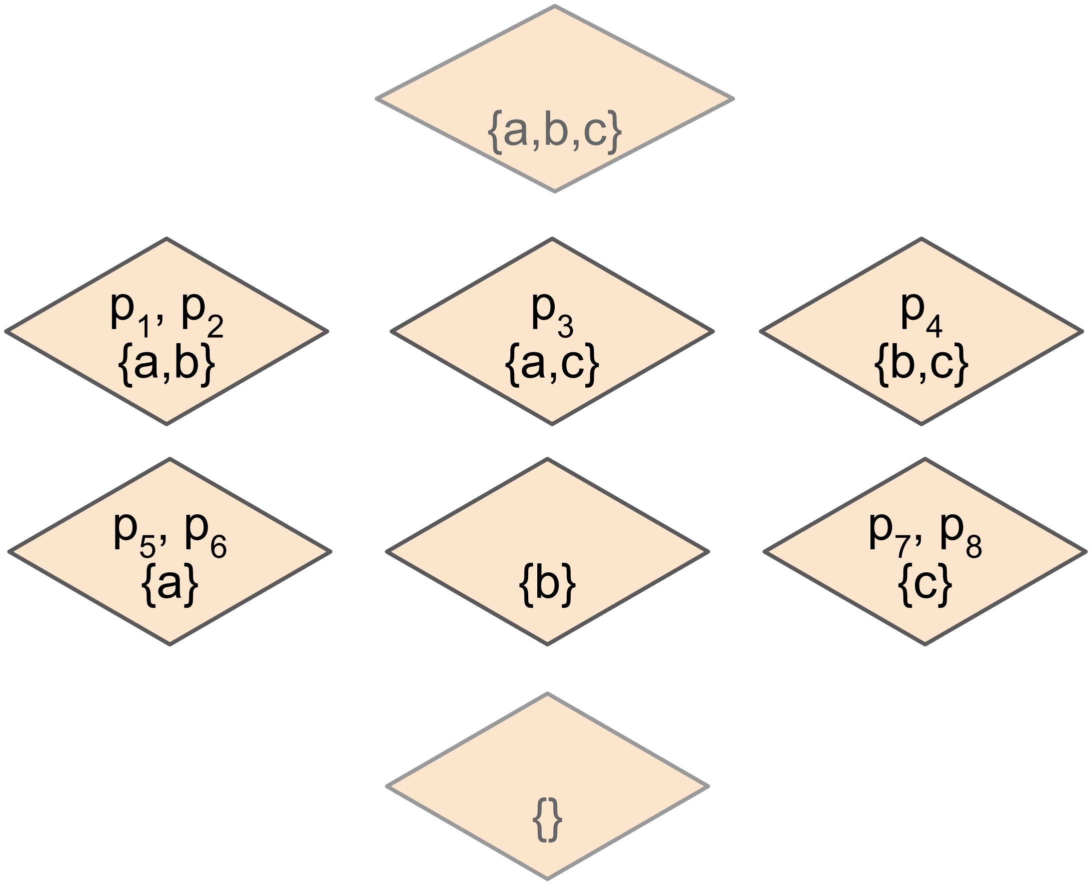
$p_2 \sqsupset p_5$

$p_5 \equiv p_6$

$p_7 \sqsubset p_4$

$p_8 \sqsupset p_4$

Lexical relations: data generation



Train:

$p_1 \equiv p_2$

$p_1 \sqsupset p_5$

$p_4 \sqsupset p_8$

$p_5 \mid p_7$

$p_7 \wedge p_1$

Test:

$p_1 \wedge p_7$

$p_2 \sqsupset p_5$

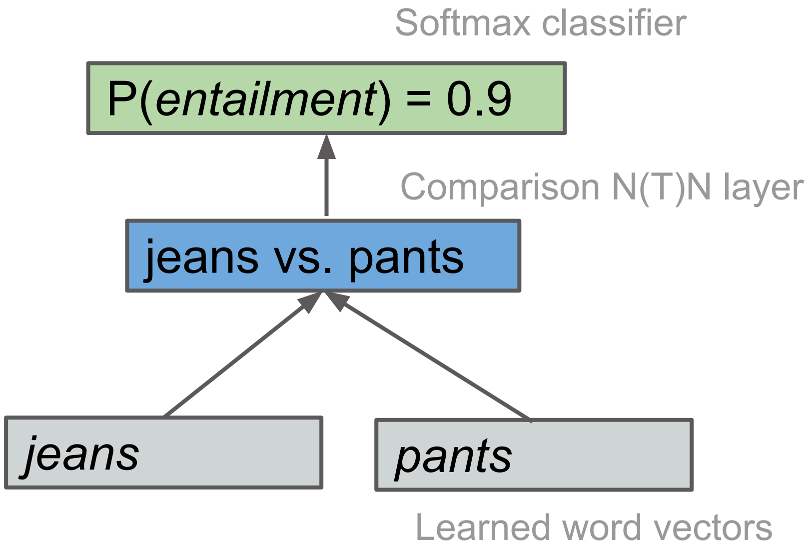
$p_5 \equiv p_6$

$p_7 \sqsubset p_4$

$p_8 \sqsupset p_4$

A minimal NN for lexical relations

- Words are learned embedding vectors.
- One plain TreeRNN or TreeRNTN layer
- Softmax emits relation labels
- Learn everything with SGD.



Lexical relations: training

- 80 random terms (p_1 - p_{80}) denoting sets
  - Sampled with replacement from the powerset of the set of 7 entities (a-g)

- 6400 statements, yielding:
  - 3200 training examples
  - about 2900 provable test examples
    (~7% not provable)

Lexical relations: results

	Train	Test
# only	53.8 (10.5)	53.8 (10.5)
15d NN	99.8 (99.0)	94.0 (87.0)
15d NTN	**100 (100)**	**99.6 (95.5)**

- Both models tuned, then trained to convergence on five randomly generated datasets
- Reported figures: % correct (macroaveraged F1)
- Both NNs used 15d embeddings, 75d comparison layer

Lexical relations: Conclusions

- Success! NTNs can learn lexical entailment networks
  - No special optimization techniques required
  - Good generalization even with small training sets

- Open questions:
  - Geometric theory of lexical relations?
  - Relationship between the number of terms and the number of dimensions in the embedding?

Recursion in propositional logic

Experimental approach: Train on relational statements generated from some formal system, test on other such relational statements.

The model needs to:

- Learn the relations between individual words. (lexical relations)

Recursion in propositional logic

Experimental approach: Train on relational statements generated from some formal system, test on other such relational statements.

The model needs to:

- Learn the relations between individual words. (lexical relations)
- Learn how lexical relations impact phrasal relations. (projectivity)

Recursion in propositional logic

Experimental approach: Train on relational statements generated from some formal system, test on other such relational statements.

The model needs to:

- Learn the relations between individual words. (lexical relations)
- Learn how lexical relations impact phrasal relations. (projectivity)
  - This needs to be recursively applicable!

$a \equiv a$, $a \wedge (\text{not } a)$, $a \equiv (\text{not } (\text{not } a))$, ...

Recursion in propositional logic

Data: randomly generated sentences with *and*, *or*, and *not*

- 6 proposition variables (a-f), at most 4 per example
- Propositions are variables over unknown truth values (2^{64} possible representations)
- Train on statements with at most 4 operators, test with more.

Formula	Interpretation
a, b, c, d, e, f	$[\![x]\!] \in \{\mathsf{T}, \mathsf{F}\}$
$\textit{not}\ \varphi$	T iff $[\![\varphi]\!] = \mathsf{F}$
$(\varphi\ \textit{and}\ \psi)$	T iff $\mathsf{F} \notin \{[\![\varphi]\!], [\![\psi]\!]\}$
$(\varphi\ \textit{or}\ \psi)$	T iff $\mathsf{T} \in \{[\![\varphi]\!], [\![\psi]\!]\}$

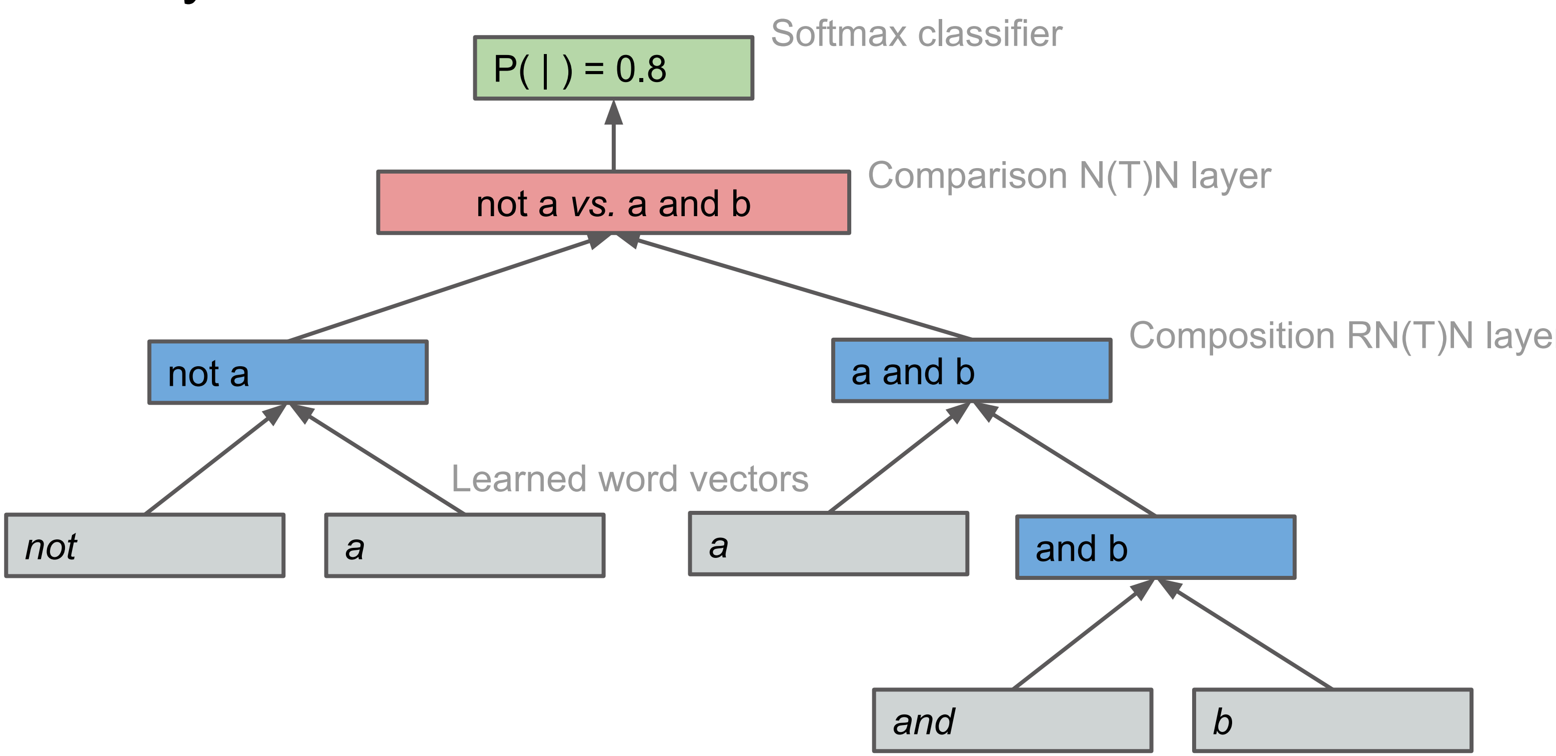
(a) Well-formed formulae. φ and ψ range over all well-formed formulae, and $[\![\cdot]\!]$ is the interpretation function mapping formulae into $\{\mathsf{T}, \mathsf{F}\}$.

$\textit{not}\ a$	$\wedge$	a
$\textit{not}\ \textit{not}\ a$	$\equiv$	a
a	$\sqsubset$	$(a\ \textit{or}\ b)$
a	$\sqsupset$	$(a\ \textit{and}\ b)$
$\textit{not}(\textit{not}\ a\ \textit{and}\ \textit{not}\ b)$	$\equiv$	$(a\ \textit{or}\ b)$

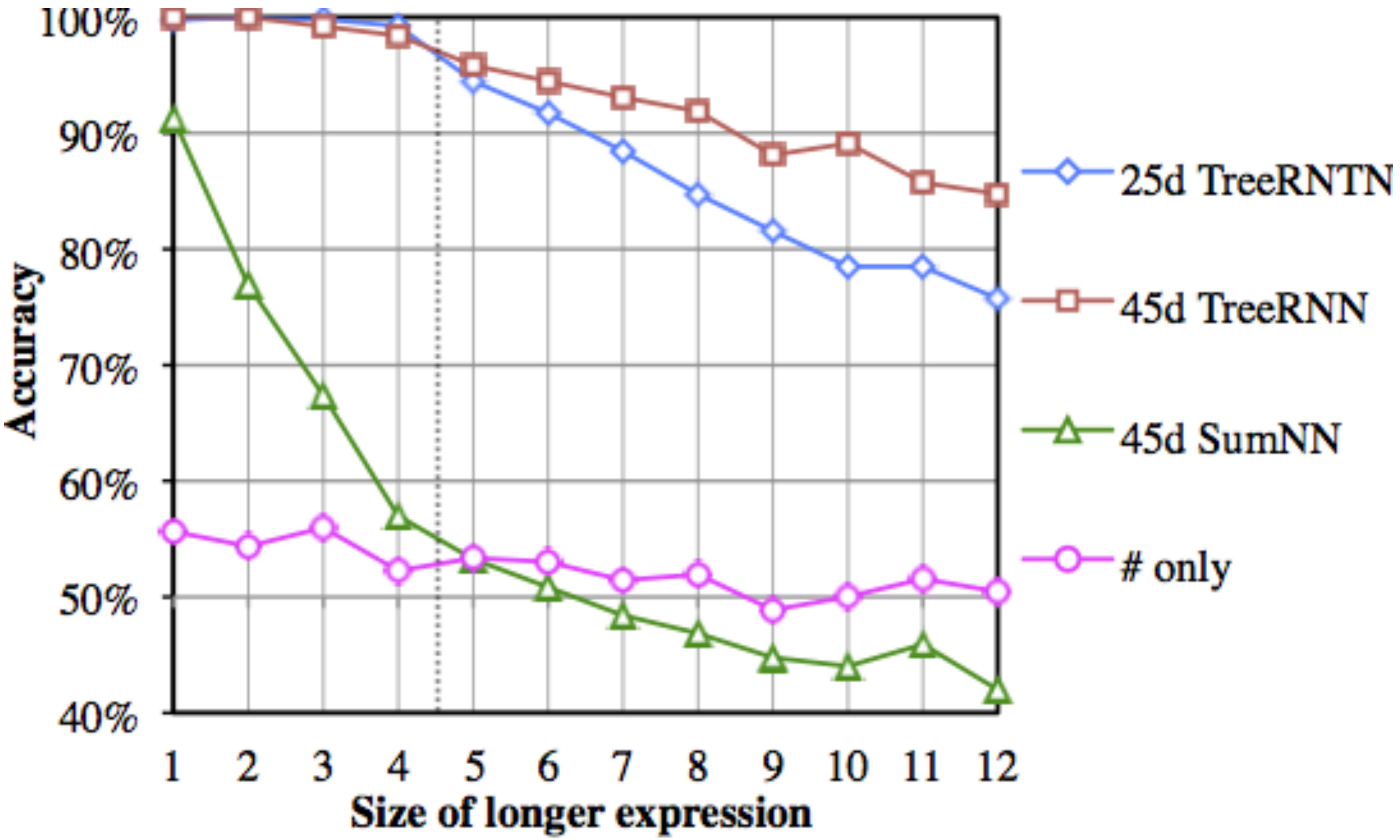
(b) Examples of statements about relations between well-formed formulae, defined in terms of sets of satisfying interpretation functions $[\![\cdot]\!]$.

NLI with TreeRNNs

- Model structure:
  - Two trees, then a separately learned comparison layer, then a classifier:



Recursion in propositional logic



Today

How do these models work?

- Survey: Deep learning models for NLU

How well can they handle anything we'd recognize as meaning?

- A measure of success: natural language inference
- Can NNs learn lexical relations?
- Can TreeRNNs learn recursive functions?
- **What about quantification and monotonicity?**
- Frontiers: What about real language?

Quantifiers

Experimental paradigm: Train on relational statements generated from some formal system, test on other such relational statements.

The model needs to:

- Learn the relations between individual words. (lexical relations)
- Learn how lexical relations impact phrasal relations. (projectivity)

Quantifiers

Experimental paradigm: Train on relational statements generated from some formal system, test on other such relational statements.

The model needs to:

- Learn the relations between individual words. (lexical relations)
- Learn how lexical relations impact phrasal relations. (projectivity)
- Quantifiers present some of the harder cases of both of these.

Two experiments

(most warthogs) walk	$\wedge$	(not-most warthogs) walk
(most mammals) move	$\#$	(not-most (not turtles)) move
(most (not pets)) (not swim)	$\sqsupset$	(not-most (not pets)) move
(no turtles) (not growl)	$\mid$	(no turtles) (not swim)
(no warthogs) swim	$\sqsupset$	(no warthogs) move
(no warthogs) move	$\sqsubset$	(no (not reptiles)) swim

Quantifiers

- Small vocabulary
  - Three basic types:
    - Quantifiers: *some*, *all*, *no*, *most*, *two*, *three, not-all, not-most, less-than-two, less-than-three*
    - Predicates: *dog, cat, mammal, animal …*
    - Negation: *not*
- 60k examples generated using a generative implementation of the relevant portion of MacCartney and Manning’s logic.
- All sentences of the form *QPP*, with optional negation on each predicate.

Quantifier results

	Train	Test
Most freq. class (# only)	35.4%	35.4%
25d SumNN (sum of words)	96.9%	93.9%
25d TreeRNN	99.6%	99.2%
25d TreeRNTN	**100%**	**99.7%**

Summary: Artificial data

- Simple NTNs can encode relation composition accurately.
- Tree structured models can learn recursive functions, and can apply them in structures that are (somewhat) larger than those seen in training.
- Tree structured models can learn to reason with quantifiers.

Do we see these behaviors in textual entailment with real natural language?

Natural language inference data

- To do NLI on real English, we need to teach an NN model English almost from scratch.
- What data do we have to work with:
  - GloVe/word2vec (useful w/ any data source)
  - SICK: Thousands of examples created by editing and pairing hundreds of sentences.
  - RTE: Hundreds of examples created by hand.
  - DenotationGraph: Millions of extremely noisy examples (~73% correct?) constructed fully automatically.

Results on SICK (+DG, +tricks) so far

	SICK Train	DG Train	Test
Most freq. class	**56.7%**	**50.0%**	**56.7%**
30 dim TreeRNN	95.4%	67.0%	74.9%
50 dim TreeRNTN	97.8%	74.0%	**76.9%**

Are we competitive? Sort of...

Best result (UIllinois)	**84.5%**
≈ interannotator agreement!	
Median submission (out of 18):	77%
Our TreeRNTN:	76.9%

We're the only purely-learned system in the competition: Everything but the parser is trained from the supplied data.

Is it realistic to learn from SICK?

A guy is mowing the lawn.

Grass is being mowed by a man.

ENTAILMENT

A guy is mowing the lawn

There is no guy mowing the lawn.

CONTRADICTION

A guy is mowing the lawn

There is no man mowing grass.

CONTRADICTION

...

Natural language inference data

- To do NLI on real English, we need to teach an NN model English almost from scratch.
- What data do we have to work with:
  - GloVe/word2vec (useful w/ any data source)
  - SICK: Thousands of examples created by editing and pairing hundreds of sentences.
  - RTE: Hundreds of examples created by hand.
  - DenotationGraph: Millions of extremely noisy examples (~73% correct?) constructed fully automatically.

Natural language inference data

- To do NLI on real English, we need to teach an NN model English almost from scratch.
- What data do we have to work with:
  - GloVe/word2vec (useful w/ any data source)
  - SICK: Thousands of examples created by editing and pairing hundreds of sentences.
  - RTE: Hundreds of examples created by hand.
  - DenotationGraph: Millions of extremely noisy examples (~73% correct?) constructed fully automatically.
  - Stanford NLI corpus: ~600k examples, written by Turkers.

The Stanford NLI corpus

Instructions

The Stanford University NLP Group is collecting data for use in research on computer understanding of English. We appreciate your help!

We will show you the caption for a photo. We will not show you the photo. Using only the caption and what you know about the world:

- Write one alternate caption that is **definitely** a **true** description of the photo.
- Write one alternate caption that **might be** a **true** description of the photo.
- Write one alternate caption that is **definitely** an **false** description of the photo.

Photo caption **A little boy in an apron helps his mother cook.**

Definitely correct Example: For the caption *"Two dogs are running through a field."* you could write *"There are animals outdoors."*

Write a sentence that follows from the given caption.

Maybe correct Example: For the caption *"Two dogs are running through a field."* you could write *"Some puppies are running to catch a stick."*

Write a sentence which may be true given the caption, and may not be.

Definitely incorrect Example: For the caption *"Two dogs are running through a field."* you could write *"The pets are sitting on a couch."*

Write a sentence which contradicts the caption.

Problems (optional) *If something is wrong with the caption that makes it difficult to understand, do your best above and let us know here.*

Some examples

A young boy rides a bike down a snow covered road.

A child is outside.

ENTAILMENT

2 female babies eating chips.

Two female babies are enjoying chips.

NEUTRAL

A woman in an apron shopping at a market.

A woman in an apron is preparing dinner.

CONTRADICTION

Results?

Not much yet:

- Train on SICK + DG, test on SICK: So-so
- Train on SNLI: Stay tuned!

Interested in being one of the first to work on this? The draft corpus is available to the class.

Deep learning for text: Logistics

- Lots of knobs to twiddle:
  - Optimization method (plain SGD, AdaGrad, ...)
  - Dimensionality
  - Initialization, regularization
  - Type of layer function/nonlinearity
  - Train/test split

    ...
- Good references for general NN methods (though 'standard' methods change often):
  - An incomplete book from the Bengio lab: http://www.iro.umontreal.ca/~bengioy/dlbook/
  - Coursera lectures from Geoff Hinton: https://www.coursera.org/course/neuralnets

Deep learning for text: Logistics

- Typical training times for models like the ones seen here: 4-48h
- No standard deep learning library yet can do everything you'll want for language.
  - CAFFE (Python), Theano (Python), Torch (Lua) all very strong for at least some model types.
  - Try my MATLAB codebase for an easy start with:
    - RNN, LSTM
    - TreeRNN, TreeRNTN, TreeLSTM

Thanks!

More questions?

sbowman@stanford.edu