Models for natural language inference

Christopher Potts and Bill MacCartney
CS224u, Stanford, Spring 2016

Plan and goals for today

1. More details on NLI as a task
2. Strategies for defining NLI features
3. Break to define some features ourselves
4. Neural architectures for NLI
5. A few notes on assignment 4 and NLI projects

Simple examples

	Premise	Relation	Hypothesis
1.	turtle	**contradicts**	linguist
2.	A turtle danced.	**entails**	A turtle moved.
3.	Every reptile danced.	**entails**	Every turtle moved.
4.	Some turtles walk.	**contradicts**	No turtles move.
5.	James Byron Dean refused to move without blue jeans.	**entails**	James Dean didn't dance without pants.
6.	Mitsubishi Motors Corp's new vehicle sales in the US fell 46 percent in June.	**contradicts**	Mitsubishi's sales rose 46 percent.
7.	Acme Corporation reported that its CEO resigned.	**entails**	Acme's CEO resigned.

NLI task formulation

Does the premise justify an inference to the hypothesis?

- Commonsense reasoning, rather than strict logic.
- Focus on local inference steps, rather than long deductive chains.
- Emphasis on variability of linguistic expression.

Perspectives:

- Zaenen, Karttunen, Crouch (2005): Local textual inference: can it be defined or circumscribed?
- Manning (2006): Local textual inference: it's hard to circumscribe, but you know it when you see it – and NLP needs it.
- Crouch, Karttunen, Zaenen (2006): Circumscribing is not excluding: a reply to Manning.

Connections to other tasks

Dagan et al. (2006), ‘The PASCAL Recognizing Textual Entailment Task’

“The Recognizing Textual Entailment (RTE) Challenge is an attempt to promote an abstract generic task that captures major semantic inference needs across applications.

[...]

It seems that major inferences, as needed by multiple applications, can indeed be cast in terms of textual entailment.”

Connections to other tasks (Dagan et al. 2006)

Question Answering: Given a question (premise), identify a text that entails an answer (hypothesis).

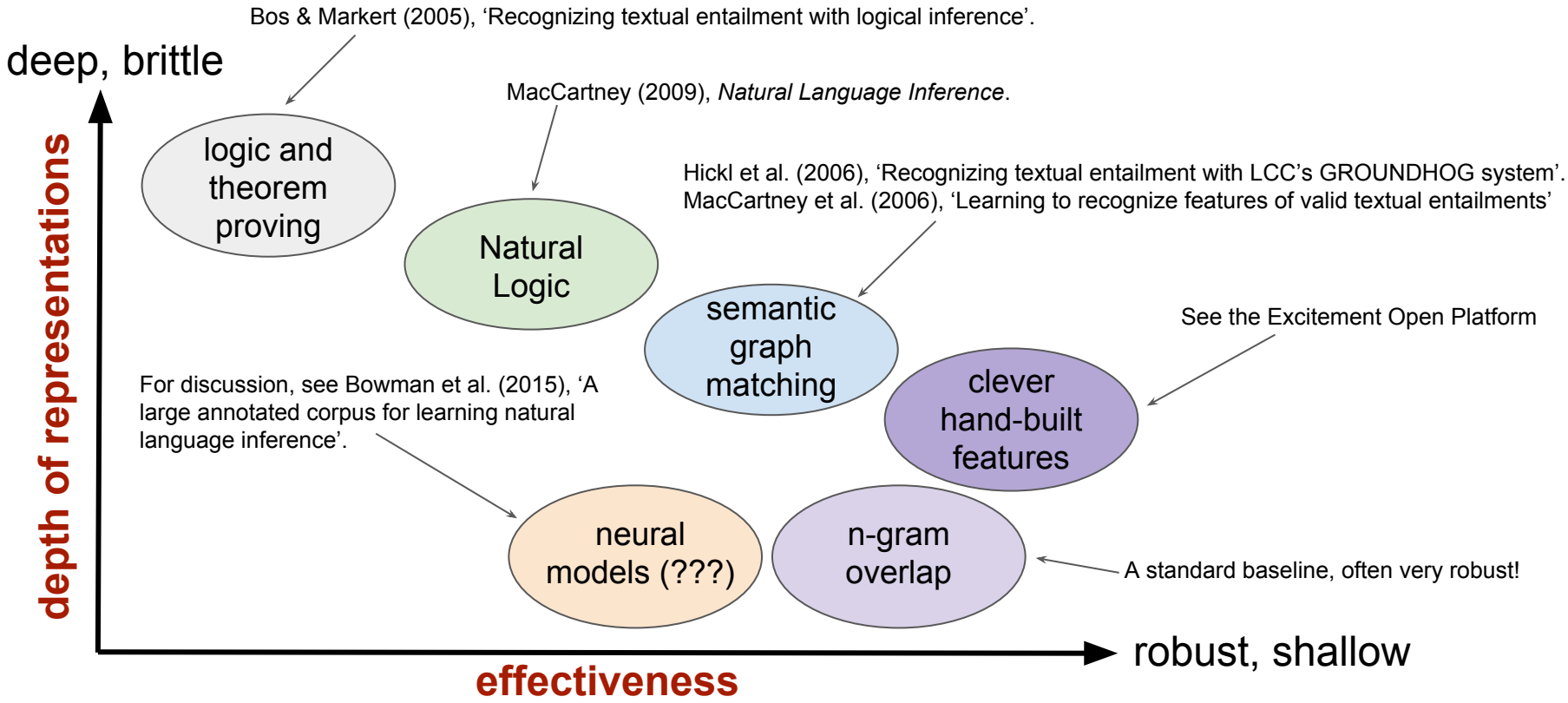
Information Retrieval: Given a query (hypothesis), identify texts that entail that query (premises).

Summarization: Given a text (premise) *T*, create or identify a text that *T* entails.

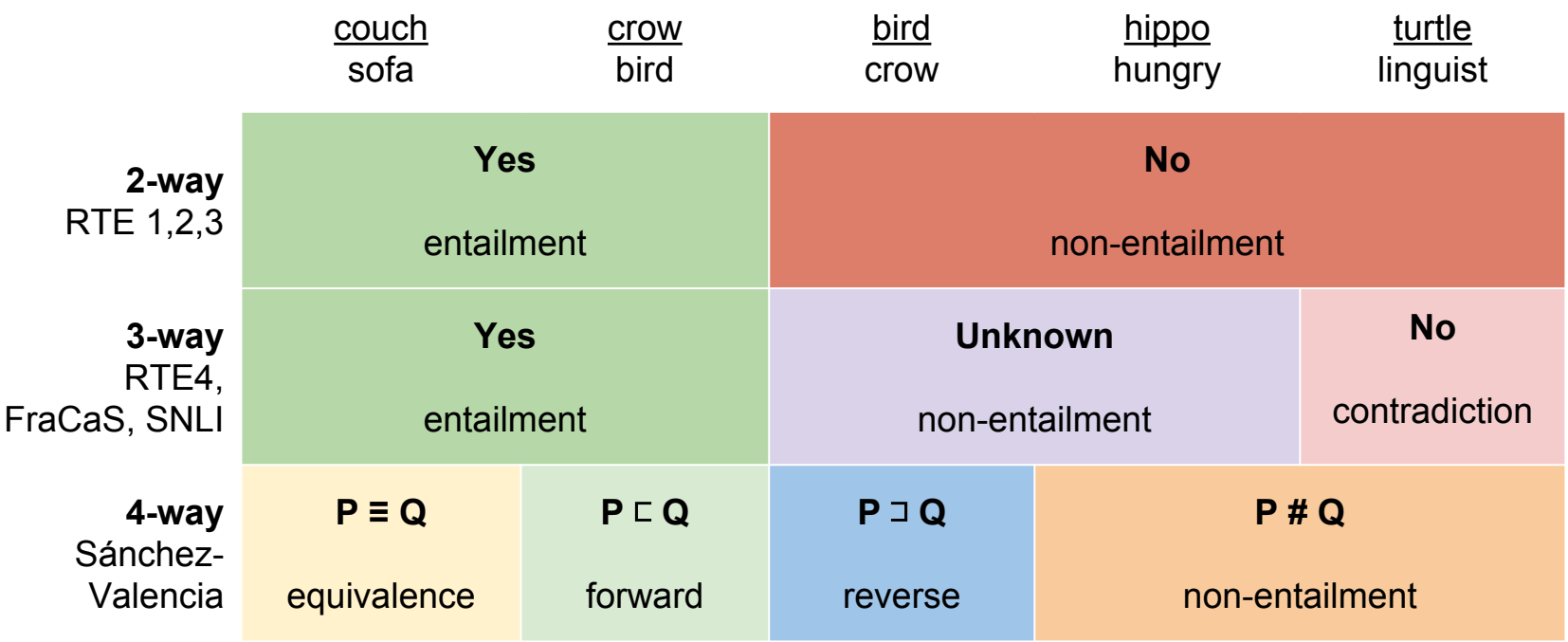
Summarization: Omit sentences that are entailed by others.

Machine translation: Mutual entailment between texts in different languages.

Models for NLI

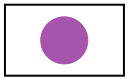
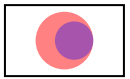
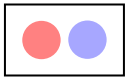
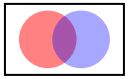


Labels


couch
sofa
crow
bird
bird
crow
hippo
hungry
turtle
linguist
2-way
RTE 1,2,3
Yes
entailment
No
non-entailment
3-way
RTE4,
FraCaS, SNLI
Yes
entailment
Unknown
non-entailment
No
contradiction
4-way
Sánchez-
Valencia
P ≡ Q
equivalence
P ⊏ Q
forward
P ⊐ Q
reverse
P # Q
non-entailment

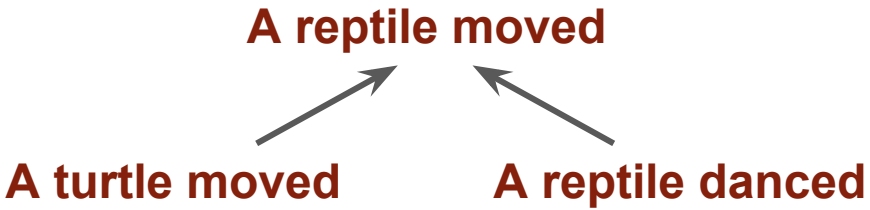
NatLog inference relations (MacCartney)

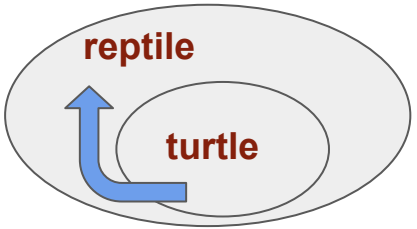
The seven elementary, mutually exclusive, non-vacuous set relations:

	X ≡ Y	equivalence	couch ≡ sofa
	X ⊏ Y	forward entailment	crow ⊏ bird
	X ⊐ Y	reverse entailment	European ⊐ French
	X ^ Y	negation	human ^ non-human
	X | Y	alternation	cat | dog
	X ‿ Y	cover	animal ‿ non-human
	X # Y	independence	hungry # hippo

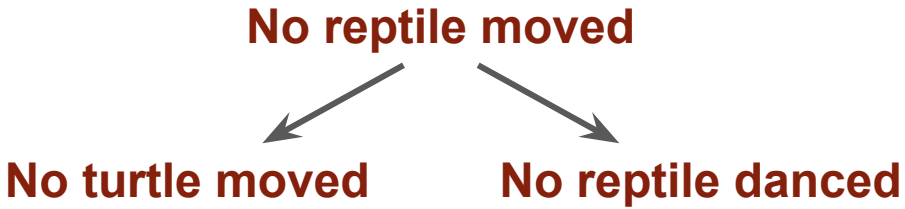
Monotonicity

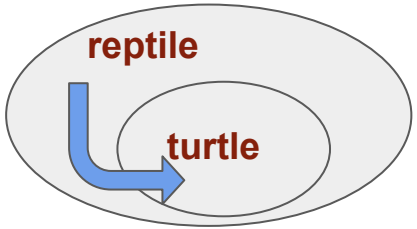
Upward monotone: preserve entailments from **subsets** to **supersets**:





Downward monotone: preserve entailments from **supersets** to **subsets**:





Non-monotone: do not preserve entailment in either direction.

Downward monotonicity in language

- Negations (e.g., *not, n't, never, no, nothing, nowhere, none, neither*)
- The **first** argument of *every* (*turtle* in *every turtle danced*)
- Determiners like *at most, few, fewer/less than*
- Conditional antecedents (*if*-clauses)
- Negative implicatives (e.g., *forget to, refuse to, hesitate to*)
- Negative attitude verbs like *doubt* and *deny* (at least approximately)
- Adverbs like *rarely* and *hardly*

Upward monotonicity in language

- Upward monotonicity is sort of the default for lexical items
- Most determiners (e.g., *a, some, at least, more than*)
- The **second** argument of *every* (*danced* in *every turtle danced*)
- Positive implicatives (e.g., *manage to, succeed to, force to*)

Monotonicity features

- Edits that broaden/weaken preserve forward entailment:
  - Deleting modifiers
  - Changing specific terms to more general ones.
  - Dropping conjuncts, adding disjuncts.
- Edits that narrow/strengthen do not preserve forward entailment:
  - Adding modifiers
  - Changing general terms to specific ones.
  - Adding conjuncts, dropping disjuncts.
- In downward monotone environments, the above are **reversed**.

Projectivity signature for negation (HW4, problem 3)

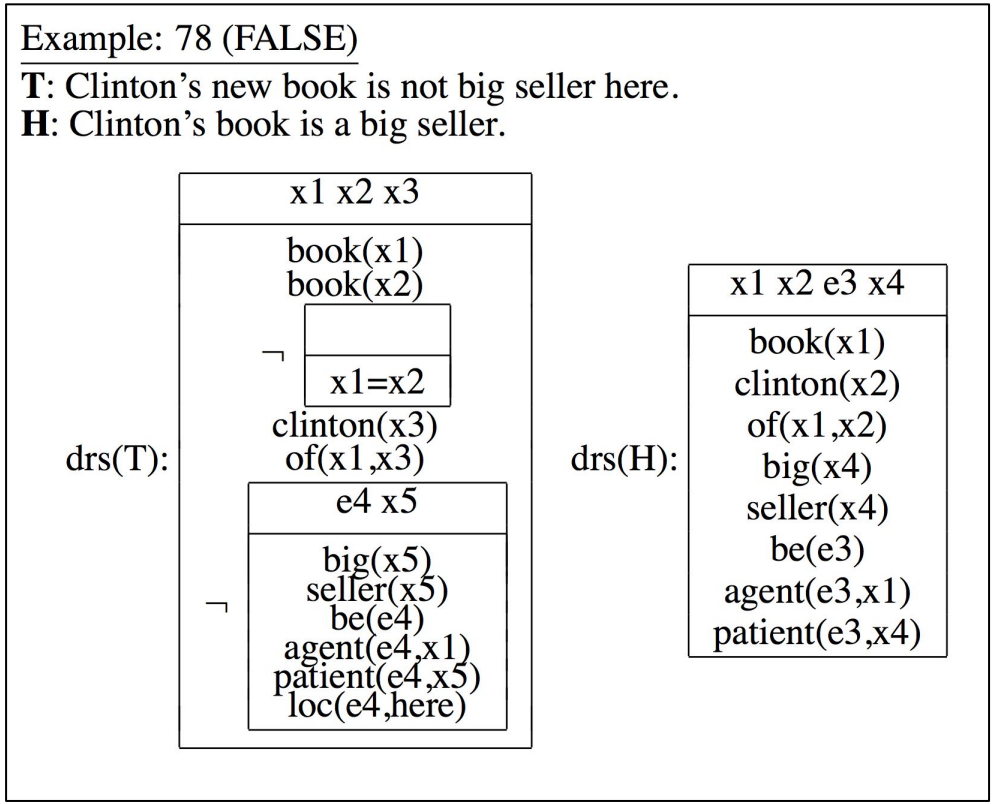
X | Y	not-X # not-Y	X ⊏ not-Y	not-X ⊐ Y
X = Y	not-X = not-Y	X | not-Y	not-X | Y
X # Y	not-X # not-Y	X # not-Y	not-X # Y
X ⊏ Y	not-X ⊐ not-Y	X | not-Y	not-X # Y
X ⊐ Y	not-X ⊏ not-Y	X # not-Y	not-X | Y

For more projectivity signatures (which suggest features): MacCartney and Manning (2010), ‘An extended model of natural logic’.

NLI datasets (slide from Sam Bowman)

Corpus	Complete Sentences	Human Labeled	Size (num. pairs)
FraCaS	✓	✓	**.3k**
RTE 1-5	✓	✓	**7k**
SICK	✓	✓	**10k**
SNLI	✓	✓	**570k**
DenotationGraph	✗	✗	**728k**
Levy Graphs	✗	✗	**1,500k**
PPDB 2.0	✗	✗	**100,000k**

Hand-built, logic-driven systems



Bos and Markert (2005),
'Recognising Textual Entailment with Logical Inference'

Linear classifiers for NLI

- Formulation as a standard supervised learning task
- Generally large, sparse features spaces
- Standard baselines: n-gram overlap, n-gram cross-product
- Systems tend to draw on rich external lexical resources
- Features approximate important aspects of semantic composition, including the monotonicity and projectivity facts discussed above
- Popular, powerful open-source system: Excitement Open Platform

Linear classifiers: hands-on exploration

- Notebook is nli.ipynb
- Make sure you have the nli-data distribution.
- Load in everything up to `linear_classifier_experiment`

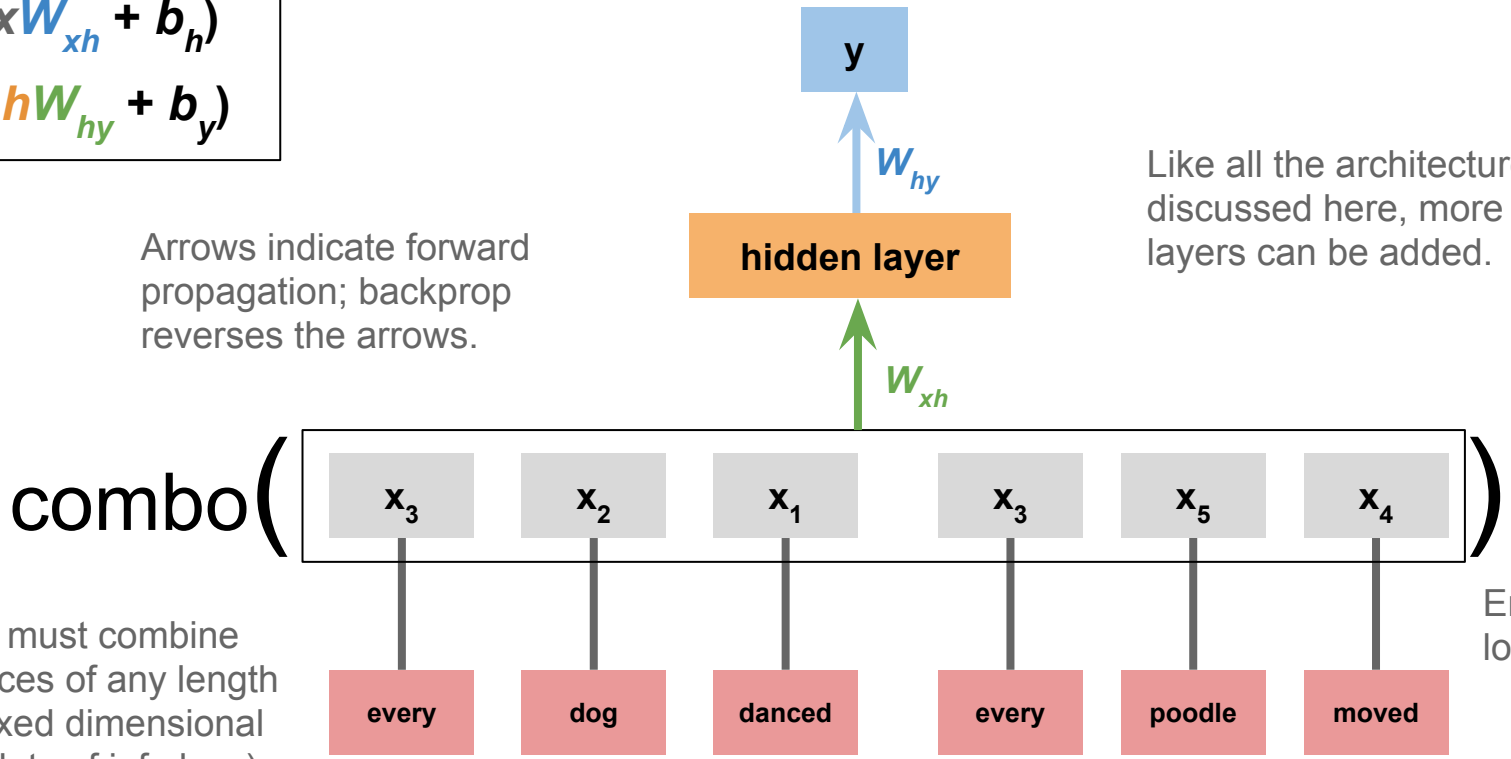
```
def word_cross_product_phi(t1, t2):
    words1 = t1.leaves()
    words2 = t2.leaves()
    feat_dict = Counter([(w1, w2)
                         for w1, w2 in product(words1, words2)])
    return feat_dict

linear_classifier_experiment(phi=word_cross_product_phi)
```

Deep classifiers for NLI

- Only recently became competitive on this task, due to a lack of data
- SNLI enabled proper training of neural models
- For NLI, neural models are arguably currently superior to standard linear classifiers with cool features, but I expect more jockeying for the lead between these two communities
- Tasks like NLI offer some interesting design options to explore:
    - How to represent each sentence?
    - How to represent the relationship between the sentences?

Feed-forward architecture

$$h = f(xW_{xh} + b_h)$$
$$y = g(hW_{hy} + b_y)$$



Arrows indicate forward propagation; backprop reverses the arrows.

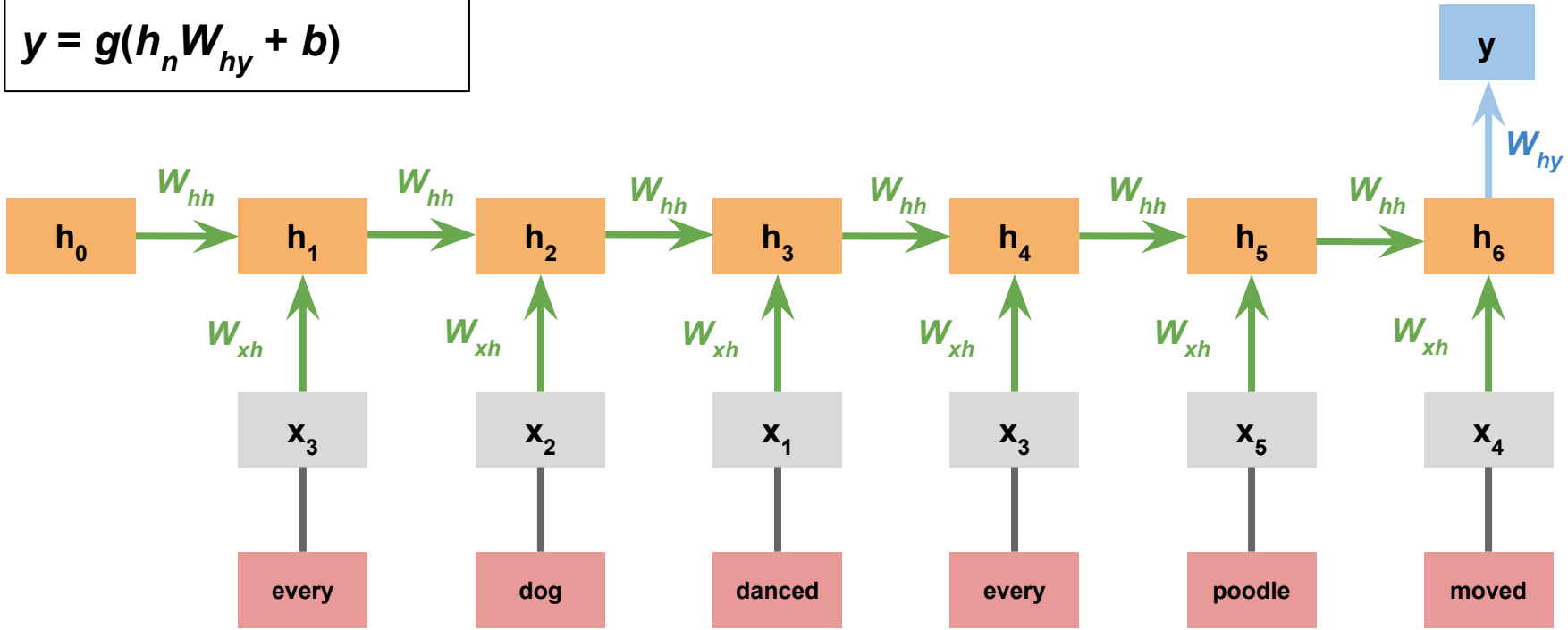
Like all the architectures discussed here, more hidden layers can be added.

Embedding look-up

combo must combine sequences of any length into a fixed dimensional vector (lots of info loss)

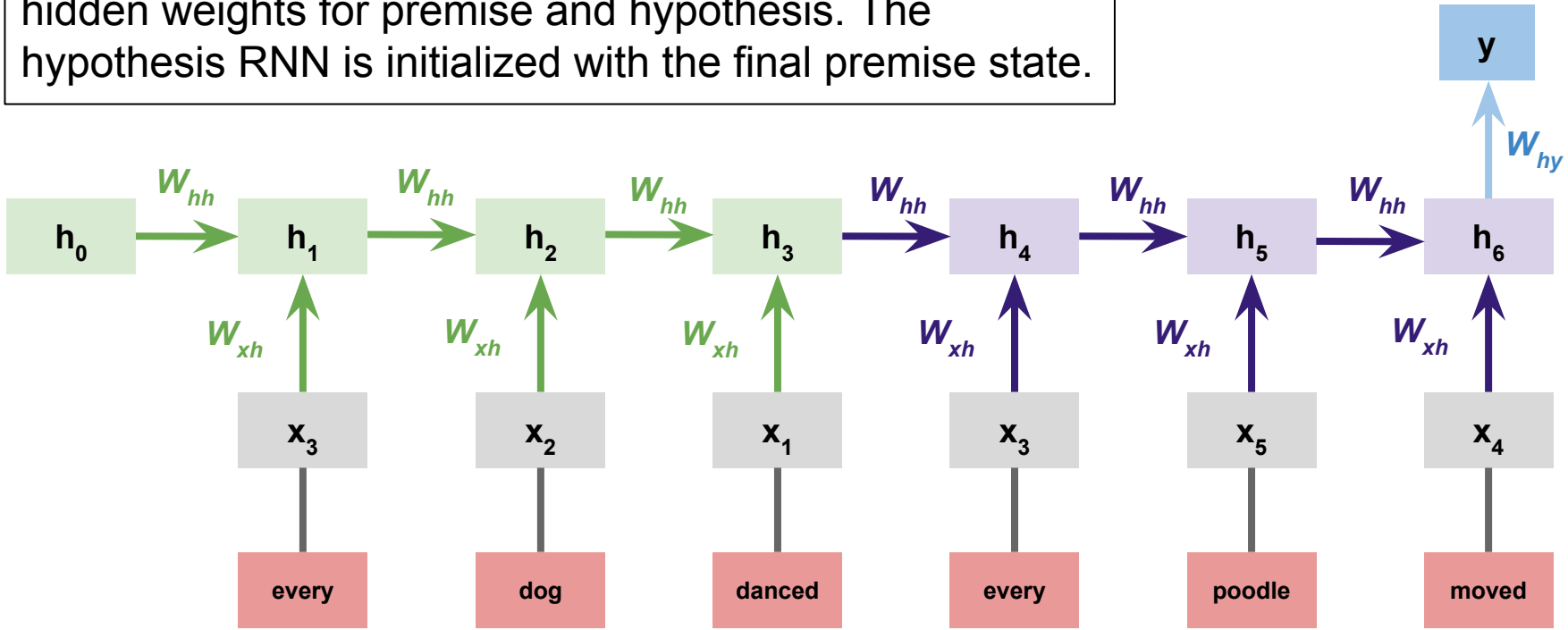
Recurrent architectures: simple classifiers

$$h_t = f(x_t W_{xh} + h_{t-1} W_{hh})$$
$$y = g(h_n W_{hy} + b)$$



Recurrent architectures: chained

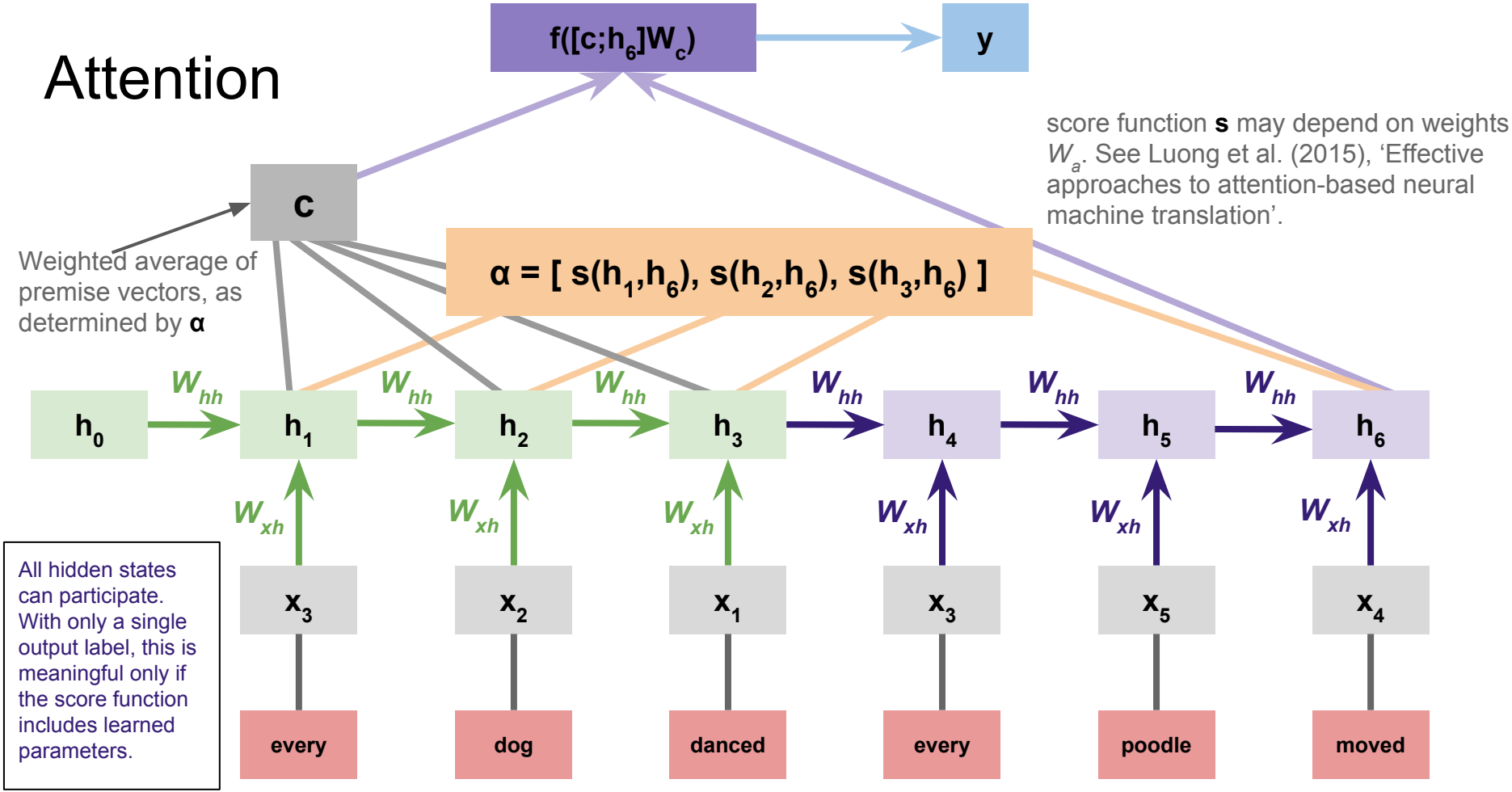
Same as the classifier RNN, but separate input and hidden weights for premise and hypothesis. The hypothesis RNN is initialized with the final premise state.



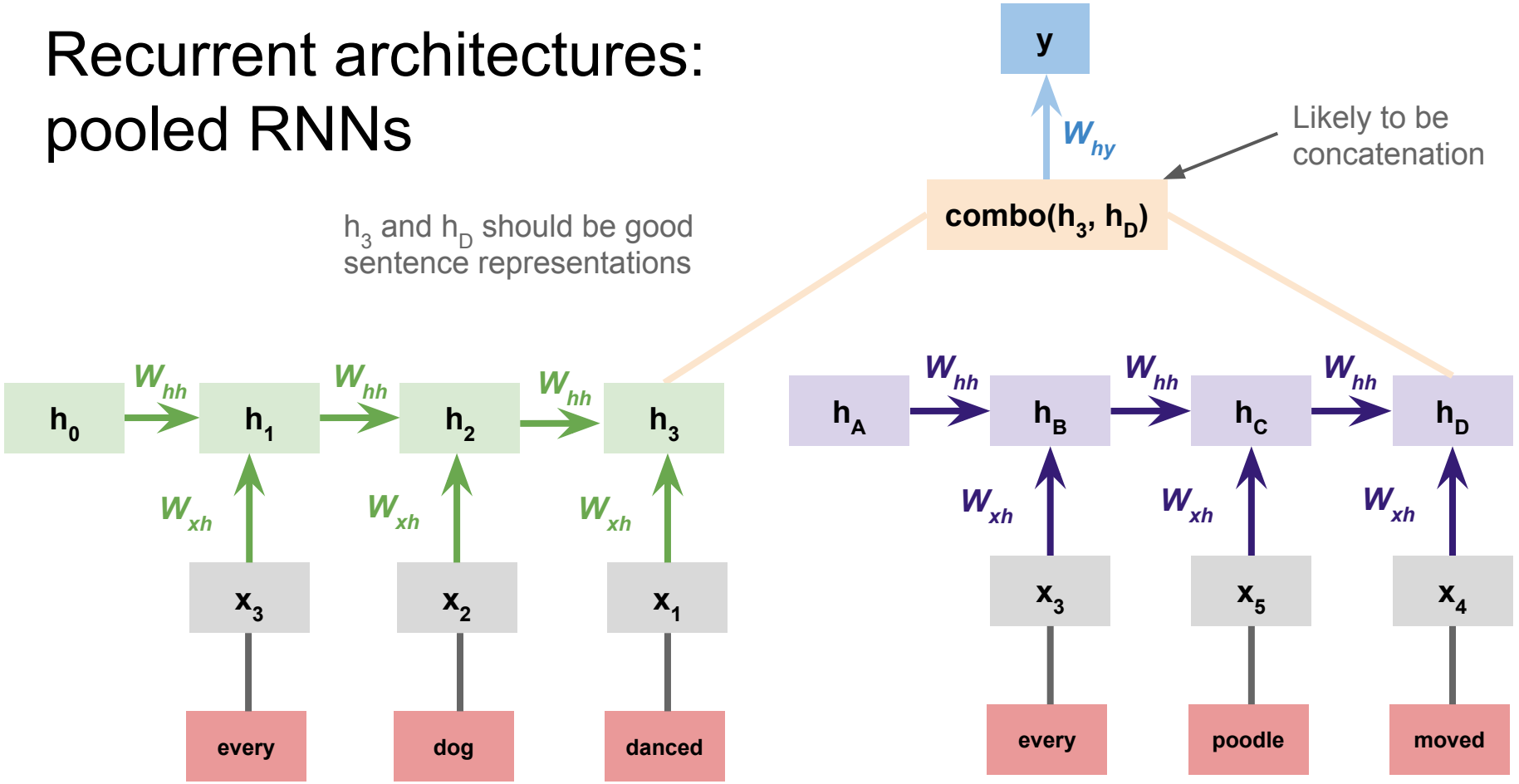
Gated cells

- The standard RNN formulation can easily lead to exploding or vanishing gradients for long sequences. (With our code, you might experience this even with our short-sentence fragment of SNLI.)
- Long Short-Term Memory (LSTM) and Gated Recurrent Unit (CRU) cells address this by introducing controlled notions of memory.
- The memory cells preserve gradient information, and the gating functions control the degree to which the current input affects the memory.
- Tree RNNs also have gradient-size problems, and they too can benefit from gated cells (Tai et al. (2015), ‘Improved semantic representations from tree-structured long short-term memory networks’).

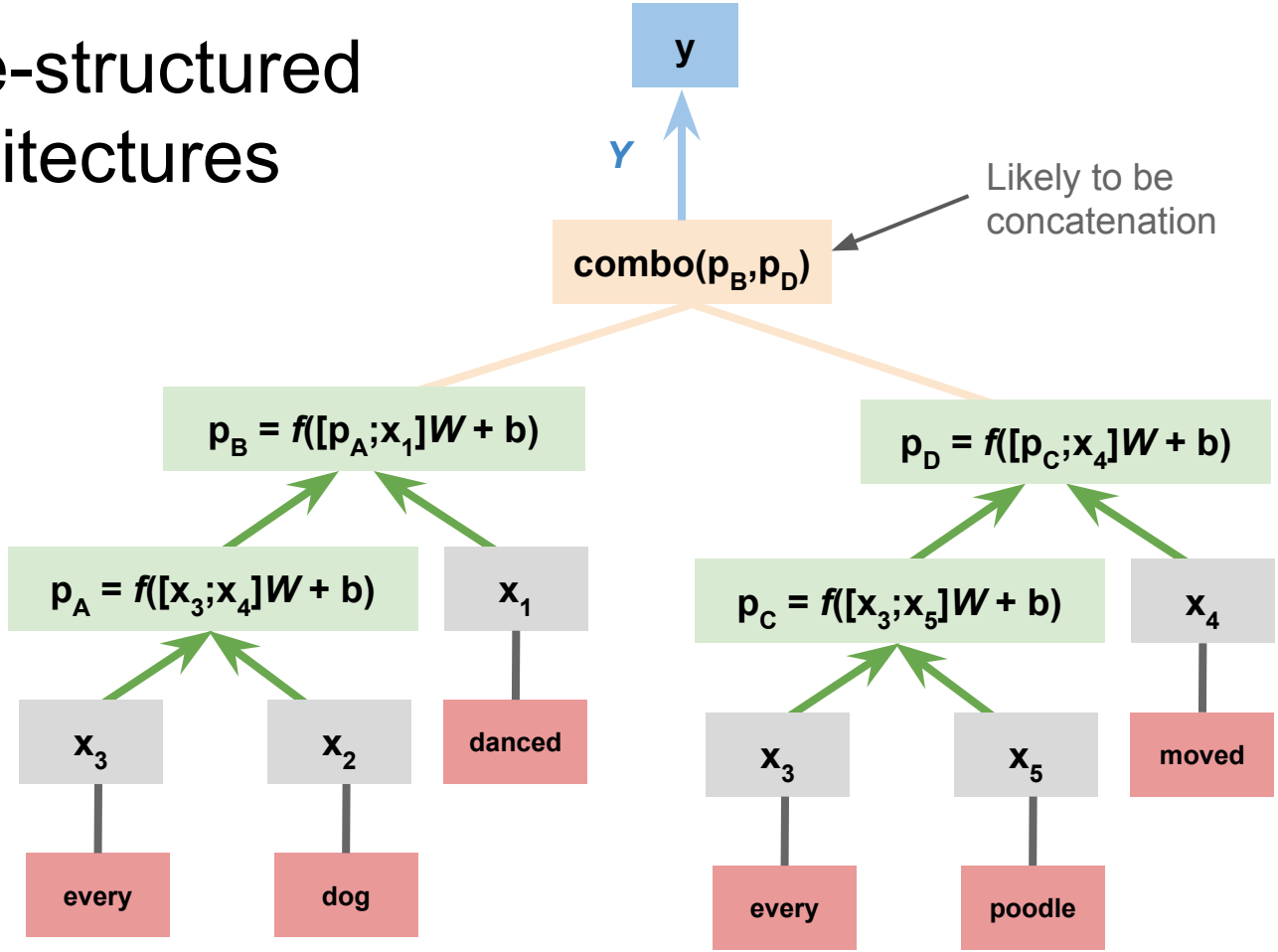
Attention


f([c;h6]Wc)
y
c
Weighted average of premise vectors, as determined by α
α = [s(h1,h6), s(h2,h6), s(h3,h6)]
score function s may depend on weights Wa. See Luong et al. (2015), 'Effective approaches to attention-based neural machine translation'.
h0
h1
h2
h3
h4
h5
h6
Whh
Wxh
x3
x2
x1
x3
x5
x4
every
dog
danced
every
poodle
moved
All hidden states can participate. With only a single output label, this is meaningful only if the score function includes learned parameters.

Recurrent architectures: pooled RNNs



Tree-structured architectures



Leaf nodes are looked up in the embedding.

Other neural architectures for NLI

Publication	Model	Train (% acc)	Test (% acc)
Feature-based models			
Bowman et al. '15	Unlexicalized features	49.4	50.4
Bowman et al. '15	+ Unigram and bigram features	99.7	*78.2*
Sentence encoding-based models			
Bowman et al. '15	100D LSTM encoders (221k params)	84.8	77.6
Bowman et al. '16	300D LSTM encoders (3.0m params)	83.9	80.6
Vendrov et al. '15	1024D GRU encoders w/ unsupervised 'skip-thoughts' pre-training (15m params)	98.8	81.4
Mou et al. '15	300D Tree-based CNN encoders (3.5m params)	83.3	82.1
Bowman et al. '16	300D SPINN-NP encoders (3.7m params)	89.2	*83.2*
Other neural network models			
Rocktäschel et al. '15	100D LSTMs w/ word-by-word attention (252k params)	85.3	83.5
Wang & Jiang '15	300D mLSTM word-by-word attention model (1.9m params)	92.0	86.1
Cheng et al. '16	300D LSTMN with deep attention fusion (1.4m params)	92.1	**89.0**

From http://nlp.stanford.edu/projects/snli/

Additional resources

- The Deep Learning course happening in parallel to ours (CS224d) has excellent code, videos, and slides: http://cs224d.stanford.edu/
- Goldberg (2015), 'A primer on neural network models for natural language processing'
- Karpathy (2015), 'The unreasonable effectiveness of recurrent neural networks'
- Denny Britz (2015), 'Recurrent neural networks tutorial'
- More NLI resources listed in our NLI notebook

Sam Bowman’s project ideas

Extending Bowman, Gauthier, et al. (2016), ‘A fast unified model for parsing and sentence understanding’ (the SPINN paper):

- Can any model do as well as attention-based models on short sentence pair classification without access to intermediate sentence part representations?
- Can neural attention models be made any more effective by having them attend over TreeRNN nodes instead of RNN states?
- Can we build a model that learns to parse and to use those parses to guide semantics interpretation (as in SPINN), where the semantic/text classification objective guides the choice of parse?

A few notes on assignment 4

Question 1: Using WordNet to improve linear classifiers. The goal is to encourage you to figure out to connect with WordNet.

Question 2: Pretrained inputs for the classifier RNN. The goal is to get you thinking about network initialization as an important part of deep learning.

Question 3: Learning negation. The goal is to get you to consider the nature of generalization and the role simulated data might play in evaluating deep learning models. (Also, the logic of negation has intrinsic interest!)

[Homework link]