Natural Language Inference

Christopher Potts

Stanford Linguistics

CS 224U: Natural language understanding

Overview

Associated materials

1. Code
   a. `nli.py`
   b. `nli_01_task_and_data.ipynb`
   c. `nli_02_models.ipynb`

2. Homework 4 and bake-off 4: `hw4_wordentail.ipynb`

3. Core readings: Bowman et al. 2015a; Rocktäschel et al. 2016

4. Auxiliary readings: Goldberg 2015; Dagan et al. 2006; MacCartney & Manning 2008; Williams et al. 2018

Simple examples

Premise	Relation	Hypothesis
turtle	contradicts	linguist
A turtle danced.	entails	A turtle moved.
Every reptile danced.	neutral	A turtle ate.
Some turtles walk.	contradicts	No turtles move.
James Byron Dean refused to move without blue jeans.	entails	James Dean didn't dance without pants.
Mitsubishi Motors Corp's new vehicle sales in the US fell 46 percent in June.	contradicts	Mitsubishi's sales rose 46 percent.
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 et al. (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 et al. (2006): Circumscribing is not excluding: a reply to Manning.

Connections to other tasks

Dagan et al. (2006)

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

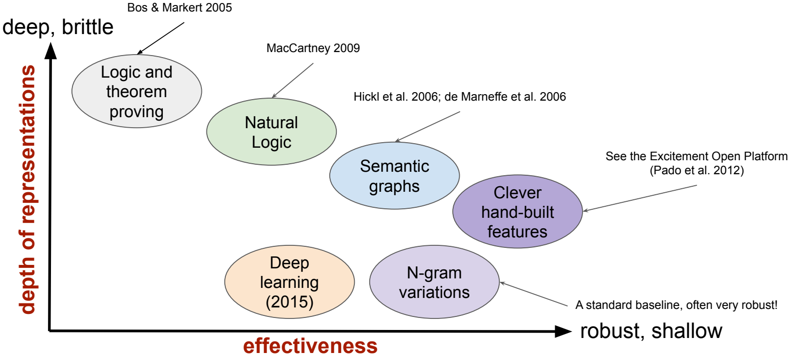
[. . .]

Consequently, we hypothesize that textual entailment recognition is a suitable generic task for evaluating and comparing applied semantic inference models. Eventually, such efforts can promote the development of entailment recognition "engines" which may provide useful generic modules across applications.

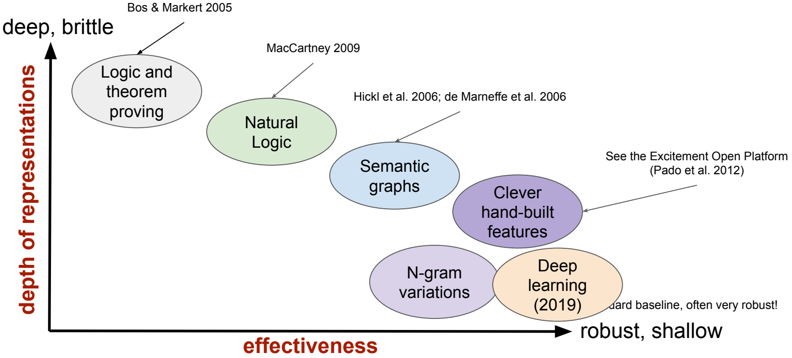
Connections to other tasks

Task	NLI framing
Paraphrase	text $\equiv$ paraphrase
Summarization	text $\sqsubset$ summary
Information retrieval	query $\sqsupset$ document
Question answering	question $\sqsupset$ answer
	Who left? $\Rightarrow$ *Someone left*
	Someone left $\sqsupset$ *Sandy left*

Models for NLI



Models for NLI



Other NLI datasets

- The FraCaS textual inference test suite
  https://nlp.stanford.edu/~wcmac/downloads/
- SemEval 2013
  https://www.cs.york.ac.uk/semeval-2013/
- SemEval 2014: Sentences Involving Compositional Knowledge (SICK)
  http://alt.qcri.org/semeval2014/task1/index.php?id=data-and-tools
- MedNLI (derived from MIMIC III)
  https://physionet.org/physiotools/mimic-code/mednli/
- XNLI is a multilingual NLI dataset derived from MultiNLI
  https://github.com/facebookresearch/XNLI
- Diverse Natural Language Inference Collection (DNC)
  http://decomp.io/projects/diverse-natural-language-inference/
- SciTail (derived from science exam questions and Web text)
  http://data.allenai.org/scitail/
- Related: 30M Factoid Question-Answer Corpus
  http://agarciaduran.org/
- Related:The Penn Paraphrase Database
  http://paraphrase.org/
- The GLUE benchmark (diverse tasks including NLI)
  https://gluebenchmark.com

Label sets

<table>
<tr><th></th><th><u>couch</u>
sofa</th><th><u>crow</u>
bird</th><th><u>bird</u>
crow</th><th><u>hippo</u>
hungry</th><th><u>turtle</u>
linguist</th></tr>
<tr><td>2-way
RTE 1,2,3</td><td colspan="2">Yes
entailment</td><td colspan="3">No
non-entailment</td></tr>
<tr><td>3-way
RTE4,
FraCaS, *NLI</td><td colspan="2">Yes
entailment</td><td colspan="2">Unknown
non-entailment</td><td>No
contradiction</td></tr>
<tr><td>4-way
Sánchez-
Valencia</td><td>$P \equiv Q$
equivalence</td><td>$P \sqsubset Q$
forward</td><td>$P \sqsupset Q$
reverse</td><td colspan="2">$P \# Q$
non-entailment</td></tr>
</table>

Hypothesis-only baselines

- In his project for this course (2016), Leonid Keselman observed that hypothesis-only models are strong.
- Other groups have since further supported this (Poliak et al. 2018; Gururangan et al. 2018; Tsuchiya 2018)
- Why does it hold? We can trace this partly to artificial biases in the texts people create, but part of the effect is the result of the way semantic spaces are organized:
  - Specific claims are likely to be premises in entailment cases.
  - General claims are likely to be hypotheses in entailment pairs.
  - Specific claims are more likely to lead to contradiction.

SNLI and MultiNLI

SNLI

1. Bowman et al. 2015a
2. All the premises are image captions from the Flickr30K corpus (Young et al. 2014).
3. All the hypotheses were written by crowdworkers.
4. Some of the sentences reflect stereotypes (Rudinger et al. 2017).
5. 550,152 train examples; 10K dev; 10K test
6. Mean length in tokens:
   - Premise: 14.1
   - Hypothesis: 8.3
7. Clause-types:
   - Premise S-rooted: 74%
   - Hypothesis S-rooted: 88.9%
8. Vocab size: 37,026
9. 56,951 examples validated by four additional annotators.
   - 58.3% examples with unanimous gold label
   - 91.2% of gold labels match the author's label
   - 0.70 overall Fleiss kappa
10. Leaderboard: `https://nlp.stanford.edu/projects/snli/`

Crowdsourcing methods

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.*

Examples

Premise	Relation	Hypothesis
A man inspects the uniform of a figure in some East Asian country.	**contradiction** c c c c c	The man is sleeping
An older and younger man smiling.	**neutral** n n e n n	Two men are smiling and laughing at the cats playing on the floor.
A black race car starts up in front of a crowd of people.	**contradiction** c c c c c	A man is driving down a lonely road.
A soccer game with multiple males playing.	**entailment** e e e e e	Some men are playing a sport.
A smiling costumed woman is holding an umbrella.	**neutral** n n e c n	A happy woman in a fairy costume holds an umbrella.

Event coreference

Premise	Relation	Hypothesis
A boat sank in the Pacific Ocean.	contradiction	A boat sank in the Atlantic Ocean.
Ruth Bader Ginsburg was appointed to the Supreme Court.	contradiction	I had a sandwich for lunch today

If premise and hypothesis *probably* describe a different photo, then the label is contradiction

MultiNLI

1. Williams et al. 2018
2. Train premises drawn from five genres:
   - Fiction: works from 1912–2010 spanning many genres
   - Government: reports, letters, speeches, etc., from government websites
   - The *Slate* website
   - Telephone: the Switchboard corpus
   - Travel: Berlitz travel guides
3. Additional genres just for dev and test (the mismatched condition):
   - The 9/11 report
   - Face-to-face: The Charlotte Narrative and Conversation Collection
   - Fundraising letters
   - Non-fiction from Oxford University Press
   - *Verbatim*: articles about linguistics
4. 392,702 train examples; 20K dev; 20K test
5. 19,647 examples validated by four additional annotators
   - 58.2% examples with unanimous gold label
   - 92.6% of gold labels match the author's label
6. Test-set labels available as a Kaggle competition.
7. Project page: `https://www.nyu.edu/projects/bowman/multinli/`

MultiNLI annotations

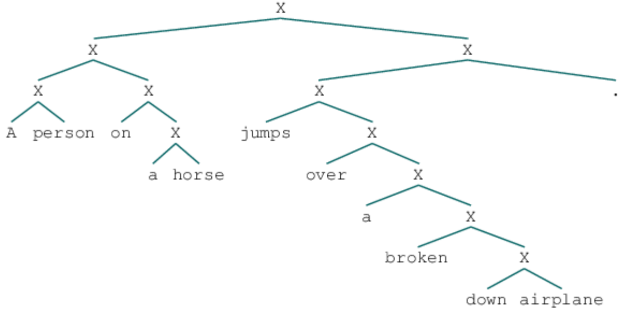
	Matched	Mismatched
ACTIVE/PASSIVE	15	10
ANTO	17	20
BELIEF	66	58
CONDITIONAL	23	26
COREF	30	29
LONG_SENTENCE	99	109
MODAL	144	126
NEGATION	129	104
PARAPHRASE	25	37
QUANTIFIER	125	140
QUANTITY/TIME_REASONING	15	39
TENSE_DIFFERENCE	51	18
WORD_OVERLAP	28	37
	767	753

Code snippets: Readers and Example objects

```
In [1]: import nli
        import os

In [2]: SNLI_HOME = os.path.join("data", "nlidata", "snli_1.0")
        MULTINLI_HOME = os.path.join("data", "nlidata", "multinli_1.0")

In [3]: snli_train_reader = nli.SNLITrainReader(SNLI_HOME, samp_percentage=0.10)

In [4]: snli_dev_reader = nli.SNLIDevReader(SNLI_HOME, samp_percentage=0.10)

In [5]: multi_train_reader = nli.MultiNLITrainReader(SNLI_HOME, samp_percentage=0.10)

In [6]: multi_matched_dev_reader = nli.MultiNLIMatchedDevReader(SNLI_HOME)

In [7]: multi_mismatched_dev_reader = nli.MultiNLIMismatchedDevReader(SNLI_HOME)

In [8]: snli_iterator = iter(nli.SNLITrainReader(SNLI_HOME).read())

In [9]: snli_ex = next(snli_iterator)

In [10]: print(snli_ex)

A person on a horse jumps over a broken down airplane.
neutral
A person is training his horse for a competition.
```

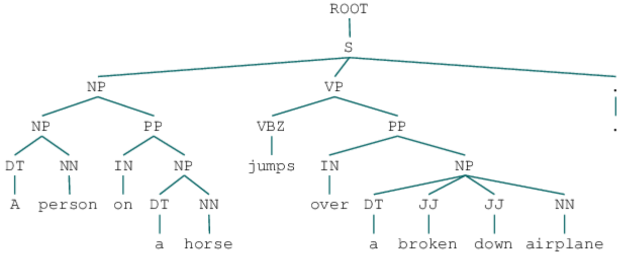
Code snippets: Readers and Example objects

```
In [11]: snli_ex.sentence1
Out[11]: 'A person on a horse jumps over a broken down airplane.'
In [12]: snli_ex.sentence2
Out[12]: 'A person is training his horse for a competition.'
In [13]: snli_ex.gold_label
Out[13]: 'neutral'
In [14]: snli_ex.sentence1_binary_parse
Out[14]:
```



Code snippets: Readers and Example objects

```
In [11]: snli_ex.sentence1
Out[11]: 'A person on a horse jumps over a broken down airplane.'
In [12]: snli_ex.sentence2
Out[12]: 'A person is training his horse for a competition.'
In [13]: snli_ex.gold_label
Out[13]: 'neutral'
In [15]: snli_ex.sentence1_parse
Out[15]:
```



Code snippets: MultiNLI annotations

```
In [1]: import nli
        import os

In [2]: ANN_HOME = os.path.join("data", "nlidata", "multinli_1.0_annotations")
        MULTINLI_HOME = os.path.join("data", "nlidata", "multinli_1.0")

In [3]: matched_filename = os.path.join(ANN_HOME, "multinli_1.0_matched_annotations.txt")

        mismatched_filename = os.path.join(ANN_HOME, "multinli_1.0_mismatched_annotations.txt")

In [4]: matched_ann = nli.read_annotated_subset(matched_filename, MULTINLI_HOME)

In [5]: len(matched_ann)

Out[5]: 495

In [6]: pair_id = '116176e'
        ann_ex = matched_ann[pair_id]
        print("pairID: {}".format(pair_id))
        print(ann_ex['annotations'])
        ex = ann_ex['example']
        print(ex.sentence1)
        print(ex.gold_label)
        print(ex.sentence2)

pairID: 116176e
['#MODAL', '#COREF']
Students of human misery can savor its underlying sadness and futility.
entailment
Those who study human misery will savor the sadness and futility.
```

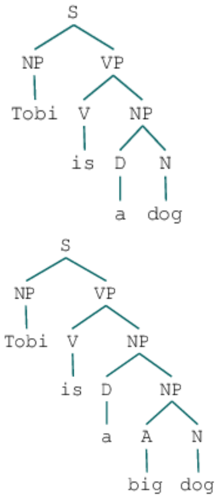
Hand-built features

Word overlap and word-cross product

```
In [1]: from collections import Counter
        from itertools import product
        import nli
        from nltk.tree import Tree
        import os

In [2]: def word_overlap_phi(t1, t2):
            overlap = set([w1 for w1 in t1.leaves() if w1 in t2.leaves()])
            return Counter(overlap)

In [3]: def word_cross_product_phi(t1, t2):
            return Counter([(w1, w2) for w1, w2 in product(t1.leaves(), t2.leaves())])

In [4]: t1 = Tree.fromstring("""(S (NP Tobi) (VP (V is) (NP (D a) (N dog))))""")

In [5]: t2 = Tree.fromstring("""(S (NP Tobi) (VP (V is) (NP (D a) (NP (A big ) (N dog)))))""")
```

Word overlap and word-cross product

```
In [6]: display(t1, t2)
```




```
In [7]: word_overlap_phi(t1, t2)

Out[7]: Counter({'Tobi': 1, 'dog': 1, 'is': 1, 'a': 1})

In [8]: word_cross_product_phi(t1, t2)

Out[8]: Counter({('Tobi', 'Tobi'): 1,
                 ('Tobi', 'is'): 1,
                 ('Tobi', 'a'): 1,
                 ('Tobi', 'big'): 1,
                 ('Tobi', 'dog'): 1,
                 ('is', 'Tobi'): 1,
                 ('is', 'is'): 1,
                 ('is', 'a'): 1,
                 ('is', 'big'): 1,
                 ('is', 'dog'): 1,
                 ('a', 'Tobi'): 1,
                 ('a', 'is'): 1,
                 ('a', 'a'): 1,
                 ('a', 'big'): 1,
                 ('a', 'dog'): 1,
                 ('dog', 'Tobi'): 1,
                 ('dog', 'is'): 1,
                 ('dog', 'a'): 1,
                 ('dog', 'big'): 1,
                 ('dog', 'dog'): 1})
```

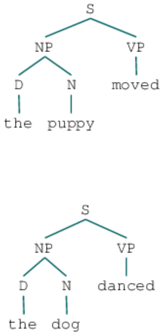
WordNet features

```
In [1]: from collections import Counter
        from itertools import product
        from nltk.corpus import wordnet as wn
        from nltk.tree import Tree

In [2]: puppies = wn.synsets('puppy')
        [h for ss in puppies for h in ss.hypernyms()]

Out[2]: [Synset('dog.n.01'), Synset('pup.n.01'), Synset('young_person.n.01')]

In [3]: # A more conservative approach uses just the first-listed
        # Synset, which should be the most frequent sense:
        wn.synsets('puppy')[0].hypernyms()

Out[3]: [Synset('dog.n.01'), Synset('pup.n.01')]

In [4]: def wordnet_features(t1, t2, methodname):
            pairs = []
            words1 = t1.leaves()
            words2 = t2.leaves()
            for w1, w2 in product(words1, words2):
                hyps = [h for ss in wn.synsets(w1) for h in getattr(ss, methodname)()]
                syns = wn.synsets(w2)
                if set(hyps) & set(syns):
                    pairs.append((w1, w2))
            return Counter(pairs)

In [5]: def hypernym_features(t1, t2):
            return wordnet_features(t1, t2, 'hypernyms')

In [6]: def hyponym_features(t1, t2):
            return wordnet_features(t1, t2, 'hyponyms')
```

WordNet features

```
In [7]: t1 = Tree.fromstring("""(S (NP (D the) (N puppy)) (VP moved))""")

In [8]: t2 = Tree.fromstring("""(S (NP (D the) (N dog)) (VP danced))""")
```

```
In [9]: display(t1, t2)
```

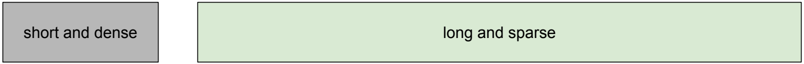



```
In [10]: hypernym_features(t1, t2)

Out[10]: Counter({('puppy', 'dog'): 1})

In [11]: hyponym_features(t1, t2)

Out[11]: Counter({('moved', 'danced'): 1})
```

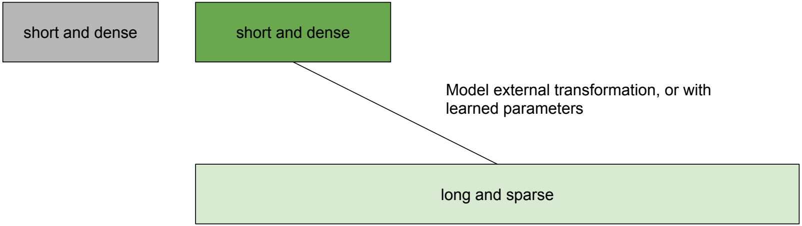
Other hand-built features

1. Additional WordNet relations
2. Edit distance
3. Word differences (cf. word overlap)
4. Alignment-based features
5. Negation
6. Quantifier relations (e.g., *every* $\sqsubset$ *some*; see MacCartney & Manning 2009)
7. Named entity features

Combining dense and sparse representations



Combining dense and sparse representations



nli.experiment

Complete experiment with nli.experiment

```
In [1]: from collections import Counter
        import nli
        import os
        from sklearn.linear_model import LogisticRegression
        import utils

In [2]: SNLI_HOME = os.path.join("data", "nlidata", "snli_1.0")

In [3]: def word_overlap_phi(t1, t2):
            overlap = set([w1 for w1 in t1.leaves() if w1 in t2.leaves()])
            return Counter(overlap)

In [4]: def fit_softmax(X, y):
            mod = LogisticRegression(
                fit_intercept=True, solver='liblinear', multi_class='auto')
            mod.fit(X, y)
            return mod

In [5]: train_reader_10 = nli.SNLITrainReader(SNLI_HOME, samp_percentage=0.10)

In [6]: basic_experiment = nli.experiment(
            train_reader_10,
            word_overlap_phi,
            fit_softmax,
            assess_reader=None,             # Default
            train_size=0.7,                 # Default
            score_func=utils.safe_macro_f1, # Default
            vectorize=True,                 # Default
            verbose=True,                   # Default
            random_state=None)              # Default
```

Hyperparameter selection on train subsets

```
In [1]: from collections import Counter
        import nli
        import os
        from sklearn.linear_model import LogisticRegression
        import utils

In [2]: SNLI_HOME = os.path.join("data", "nlidata", "snli_1.0")

In [3]: def word_overlap_phi(t1, t2):
            overlap = set([w1 for w1 in t1.leaves() if w1 in t2.leaves()])
            return Counter(overlap)
```

```
In [4]: def fit_softmax_with_crossvalidation(X, y):
            basemod = LogisticRegression(
                fit_intercept=True, solver='liblinear', multi_class='auto')
            param_grid = {'C': [0.6, 0.7, 0.8, 1.0, 1.1], 'penalty': ['l1','l2']}
            best_mod = utils.fit_classifier_with_crossvalidation(
                X, y, basemod, cv=3, param_grid=param_grid)
            return best_mod

In [5]: # Select hyperparameters based on a subset of the data:
        tuning_experiment = nli.experiment(
            nli.SNLITrainReader(SNLI_HOME, samp_percentage=0.10),
            word_overlap_phi,
            fit_softmax_with_crossvalidation)

Best params: {'C': 1.0, 'penalty': 'l2'}
Best score: 0.413
```

Hyperparameter selection on train subsets

```
In [1]: from collections import Counter
        import nli
        import os
        from sklearn.linear_model import LogisticRegression
        import utils

In [2]: SNLI_HOME = os.path.join("data", "nlidata", "snli_1.0")

In [3]: def word_overlap_phi(t1, t2):
            overlap = set([w1 for w1 in t1.leaves() if w1 in t2.leaves()])
            return Counter(overlap)
```

```
In [6]: def fit_softmax_classifier_with_preselected_params(X, y):
            mod = LogisticRegression(
                fit_intercept=True, solver='liblinear', multi_class='auto',
                C=1.0, penalty='l2')
            mod.fit(X, y)
            return mod

In [7]: # Use the selected hyperparamters in a (costly) full dataset training run:
        full_experiment = nli.experiment(
            nli.SNLITrainReader(SNLI_HOME),
            word_overlap_phi,
            fit_softmax_classifier_with_preselected_params,
            assess_reader=nli.SNLIDevReader(SNLI_HOME))
```

Hyperparameter selection with a few iterations

```
In [8]: def fit_softmax_with_crossvalidation_small_iter(X, y):
            basemod = LogisticRegression(
                fit_intercept=True, solver='liblinear', multi_class='auto',
                max_iter=3)
            param_grid = {'C': [0.6, 0.7, 0.8, 1.0, 1.1], 'penalty': ['l1','l2']}
            best_mod = utils.fit_classifier_with_crossvalidation(
                X, y, basemod, cv=3, param_grid=param_grid)
            return best_mod

In [9]: # Select hyperparameters based on a few iterations:
        tuning_experiment_small_iter = nli.experiment(
            nli.SNLITrainReader(SNLI_HOME),
            word_overlap_phi,
            fit_softmax_with_crossvalidation_small_iter)

.../base.py:922: ConvergenceWarning: Liblinear failed to converge,
increase the number of iterations.


Best params: {'C': 1.0, 'penalty': 'l1'}
Best score: 0.425
```

A hypothesis-only experiment

```
In [1]: from collections import Counter
        import nli
        import os
        from sklearn.dummy import DummyClassifier
        from sklearn.linear_model import LogisticRegression
        import utils

In [2]: SNLI_HOME = os.path.join("data", "nlidata", "snli_1.0")

In [3]: def hypothesis_only_unigrams_phi(t1, t2):
            return Counter(t2.leaves())

In [4]: def fit_softmax_classifier_with_preselected_params(X, y):
            mod = LogisticRegression(
                fit_intercept=True, solver='liblinear', multi_class='auto',
                C=1.0, penalty='l2')
            mod.fit(X, y)
            return mod

In [5]: hypothesis_only_experiment = nli.experiment(
            nli.SNLITrainReader(SNLI_HOME),
            hypothesis_only_unigrams_phi,
            fit_softmax_classifier_with_preselected_params,
            assess_reader=nli.SNLIDevReader(SNLI_HOME))

               precision    recall  f1-score   support

contradiction      0.654     0.631     0.642      3278
   entailment      0.639     0.715     0.675      3329
      neutral      0.670     0.613     0.640      3235

    micro avg      0.653     0.653     0.653      9842
    macro avg      0.655     0.653     0.653      9842
 weighted avg      0.654     0.653     0.653      9842
```

A hypothesis-only experiment

```
In [6]: def fit_dummy_classifier(X, y):
            mod = DummyClassifier(strategy='stratified')
            mod.fit(X, y)
            return mod

In [7]: random_experiment = nli.experiment(
            nli.SNLITrainReader(SNLI_HOME),
            lambda t1, t2: {'constant': 1},  # `DummyClassifier` ignores this!
            fit_dummy_classifier,
            assess_reader=nli.SNLIDevReader(SNLI_HOME))

               precision    recall  f1-score   support

contradiction      0.336     0.338     0.337      3278
   entailment      0.336     0.330     0.333      3329
      neutral      0.331     0.335     0.333      3235

    micro avg      0.334     0.334     0.334      9842
    macro avg      0.334     0.334     0.334      9842
 weighted avg      0.334     0.334     0.334      9842
```

A premise-only experiment

```
In [8]: def premise_only_unigrams_phi(t1, t2):
            return Counter(t1.leaves())

In [9]: premise_only_experiment = nli.experiment(
            nli.SNLITrainReader(SNLI_HOME),
            premise_only_unigrams_phi,
            fit_softmax_classifier_with_preselected_params,
            assess_reader=nli.SNLIDevReader(SNLI_HOME))

               precision    recall  f1-score   support

contradiction      0.337     0.255     0.290      3278
   entailment      0.340     0.388     0.363      3329
      neutral      0.330     0.364     0.346      3235

    micro avg      0.336     0.336     0.336      9842
    macro avg      0.336     0.336     0.333      9842
 weighted avg      0.336     0.336     0.333      9842
```

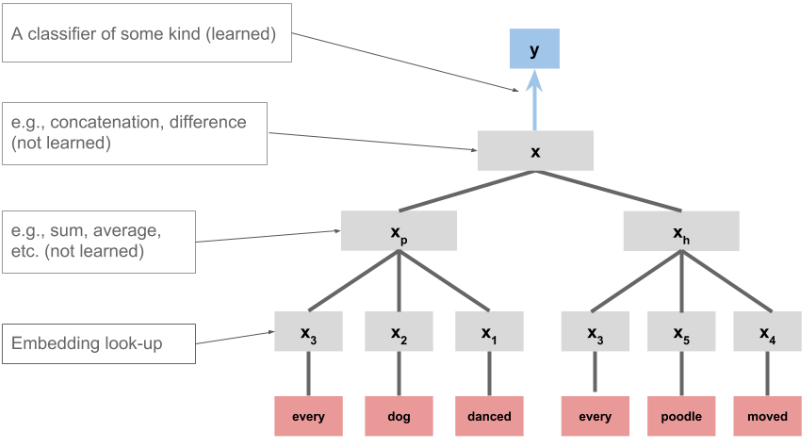
A premise-only experiment

```
In [8]: def premise_only_unigrams_phi(t1, t2):
            return Counter(t1.leaves())

In [9]: premise_only_experiment = nli.experiment(
            nli.SNLITrainReader(SNLI_HOME),
            premise_only_unigrams_phi,
            fit_softmax_classifier_with_preselected_params,
            assess_reader=nli.SNLIDevReader(SNLI_HOME))

               precision    recall  f1-score   support

contradiction      0.337     0.255     0.290      3278
   entailment      0.340     0.388     0.363      3329
      neutral      0.330     0.364     0.346      3235

    micro avg      0.336     0.336     0.336      9842
    macro avg      0.336     0.336     0.333      9842
 weighted avg      0.336     0.336     0.333      9842
```

- A result of the data collection method: each premise is paired with one hypothesis from each class.
- The logistic regression premise-only baseline for the word-entailment bake-off is ≈0.47, vs. ≈0.50 for hypothesis-only.

Sentence-encoding models

Distributed representations as features



Code: Distributed representations as features

```
In [1]: import nli
        import numpy as np
        import os
        from sklearn.linear_model import LogisticRegression
        import utils

In [2]: SNLI_HOME = os.path.join("data", "nlidata", "snli_1.0")
        GLOVE_HOME = os.path.join('data', 'glove.6B')

In [3]: glove_lookup = utils.glove2dict(
            os.path.join(GLOVE_HOME, 'glove.6B.50d.txt'))

In [4]: def _get_tree_vecs(tree, lookup, np_func):
            allvecs = np.array([lookup[w] for w in tree.leaves() if w in lookup])
            if len(allvecs) == 0:
                dim = len(next(iter(lookup.values())))
                feats = np.zeros(dim)
            else:
                feats = np_func(allvecs, axis=0)
            return feats

In [5]: def glove_leaves_phi(t1, t2, np_func=np.sum):
            prem_vecs = _get_tree_vecs(t1, glove_lookup, np_func)
            hyp_vecs = _get_tree_vecs(t2, glove_lookup, np_func)
            return np.concatenate((prem_vecs, hyp_vecs))

In [6]: def glove_leaves_sum_phi(t1, t2):
            return glove_leaves_phi(t1, t2, np_func=np.sum)
```

Code: Distributed representations as features

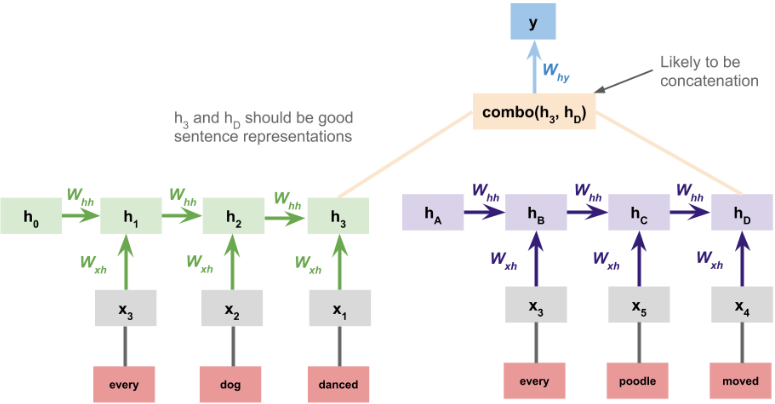
```
In [7]: def fit_softmax(X, y):
            mod = LogisticRegression(
                fit_intercept=True, solver='liblinear', multi_class='auto')
            mod.fit(X, y)
            return mod

In [8]: glove_sum_experiment = nli.experiment(
            nli.SNLITrainReader(SNLI_HOME),
            glove_leaves_sum_phi,
            fit_softmax,
            assess_reader=nli.SNLIDevReader(SNLI_HOME),
            vectorize=False) # We already have vectors!

               precision    recall  f1-score   support

contradiction      0.505     0.476     0.490      3278
   entailment      0.500     0.561     0.529      3329
      neutral      0.549     0.513     0.530      3235

    micro avg      0.517     0.517     0.517      9842
    macro avg      0.518     0.516     0.516      9842
 weighted avg      0.518     0.517     0.516      9842
```

Rationale for sentence-encoding models

1. Encoding the premise and hypothesis separately might give the model a chance to find rich abstract relationships between them.

2. Sentence-level encoding could facilitate transfer to other tasks (Dagan et al.'s (2006) vision).

Sentence-encoding RNNs



PyTorch strategy: Sentence-encoding RNNs

The full implementation is in `nli_02_models.ipynb`.

`TorchRNNSentenceEncoderDataset`

This is conceptually a list of pairs of sequences, each with their lengths, and a label vector:

$$\left[\left(\left[\texttt{every}, \texttt{dog}, \texttt{danced}\right], \left[\texttt{every}, \texttt{poodle}, \texttt{moved}\right]\right), (3, 3), \textbf{entailment}\right]$$

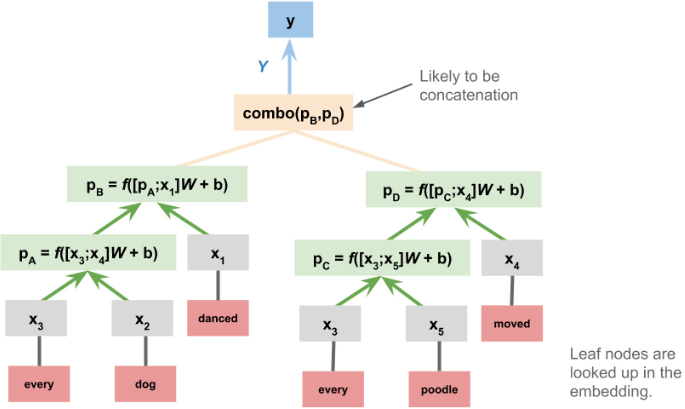
`TorchRNNSentenceEncoderClassifierModel`

This is concetually a premise RNN and a hypothesis RNN. The `forward` method uses them to process the two parts of the example, concatenate the outputs of those passes, and feed them into a classifier.

`TorchRNNSentenceEncoderClassifier`

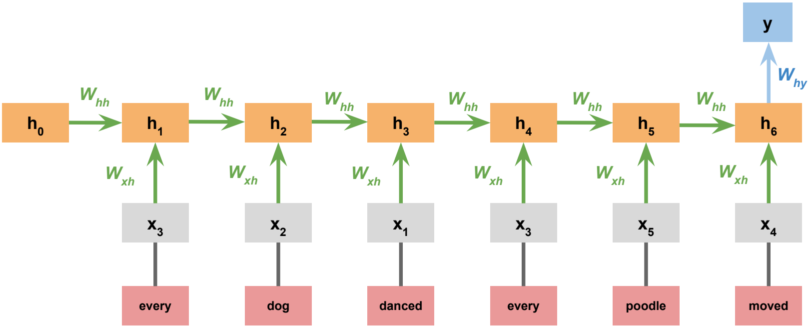
This is basically unchanged from its super class `TorchRNNClassifier`, except the `predict_proba` method needs to deal with the new example format.

Sentence-encoding TreeNNs



Chained models

Simple RNN


y
W_{hy}
h_0
W_{hh}
h_1
h_2
h_3
h_4
h_5
h_6
W_{xh}
x_3
x_2
x_1
x_3
x_5
x_4
every
dog
danced
every
poodle
moved

Rationale for sentence-encoding models

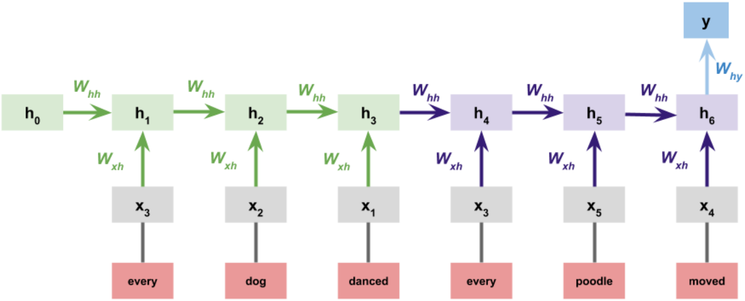
1. The premise truly establishes the context for the hypothesis.

2. Might be seen as corresponding to a real processing model.

Code snippet: Simple RNN

```
In [1]: import nli
        import os
        from torch_rnn_classifier import TorchRNNClassifier
        import utils

In [2]: SNLI_HOME = os.path.join("data", "nlidata", "snli_1.0")

In [3]: # Consider adding a fixed boundary symbol between premise and hypothesis.
        def simple_chained_rep_rnn_phi(t1, t2):
            return t1.leaves() + t2.leaves()

In [4]: def fit_simple_chained_rnn(X, y):
            vocab = utils.get_vocab(X, n_words=10000)
            mod = TorchRNNClassifier(vocab, hidden_dim=50, max_iter=50)
            mod.fit(X, y)
            return mod

In [5]: simple_chained_rnn_experiment = nli.experiment(
            nli.SNLITrainReader(SNLI_HOME, samp_percentage=0.10),
            simple_chained_rep_rnn_phi,
            fit_simple_chained_rnn,
            vectorize=False)
```

Premise and hypothesis RNNs



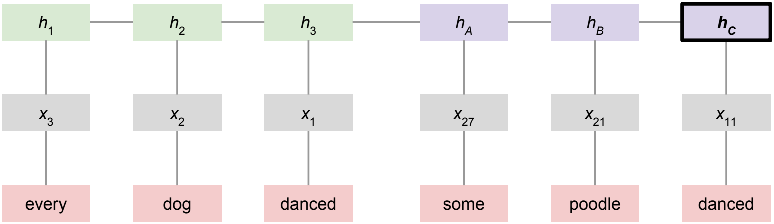
The PyTorch implementation strategy is similar to the one outlined earlier for sentence-encoding RNNs, except the final hidden state of the premise RNN becomes the initial hidden state for the hypothesis RNN.

Attention

Guiding ideas

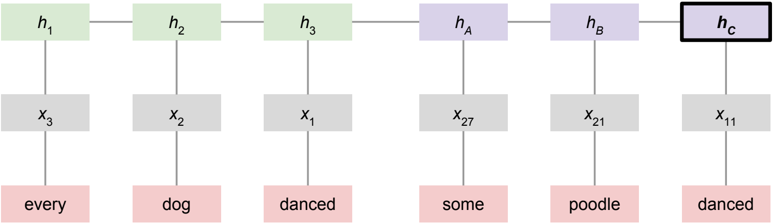
1. We need more connections between premise and hypothesis.
2. In processing the hypothesis, the model needs “reminders” of what the premise contained; the final premise hidden state isn’t enough.
3. Soft alignment between premise and hypothesis – a neural interpretation of an old idea in NLI.

Global attention



Global attention

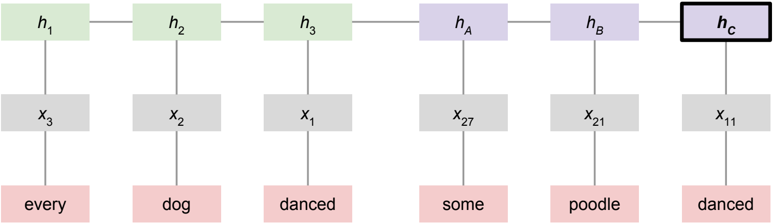
scores $\tilde{\alpha} = \begin{bmatrix} h_C^\top h_1 & h_C^\top h_2 & h_C^\top h_3 \end{bmatrix}$



Global attention

attention weights $\alpha = \mathbf{softmax}(\tilde{\alpha})$

scores $\tilde{\alpha} = \begin{bmatrix} h_C^\top h_1 & h_C^\top h_2 & h_C^\top h_3 \end{bmatrix}$

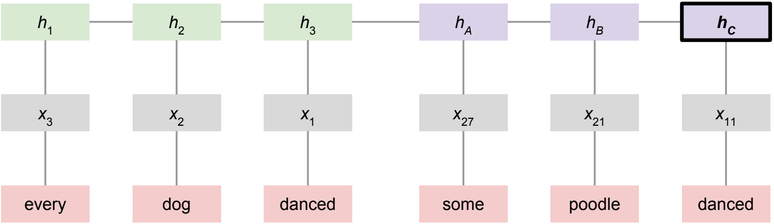


Global attention

context $\kappa = \mathbf{mean}(\alpha_1 h_1, \alpha_2 h_2, \alpha_3 h_3)$

attention weights $\alpha = \mathbf{softmax}(\tilde{\alpha})$

scores $\tilde{\alpha} = \begin{bmatrix} h_C^\top h_1 & h_C^\top h_2 & h_C^\top h_3 \end{bmatrix}$



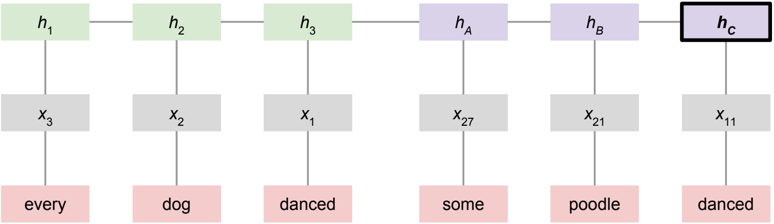
Global attention

attention combo $\tilde{h} = \tanh([\kappa; h_C]W_\kappa)$

context $\kappa = \mathbf{mean}(\alpha_1 h_1, \alpha_2 h_2, \alpha_3 h_3)$

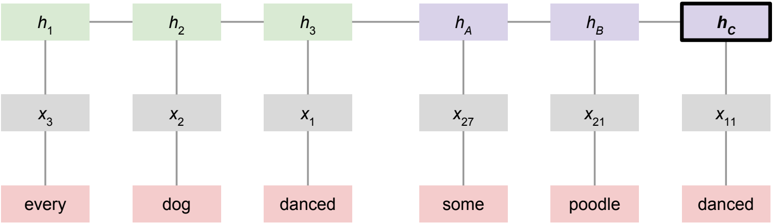
attention weights $\alpha = \mathbf{softmax}(\tilde{\alpha})$

scores $\tilde{\alpha} = \left[\begin{array}{ccc} h_C^\top h_1 & h_C^\top h_2 & h_C^\top h_3 \end{array}\right]$



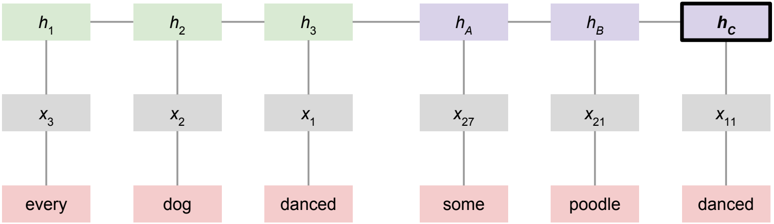
Global attention

attention combo	$\tilde{h} = \tanh([\kappa; h_C]W_\kappa)$ or $\tilde{h} = \tanh(\kappa W_\kappa + h_C W_h)$
context	$\kappa = \textbf{mean}(\alpha_1 h_1, \alpha_2 h_2, \alpha_3 h_3)$
attention weights	$\alpha = \textbf{softmax}(\tilde{\alpha})$
scores	$\tilde{\alpha} = \begin{bmatrix} h_C^\top h_1 & h_C^\top h_2 & h_C^\top h_3 \end{bmatrix}$

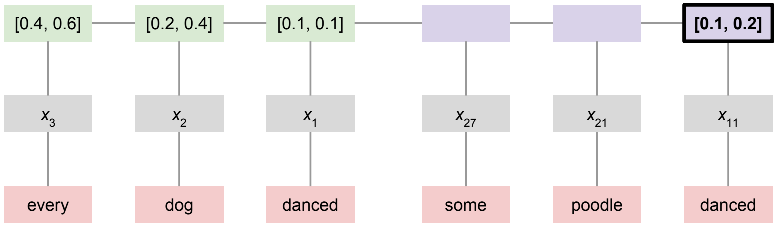


Global attention

classifier	$y = \textbf{softmax}(\tilde{h}W + b)$
attention combo	$\tilde{h} = \tanh([\kappa; h_C]W_\kappa)$
context	$\kappa = \textbf{mean}(\alpha_1 h_1, \alpha_2 h_2, \alpha_3 h_3)$
attention weights	$\alpha = \textbf{softmax}(\tilde{\alpha})$
scores	$\tilde{\alpha} = \begin{bmatrix} h_C^\top h_1 & h_C^\top h_2 & h_C^\top h_3 \end{bmatrix}$

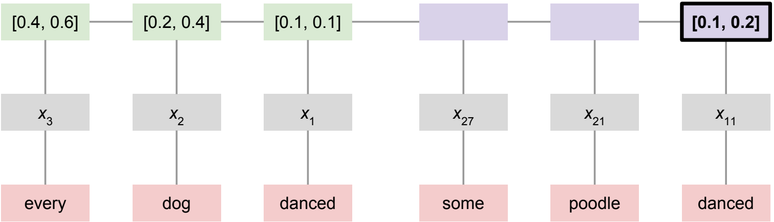


Global attention



Global attention

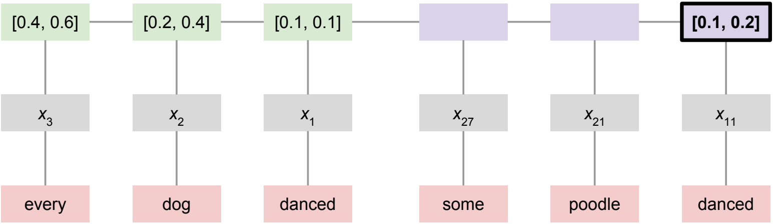
scores $\tilde{\alpha} = [0.16, 0.10, 0.03]$



Global attention

attention weights $\alpha = [0.35, 0.33, 0.31]$

scores $\tilde{\alpha} = [0.16, 0.10, 0.03]$

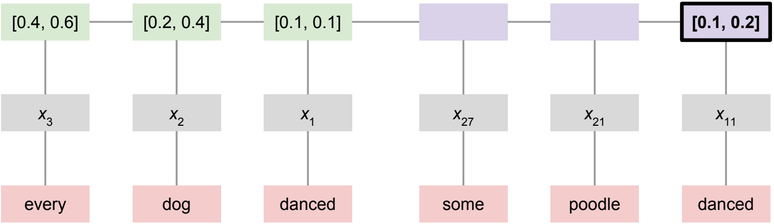


Global attention

context $\kappa = \textbf{mean}(.35 \cdot [.4, .6], .33 \cdot [.2, .4], .31 \cdot [.1, .1])$

attention weights $\alpha = [0.35, 0.33, 0.31]$

scores $\tilde{\alpha} = [0.16, 0.10, 0.03]$



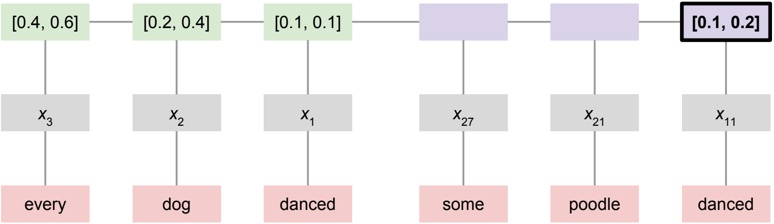
Global attention

attention combo $\tilde{h} = \tanh([0.07, 0.11, 0.1, 0.2]W_{\kappa})$

context $\kappa = \textbf{mean}(.35 \cdot [.4, .6], .33 \cdot [.2, .4], .31 \cdot [.1, .1])$

attention weights $\alpha = [0.35, 0.33, 0.31]$

scores $\tilde{\alpha} = [0.16, 0.10, 0.03]$



Global attention

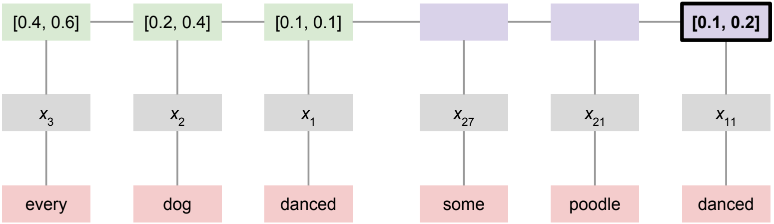
classifier $y = \textbf{softmax}(\tilde{h}W + b)$

attention combo $\tilde{h} = \tanh([0.07, 0.11, 0.1, 0.2]W_{\kappa})$

context $\kappa = \textbf{mean}(.35 \cdot [.4, .6], .33 \cdot [.2, .4], .31 \cdot [.1, .1])$

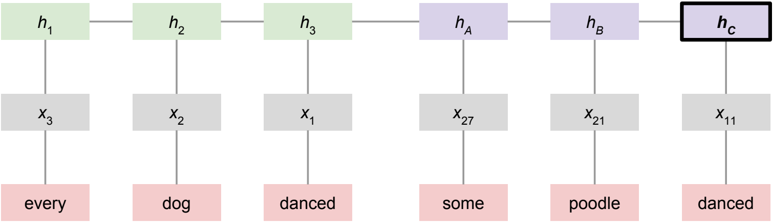
attention weights $\alpha = [0.35, 0.33, 0.31]$

scores $\tilde{\alpha} = [0.16, 0.10, 0.03]$

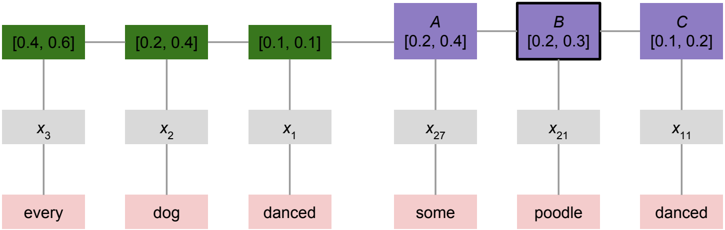


Other scoring functions (Luong et al. 2015)

$$\textbf{score}(h_C, h_i) = \begin{cases} h_C^\top h_i & \text{dot} \\ h_C^\top W_\alpha h_i & \text{general} \\ W_\alpha[h_C; h_i] & \text{concat} \end{cases}$$

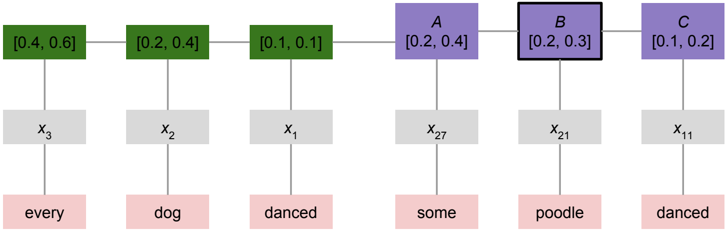


Word-by-word attention



Word-by-word attention

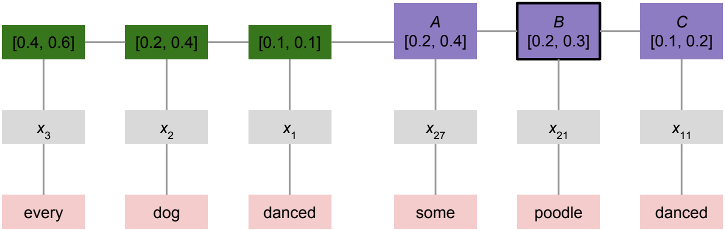
$$M = \tanh\left(\begin{bmatrix} 0.4 & 0.6 \\ 0.2 & 0.4 \\ 0.1 & 0.1 \end{bmatrix} + \begin{bmatrix} 0.2 & 0.3 & \kappa_A \\ 0.2 & 0.3 & \kappa_A \\ 0.2 & 0.3 & \kappa_A \end{bmatrix} W_h\right)$$



Word-by-word attention

weights at B $\quad \alpha_B = \textbf{softmax}(Mw)$

$$M = \tanh\left(\begin{bmatrix} 0.4 & 0.6 \\ 0.2 & 0.4 \\ 0.1 & 0.1 \end{bmatrix} + \begin{bmatrix} 0.2 & 0.3 & \kappa_A \\ 0.2 & 0.3 & \kappa_A \\ 0.2 & 0.3 & \kappa_A \end{bmatrix} W_h\right)$$

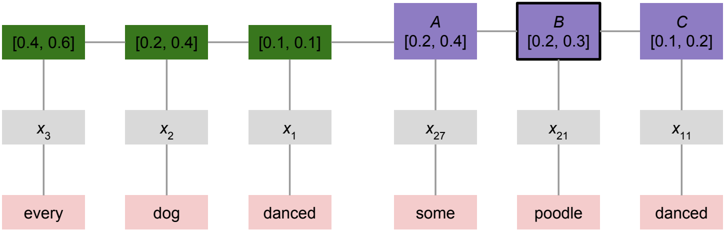


Word-by-word attention

context at B $\qquad \kappa_B = \begin{bmatrix} 0.4 & 0.6 \\ 0.2 & 0.4 \\ 0.1 & 0.1 \end{bmatrix} \alpha_B + \tanh\left(\kappa_A W_\alpha\right)$

weights at B $\qquad \alpha_B = \mathbf{softmax}(Mw)$

$$M = \tanh\left(\begin{bmatrix} 0.4 & 0.6 \\ 0.2 & 0.4 \\ 0.1 & 0.1 \end{bmatrix} + \begin{bmatrix} 0.2 & 0.3 & \kappa_A \\ 0.2 & 0.3 & \kappa_A \\ 0.2 & 0.3 & \kappa_A \end{bmatrix} W_h \right)$$



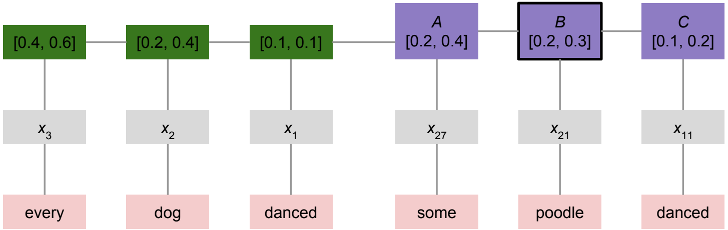
Word-by-word attention

classifier input $\tilde{h} = \tanh([\kappa_C; h_C]W_\kappa)$

context at B $\kappa_B = \begin{bmatrix} 0.4 & 0.6 \\ 0.2 & 0.4 \\ 0.1 & 0.1 \end{bmatrix} \alpha_B + \tanh(\kappa_A W_\alpha)$

weights at B $\alpha_B = \textbf{softmax}(Mw)$

$$M = \tanh\left(\begin{bmatrix} 0.4 & 0.6 \\ 0.2 & 0.4 \\ 0.1 & 0.1 \end{bmatrix} + \begin{bmatrix} 0.2 & 0.3 & \kappa_A \\ 0.2 & 0.3 & \kappa_A \\ 0.2 & 0.3 & \kappa_A \end{bmatrix} W_h \right)$$



Other variants

- Local attention (Luong et al. 2015) builds connections between selected points in the premise and hypothesis.
- Word-by-word attention can be set up in many ways, with many more learned parameters than my simple example. A pioneering instance for NLI is Rocktäschel et al. 2016.
- The attention representation at time t could be appended to the hidden representation at $t+1$ (Luong et al. 2015).
- Vaswani et al. (2017) use attention for their *primary* connections, a reversal of the usual pattern.
- Memory networks (Weston et al. 2015) can be used to address similar issues related to properly recalling past experiences.

Error analysis

MultiNLI annotations

Annotations	Premise	Relation	Hypothesis
#MODAL, #COREF	Students of human misery can savor its underlying sadness and futility. entailment	entailment	Those who study human misery will savor the sadness and futility.
#NEGATION, #TENSE_ DIFFERENCE, #CONDITIONAL	oh really it wouldn't matter if we plant them when it was starting to get warmer	contradiction	It is better to plant when it is colder.
#QUANTIFIER, #AC- TIVE/PASSIVE	They consolidated programs to increase efficiency and deploy resources more effectively	entailment	Programs to increase efficiency were consolidated.

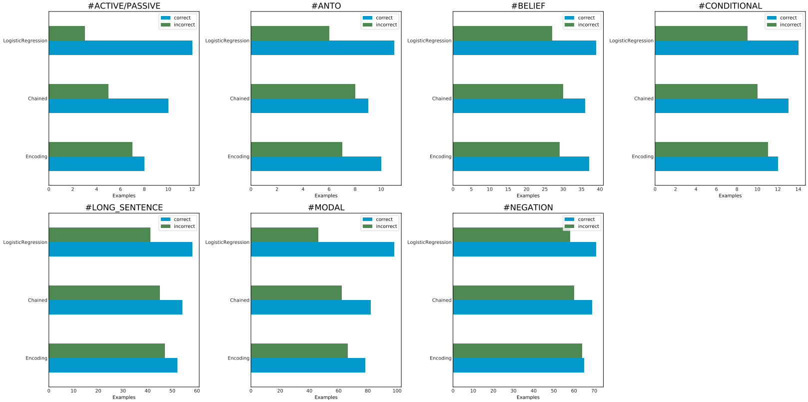
Matched MultiNLI annotations

Model	Features	Macro-F1
Logistic regression	cross-product	0.58
Chained LSTM	random embedding	0.55
Sentence-encoding LSTM	random embedding	0.51

- Logistic regression tuned hyperparameters: `C` (0.1 to 1.2 by 0.1) and `penalty` (L1, L2). Model file is ≈ 600MB; ≈ 16M features.
- LSTM tuned hyperparameters: `embed_dim` (50, 100), `hidden_dim` (50, 100, 150), learning rate (0.001, 0.01, 0.05), and activation function (Tanh, ReLU). Model files are ≈ 1MB each.

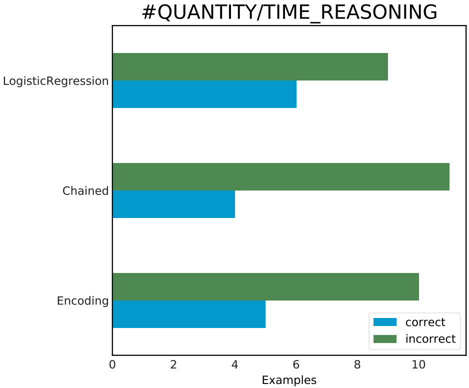
MultiNLI annotations: LSTMs by category

All models more correct than incorrect



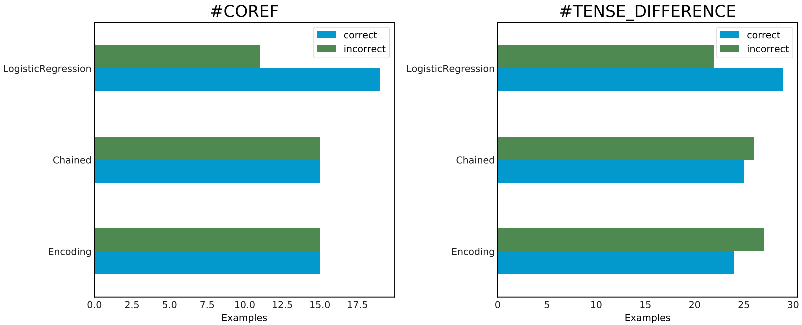
MultiNLI annotations: LSTMs by category

All models more incorrect than correct



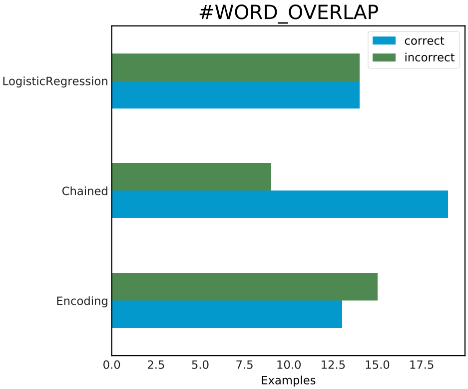
MultiNLI annotations: LSTMs by category

Only Logistic Regression more correct than incorrect



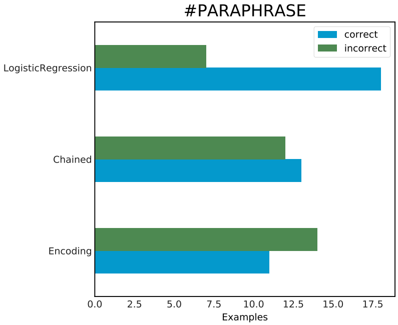
MultiNLI annotations: LSTMs by category

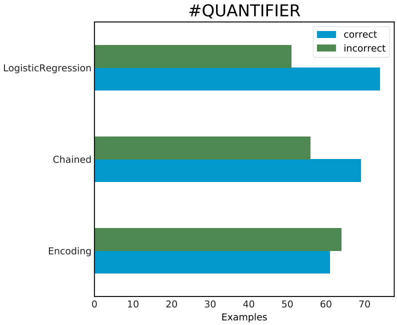
Only chained LSTM more correct than incorrect



MultiNLI annotations: LSTMs by category

Only sentence-encoding LSTM more incorrect than correct





(There were no categories in which only the sentence-encoding LSTM was more correct than incorrect.)

Testing for specific patterns

Does your model know that negation is downward monotone?

Fido moved. $\uparrow$ Fido ran.

Fido did**n't** move. $\downarrow$ Fido didn't run.

Does your model know that *every* is downward monotone on its first argument and upward monotone on its second?

Every dog moved.

$\swarrow$ $\nwarrow$

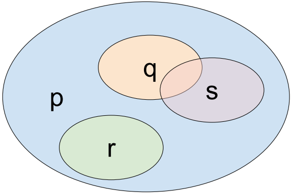
Every puppy moved. **Every** dog ran.

Does your model *systematically* capture such patterns?

Probing with artificial data

Negation (after MacCartney & Manning 2007)

	not-p, not-q	p, not-q	not-p, q
p disjoint q	neutral	subset	superset
p equal q	equal	disjoint	disjoint
p neutral q	neutral	neutral	neutral
p subset q	superset	disjoint	neutral
p superset q	subset	neutral	disjoint



The issue

If your model does perfectly on a doubly negated dataset, will its performance generalize to triply negated cases? This would be evidence that it had truly learned the algebra of negation. See Bowman et al. 2015b; Evans et al. 2018; Geiger et al. 2018.

References I

Bos, Johan & Katja Markert. 2005. Recognising textual entailment with logical inference. In *Proceedings of human language technology conference and conference on empirical methods in natural language processing*, 628–635. Stroudsburg, PA: ACL.

Bowman, Samuel R., Gabor Angeli, Christopher Potts & Christopher D. Manning. 2015a. Learning natural language inference from a large annotated corpus. In *Proceedings of the 2015 conference on Empirical Methods in Natural Language Processing*, 632–642. Stroudsburg, PA: Association for Computational Linguistics.

Bowman, Samuel R., Christopher Potts & Christopher D. Manning. 2015b. Recursive neural networks can learn logical semantics. In *Proceedings of the 3rd workshop on continuous vector space models and their compositionality*, Stroudsburg, PA: Association for Computational Linguistics.

Crouch, Richard, Lauri Karttunen & Annie Zaenen. 2006. Circumscribing is not excluding: A reply to Manning. Ms., Palo Alto Research Center.

Dagan, Ido, Oren Glickman & Bernardo Magnini. 2006. The PASCAL recognising textual entailment challenge. In J. Quinonero-Candela, I. Dagan, B. Magnini & F. d'Alché Buc (eds.), *Machine learning challenges, lecture notes in computer science*, vol. 3944, 177–190. Springer-Verlag.

Evans, Richard, David Saxton, David Amos, Pushmeet Kohli & Edward Grefenstette. 2018. Can neural networks understand logical entailment? *arXiv:1802.08535* .

Geiger, Atticus, Ignacio Cases, Lauri Karttunen & Christopher Potts. 2018. Stress-testing neural models of natural language inference with multiply-quantified sentences. Ms., Stanford University. arXiv 1810.13033.

Goldberg, Yoav. 2015. A primer on neural network models for natural language processing. Ms., Bar Ilan University.

Gururangan, Suchin, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman & Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 conference of the north american chapter of the association for computational linguistics: Human language technologies, volume 2 (short papers)*, 107–112. New Orleans, Louisiana: Association for Computational Linguistics. doi:10.18653/v1/N18-2017. https://www.aclweb.org/anthology/N18-2017.

Hickl, Andrew & Jeremy Bensley. 2007. A discourse commitment-based framework for recognizing textual entailment. In *Proceedings of the workshop on textual entailment and paraphrasing*, .

Luong, Thang, Hieu Pham & Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, 1412–1421. Lisbon, Portugal: Association for Computational Linguistics. doi:10.18653/v1/D15-1166. https://www.aclweb.org/anthology/D15-1166.

MacCartney, Bill. 2009. *Natural language inference*: Stanford University dissertation.

MacCartney, Bill & Christopher D. Manning. 2007. Natural logic for textual inference. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, 193–200. Prague: Association for Computational Linguistics. http://www.aclweb.org/anthology/W/W07/W07-1431.

References II

MacCartney, Bill & Christopher D. Manning. 2008. Modeling semantic containment and exclusion in natural language inference. In *Proceedings of the 22nd international conference on computational linguistics (coling 2008)*, 521–528. Manchester, UK: Coling 2008 Organizing Committee. http://www.aclweb.org/anthology/C08-1066.

MacCartney, Bill & Christopher D. Manning. 2009. An extended model of natural logic. In *Proceedings of the eighth international conference on computational semantics*, 140–156. Tilburg, The Netherlands: Association for Computational Linguistics. http://www.aclweb.org/anthology/W09-3714.

Manning, Christopher D. 2006. Local textual inference: It's hard to circumscribe, but you know it when you see it – and NLP needs it. Ms., Stanford University.

de Marneffe, Marie-Catherine, Bill MacCartney, Trond Grenager, Daniel Cer, Anna Rafferty & Christopher D Manning. 2006. Learning to distinguish valid textual entailments. In *Proceedings of the 2nd pascal rte challenge workshop*, .

Pado, Sebastian, Tae-Gil Noh, Asher Stern & Rui Wang. 2013. Design and realization of a modular architecture for textual entailment. *Journal of Natural Language Engineering.* 21(2). 167–200. doi:10.1017/S1351324913000351.

Poliak, Adam, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger & Benjamin Van Durme. 2018. Hypothesis only baselines in natural language inference. In *Proceedings of the seventh joint conference on lexical and computational semantics*, 180–191. New Orleans, Louisiana: Association for Computational Linguistics. http://www.aclweb.org/anthology/S18-2023.

Rocktäschel, Tim, Edward Grefenstette, Karl Moritz Hermann, Tomás Kočiský & Phil Plunsom. 2016. Reasoning about entailment with neural attention. ArXiv:1509.06664.

Rudinger, Rachel, Chandler May & Benjamin Van Durme. 2017. Social bias in elicited natural language inferences. In *Proceedings of the first acl workshop on ethics in natural language processing*, 74–79. Valencia, Spain: Association for Computational Linguistics. http://www.aclweb.org/anthology/W17-1609.

Tsuchiya, Masatoshi. 2018. Performance impact caused by hidden bias of training data for recognizing textual entailment. In *Proceedings of the 11th language resources and evaluation conference*, Miyazaki, Japan: European Language Resource Association. https://www.aclweb.org/anthology/L18-1239.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser & Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan & R. Garnett (eds.), *Advances in neural information processing systems 30*, 5998–6008. Curran Associates, Inc. http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

Weston, Jason, Sumit Chopra & Antoine Bordes. 2015. Memory networks. In *Proceedings of ICLR 2015*, .

Williams, Adina, Nikita Nangia & Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 conference of the north american chapter of the association for computational linguistics: Human language technologies, volume 1 (long papers)*, 1112–1122. Association for Computational Linguistics. doi:10.18653/v1/N18-1101. http://aclweb.org/anthology/N18-1101.

References III

Young, Peter, Alice Lai, Micah Hodosh & Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics* 2. 67–78.

Zaenen, Annie, Lauri Karttunen & Richard Crouch. 2005. Local textual inference: Can it be defined or circumscribed? In *Proceedings of the ACL workshop on empirical modeling of semantic equivalence and entailment*, 31–36. Ann Arbor, MI: Association for Computational Linguistics. http://www.aclweb.org/anthology/W/W05/W05-1206.