Natural Language Inference

Christopher Potts

Stanford Linguistics

CS 224U: Natural language understanding
April 27 and 29

Overview

1. Overview
2. SNLI, MultiNLI, and Adversarial NLI
3. Hand-built features
4. nli.experiment
5. Sentence-encoding models
6. Chained models
7. Attention
8. Error analysis

Associated materials

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

2. Homework and bake-off: `hw_wordentail.ipynb`

3. Core readings: Bowman et al. 2015; Williams et al. 2018; Nie et al. 2019b; Rocktäschel et al. 2016

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

Simple examples

Premise	Relation	Hypothesis
A turtle danced.	entails	A turtle moved.
turtle	contradicts	linguist
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.

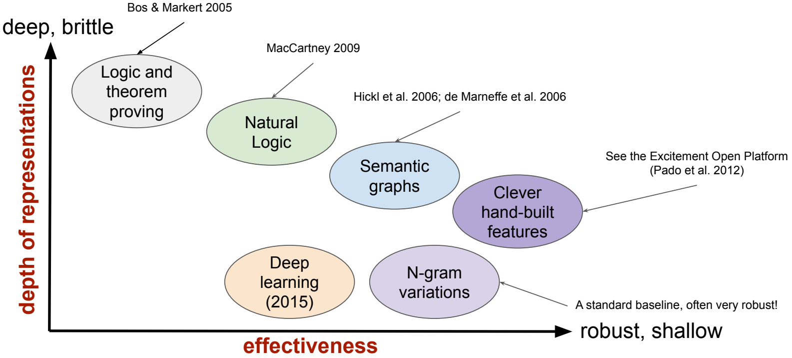
[. . .]

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.

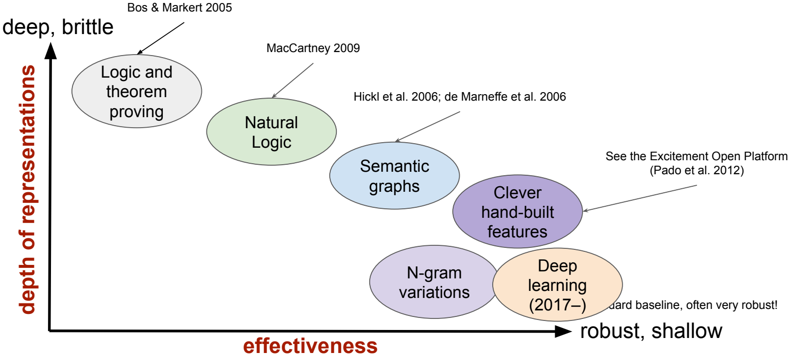
Connections to other tasks

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

Other NLI datasets

Recent

- The GLUE benchmark (diverse tasks including NLI)
  https://gluebenchmark.com
- NLI Style FEVER
  https://github.com/easonnie/combine-FEVER-NSMN/blob/master/other_resources/nli_fever.md
- 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/

Other NLI datasets

Older

- 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
- The FraCaS textual inference test suite
  https://nlp.stanford.edu/~wcmac/downloads/

Other NLI datasets

Related

- 30M Factoid Question-Answer Corpus
  `http://agarciaduran.org/`
- The Penn Paraphrase Database
  `http://paraphrase.org/`

Label sets

<table>
<tr><th></th><th>couch
sofa</th><th>crow
bird</th><th>bird
crow</th><th>hippo
hungry</th><th>turtle
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 ≡ Q
equivalence</td><td>P ⊏ Q
forward</td><td>P ⊐ Q
reverse</td><td colspan="2">P # Q
non-entailment</td></tr>
</table>

NLI dataset artifacts

1. **Artifact**: A dataset bias that would make a system susceptible to adversarial attack even if the bias is linguistically motivated.

2. Tricky example: negated hypotheses signal contradiction
    - Linguistically motivated: negation is our best way of establishing relevant contradictions.
    - An artifact because we would curate a dataset in which negation correlated with the other labels but led to no human confusion.

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; Belinkov et al. 2019)
- Likely due to artifacts:
  - 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.

Known artifacts in SNLI and MultiNLI

- These datasets contain words whose appearance nearly perfectly correlates with specific labels [1, 2].
- Entailment hypotheses over-represent general and approximating words [2].
- Neutral hypotheses often introduce modifiers [2].
- Contradiction hypotheses over-represent negation [1, 2].
- Neutral hypotheses tend to be longer [2].

1 = Poliak et al. 2018, 2 = Gururangan et al. 2018

Artifacts in other tasks

- Visual Question Answering: Kafle and Kanan 2017; Chen et al. 2020
- Story Completion: Schwartz et al. 2017
- Reading Comprehension/Question Answering: Kaushik and Lipton 2018
- Stance Detection: Schiller et al. 2020
- Fact Verification: Schuster et al. 2019

SNLI, MultiNLI, and Adversarial NLI

SNLI

1. Bowman et al. 2015
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

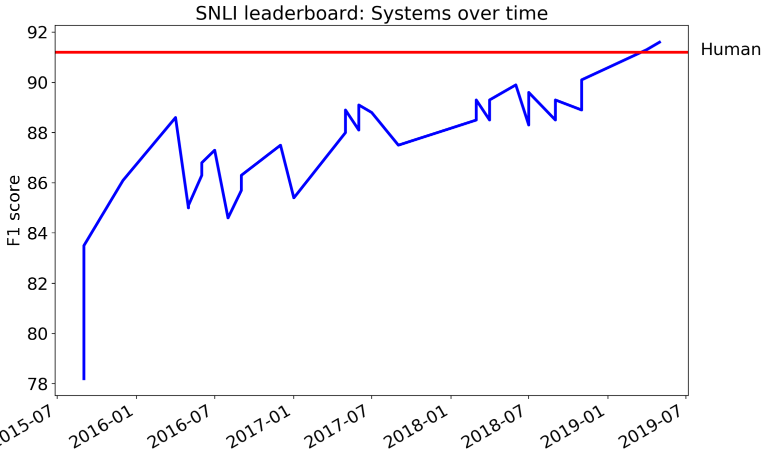
Premise	Relation	Hypothesis
A man inspects the uniform of a figure in some East Asian country.	**contradiction** c c c c c	The man is sleeping
An older and younger man smiling.	**neutral** n n e n n	Two men are smiling and laughing at the cats playing on the floor.
A black race car starts up in front 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

Progress on SNLI



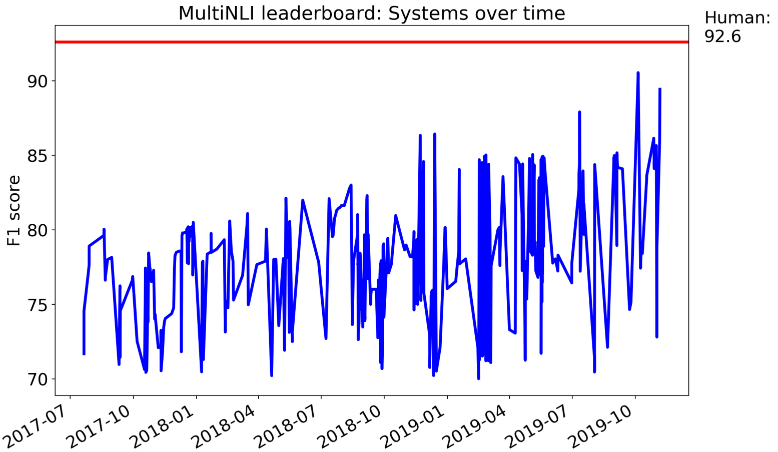
MultiNLI

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

Progress on MultiNLI



NLI adversarial testing

Premise	Relation	Hypothesis
A turtle danced.	entails	A turtle moved.
Every reptile danced.	neutral	A turtle ate.
Some turtles walk.	contradicts	No turtles move.

NLI adversarial testing

	Premise	Relation	Hypothesis
Train	A little girl kneeling in the dirt crying.	entails	A little girl is very sad.
Adversarial		entails	A little girl is very unhappy.

Glockner et al. 2018

NLI adversarial testing

	Premise	Relation	Hypothesis
Train	A **woman** is pulling a **child** on a sled in the snow.	entails	A child is sitting on a sled in the snow.
Adversarial	A **child** is pulling a **woman** on a sled in the snow.	neutral	

Nie et al. 2019a

Adversarial NLI dataset (ANLI)

1. Nie et al. 2019b
2. 162,865 labeled examples
3. The premises come from diverse sources.
4. The hypotheses are written by crowdworkers with the explicit goal of fooling state-of-the-art models.
5. This effort is a direct response to the results and findings for SNLI and MultiNLI that we just reviewed.

ANLI dataset creation

1. The annotator is presented with a premise sentence and a condition (entailment, contradiction, neutral).
2. The annotator writes a hypothesis.
3. A state-of-the-art model makes a prediction about the premise–hypothesis pair.
4. If the model's prediction matches the condition, the annotator returns to step 2 to try again.
5. If the model was fooled, the premise–hypothesis pair is independently validated by other annotators.

Additional ANLI details

Round	Model	Training data	Context sources	Examples
R1	BERT-large (Devlin et al. 2019)	SNLI + MultiNLI	Wikipedia	16,946
R2	ROBERTa (Liu et al. 2019)	SNLI + MultiNLI + NLI-FEVER + R1	Wikipedia	45,460
R3	ROBERTa (Liu et al. 2019)	SNLI + MultiNLI + NLI-FEVER + R2	Various	100,459
				162,865

- The train sets mix cases where the model's predictions were correct and incorrect. The majority of the model predictions are correct, though.
- The dev and test sets contain only cases where the model's prediction was incorrect.

Code snippets: Readers and Example objects

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

[2]:
```
SNLI_HOME = os.path.join("data", "nlidata", "snli_1.0")
MULTINLI_HOME = os.path.join("data", "nlidata", "multinli_1.0")
ANLI_HOME = os.path.join("data", "nlidata", "anli_v1.0")
```

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

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

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

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

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

[8]:
```
anli_train_reader = nli.ANLITrainReader(ANLI_HOME, rounds=(1,2,3))
```

[9]:
```
anli_dev_reader = nli.ANLIDevReader(ANLI_HOME, rounds=(1,2,3))
```

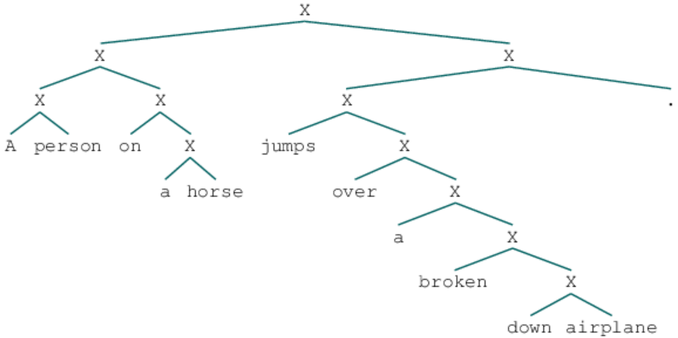
Code snippets: Examples

```
[10]: snli_iterator = iter(nli.SNLITrainReader(SNLI_HOME).read())

[11]: snli_ex = next(snli_iterator)

[12]: snli_ex.sentence1

[12]: 'A person on a horse jumps over a broken down airplane.'

[13]: snli_ex.sentence2

[13]: 'A person is training his horse for a competition.'

[14]: snli_ex.gold_label

[14]: 'neutral'
```

Code snippets: Examples

[15]:
```
snli_ex.sentence1_binary_parse
```

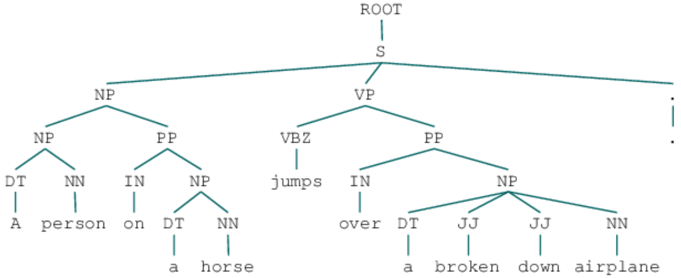
[15]:



Code snippets: Examples

[16]:
```
snli_ex.sentence1_parse
```

[16]:



Code snippets: MultiNLI annotations

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

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

[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")
```

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

[5]:
```
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

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

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

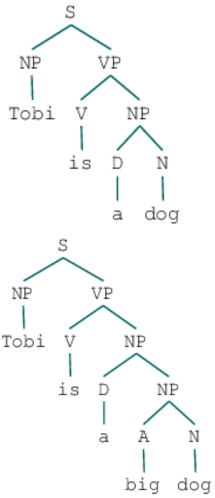
[3]:
```
def word_cross_product_phi(t1, t2):
    return Counter([(w1, w2) for w1, w2 in product(t1.leaves(), t2.leaves())])
```

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

[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

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

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

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

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

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

```
[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)
```

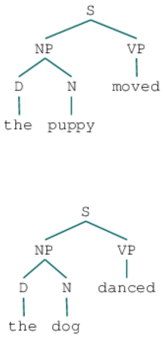
```
[5]: def hypernym_features(t1, t2):
         return wordnet_features(t1, t2, 'hypernyms')
```

```
[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 and Manning 2009)
7. Named entity features

nli.experiment

1. Overview
2. SNLI, MultiNLI, and Adversarial NLI
3. Hand-built features
4. nli.experiment
5. Sentence-encoding models
6. Chained models
7. Attention
8. Error analysis

Complete experiment with nli.experiment

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

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

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

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

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

[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

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

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

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

Hyperparameter selection on train subsets

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

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

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

[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
```

[5]:
```
# Select hyperparameters based on a subset of the data:
tuning_experiment_sample = 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

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

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

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

[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
```

[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

```
[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
```

```
[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

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

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

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

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

[5]: hypothesis_only_experiment = nli.experiment(
         nli.SNLITrainReader(SNLI_HOME),
         hypothesis_only_unigrams_phi,
         fit_softmax,
         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
accuracy			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

```
[6]: from sklearn.dummy import DummyClassifier
```

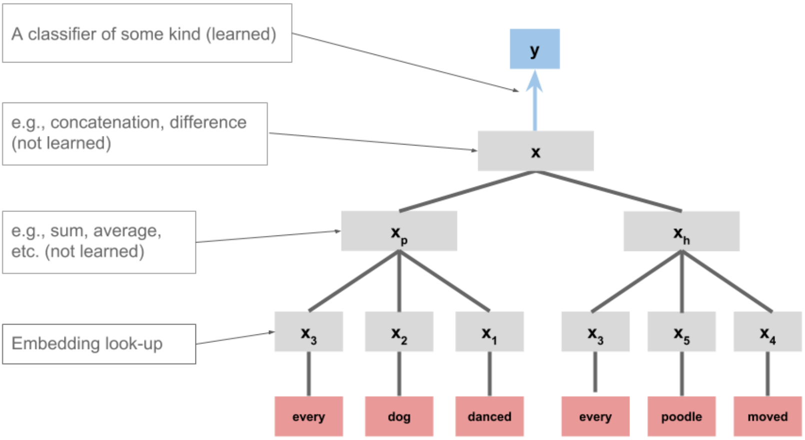
```
[7]: def fit_dummy_classifier(X, y):
         mod = DummyClassifier(strategy='stratified')
         mod.fit(X, y)
         return mod
```

```
[8]: 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      0.333     0.333     0.333      3278
   entailment      0.343     0.339     0.341      3329
      neutral      0.332     0.335     0.333      3235

     accuracy                          0.336      9842
    macro avg      0.336     0.336     0.336      9842
 weighted avg      0.336     0.336     0.336      9842
```

Sentence-encoding models

Distributed representations as features



Code: Distributed representations as features

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

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

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

[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
```

[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))
```

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

Code: Distributed representations as features

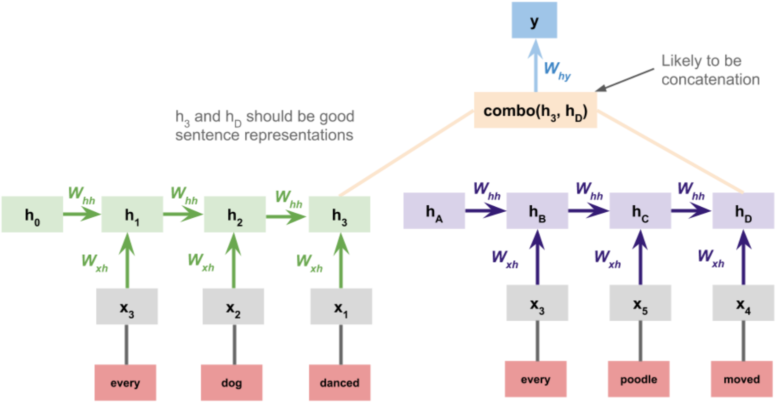
[7]:
```
def fit_softmax(X, y):
    mod = LogisticRegression(
        fit_intercept=True, solver='liblinear', multi_class='auto')
    mod.fit(X, y)
    return mod
```

[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!
```

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:

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

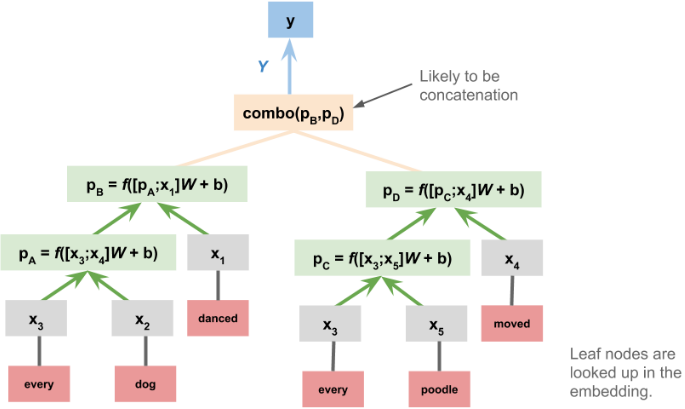
`TorchRNNSentenceEncoderClassifierModel`

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`

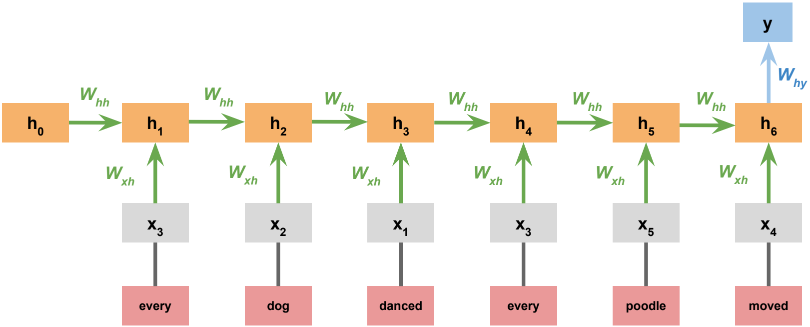
This is basically unchanged from its super class `TorchRNNClassifier`, except the `predict_proba` method needs to deal with the new example format.

Sentence-encoding TreeNNs



Chained models

Simple RNN


y
W_{hy}
W_{hh}
h_0
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

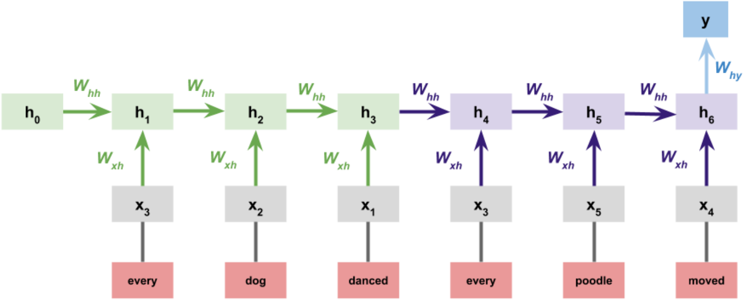
Rationale for sentence-encoding models

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

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

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

[3]: def simple_chained_rep_rnn_phi(t1, t2):
         return t1.leaves() + ["[SEP]"] + t2.leaves()

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

[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.

Other strategies

`TorchRNNClassifier`

- `TorchRNNClassifier` feeds its final hidden state directly to the classifier layer.
- If `bidirectional=True`, then the two final states are concatenated and fed directly to the classifier layer.

Other ideas

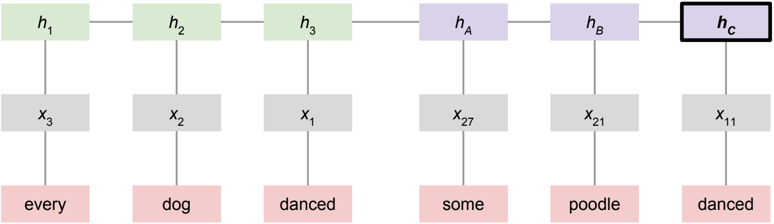
- *Pool* all the hidden states with **max** or **mean**.
- Different pooling options can be combined.
- Additional layers between the hidden representation (however defined) and the classifier layer.

Attention

Guiding ideas

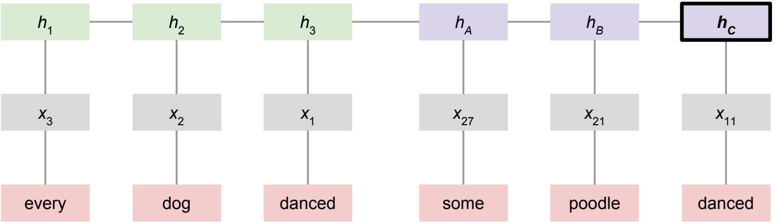
1. We need more connections between premise and hypothesis.
2. In processing the hypothesis, the model needs “reminders” of what the premise contained; the final premise hidden state isn’t enough.
3. Soft alignment between premise and hypothesis – a neural interpretation of an old idea in NLI.

Global attention



Global attention

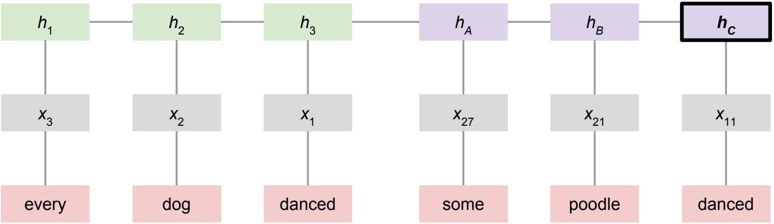
scores $$\tilde{\alpha} = \begin{bmatrix} h_C^\top h_1 & h_C^\top h_2 & h_C^\top h_3 \end{bmatrix}$$



Global attention

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

scores $\tilde{\alpha} = \left[\; h_C^\top h_1 \quad h_C^\top h_2 \quad h_C^\top h_3 \;\right]$

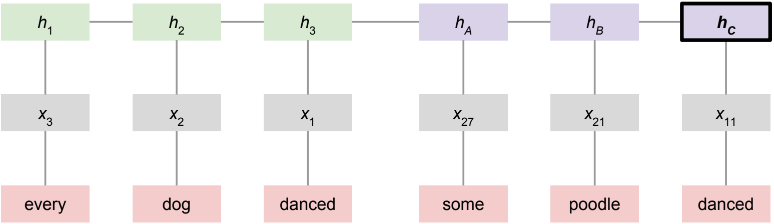


Global attention

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

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

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



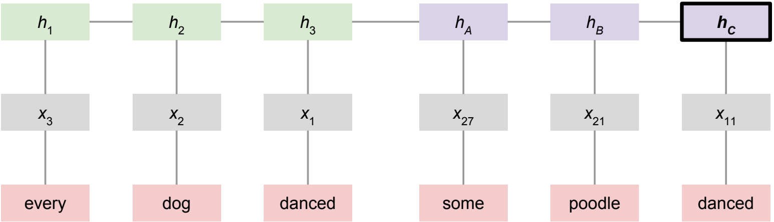
Global attention

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

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

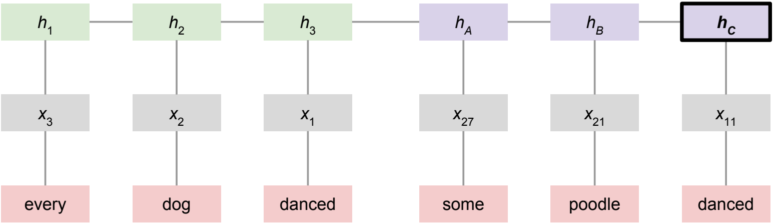
attention weights $\alpha = \mathbf{softmax}(\tilde{\alpha})$

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



Global attention

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

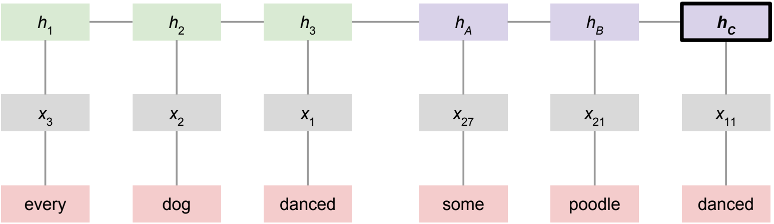
classifier $y = \textbf{softmax}(\tilde{h}W + b)$

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

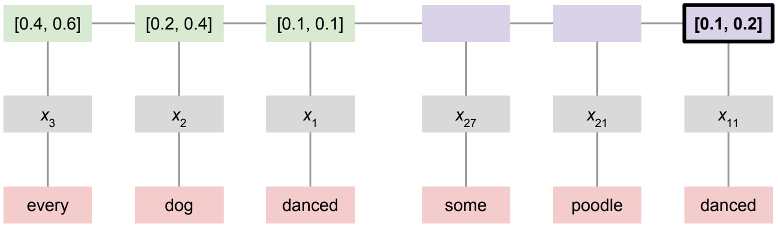
context $\kappa = \textbf{mean}(\alpha_1 h_1, \alpha_2 h_2, \alpha_3 h_3)$

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

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

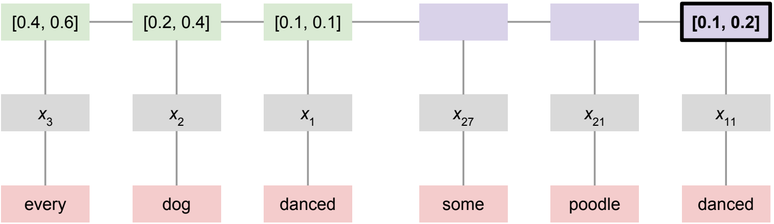


Global attention



Global attention

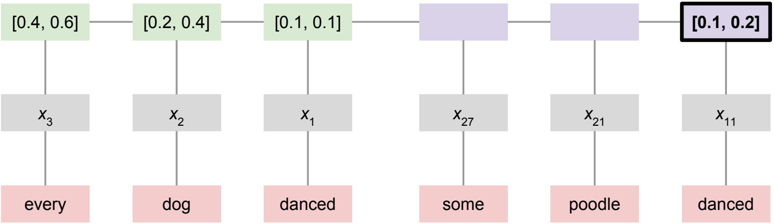
scores $\tilde{\alpha} = [0.16, 0.10, 0.03]$



Global attention

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

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

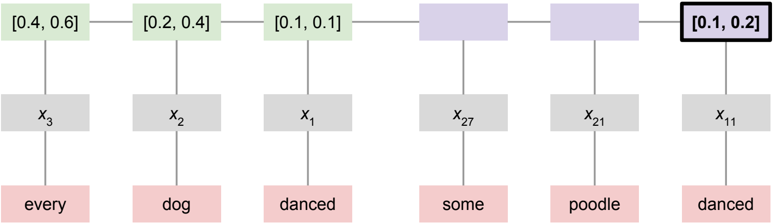


Global attention

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



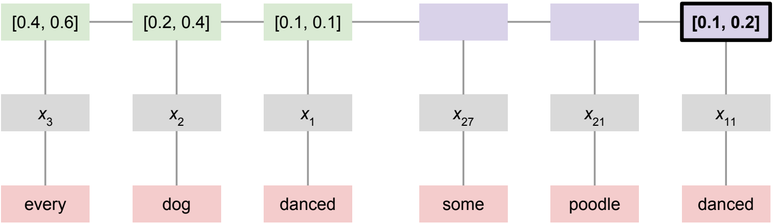
Global attention

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

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

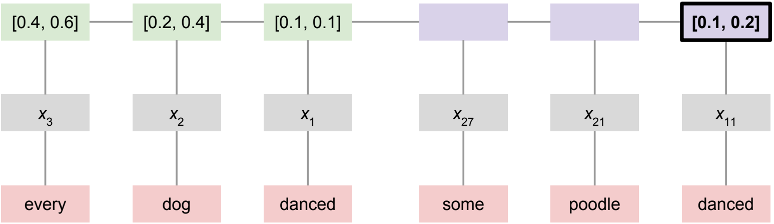
attention weights $\alpha = [0.35, 0.33, 0.31]$

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



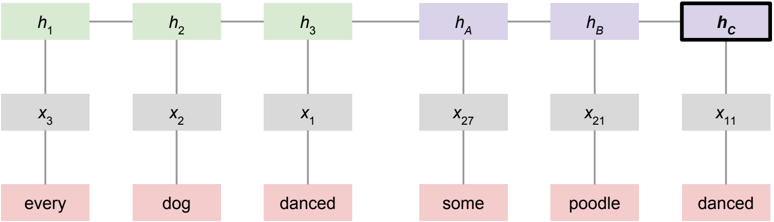
Global attention

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

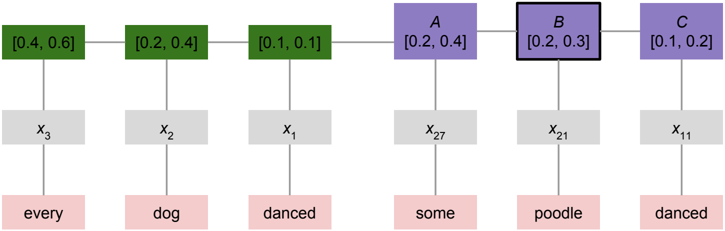


Other scoring functions (Luong et al. 2015)

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

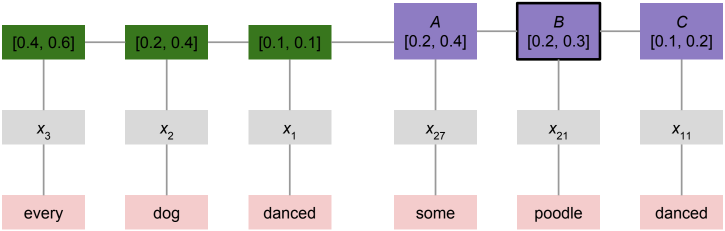


Word-by-word attention



Word-by-word attention

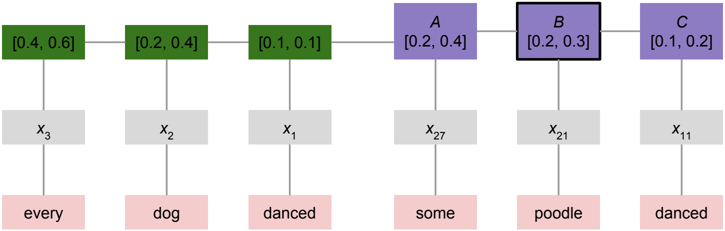
$$M = \tanh\left(\begin{bmatrix} 0.4 & 0.6 \\ 0.2 & 0.4 \\ 0.1 & 0.1 \end{bmatrix} + \begin{bmatrix} 0.2 & 0.3 & \kappa_A \\ 0.2 & 0.3 & \kappa_A \\ 0.2 & 0.3 & \kappa_A \end{bmatrix} W_h\right)$$



Word-by-word attention

weights at B $\quad \alpha_B = \textbf{softmax}(M{\color{orange}w})$

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

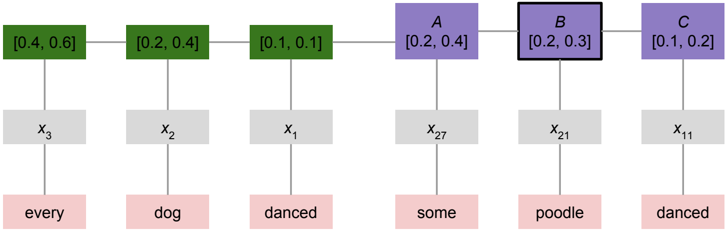


Word-by-word attention

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

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

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



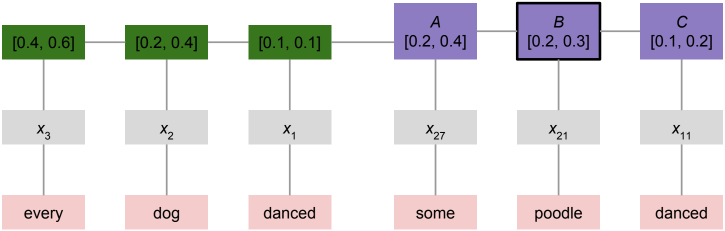
Word-by-word attention

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

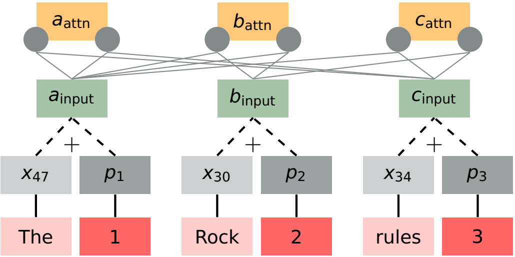
context at B $\kappa_B = \begin{bmatrix} 0.4 & 0.6 \\ 0.2 & 0.4 \\ 0.1 & 0.1 \end{bmatrix} \alpha_B + \tanh(\kappa_A W_\alpha)$

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

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



Connection with the Transformer



$$c_{\text{attn}} = \textbf{sum}\left([\alpha_1 a_{\text{input}}, \alpha_2 b_{\text{input}}]\right)$$

$$\alpha = \textbf{softmax}(\tilde{\alpha})$$

$$\tilde{\alpha} = \left[\frac{{c_{\text{input}}}^{\top} a_{\text{input}}}{\sqrt{d_k}}, \frac{{c_{\text{input}}}^{\top} b_{\text{input}}}{\sqrt{d_k}}\right]$$

$$c_{\text{input}} = x_{34} + p_3$$

Vaswani et al. 2017

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).
- Memory networks (Weston et al. 2015) can be used to address similar issues related to properly recalling past experiences.

Error analyses

1. Overview
2. SNLI, MultiNLI, and Adversarial NLI
3. Hand-built features
4. nli.experiment
5. Sentence-encoding models
6. Chained models
7. Attention
8. Error analysis

Systems compared

Model	Features
Logistic Regression	cross-product
Chained LSTM with deep classifier	GloVe
Fine-tuned ROBERTa	ROBERTa-large

The Logistic Regression implementation

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

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

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

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

[5]:
```
train_reader = nli.MultiNLITrainReader(MULTINLI_HOME)
```

[6]:
```
dev_reader = nli.MultiNLIMatchedDevReader(MULTINLI_HOME)
```

[7]:
```
experiment = nli.experiment(
    train_reader,
    word_cross_product_phi,
    fit_softmax,
    assess_reader=dev_reader,
    verbose=True)
```

The Chained LSTM implementation

```
[1]: import os
     import torch.nn as nn
     from torch_rnn_classifier import TorchRNNClassifier, TorchRNNClassifierModel
     import nli, utils
```

```
[2]: class DeepRNNClassifierModel(TorchRNNClassifierModel):
         def __init__(self, *args, **kwargs):
             super().__init__(*args, **kwargs)
             drop_prob = 0.1
             self.dropout = nn.Dropout(p=drop_prob)
             self.relu = nn.ReLU()
             self.bidirectional = kwargs['bidirectional']
             self.hidden_dim = kwargs['hidden_dim']
             if self.bidirectional:
                 classifier_dim = self.hidden_dim * 2
             else:
                 classifier_dim = self.hidden_dim
             self.mlp_layer = nn.Linear(classifier_dim, classifier_dim)

         def forward(self, X, seq_lengths):
             state = self.rnn_forward(X, seq_lengths, self.rnn)
             h = self.relu(self.mlp_layer(state))
             h = self.dropout(h)
             logits = self.classifier_layer(h)
             return logits

     class DeepRNNClassifier(TorchRNNClassifier):
         def build_graph(self):
             return DeepRNNClassifierModel(
                 vocab_size=len(self.vocab),
                 embedding=self.embedding,
                 use_embedding=self.use_embedding,
                 embed_dim=self.embed_dim,
                 hidden_dim=self.hidden_dim,
                 output_dim=self.n_classes_,
                 bidirectional=self.bidirectional,
                 device=self.device)
```

Inspired by the BiLSTM of Williams et al. 2018

The Chained LSTM implementation

```
[3]: utils.fix_random_seeds()

[4]: GLOVE_HOME = os.path.join("data", 'glove.6B')
     MULTINLI_HOME = os.path.join("data", "nlidata", "multinli_1.0")

[5]: SEP = "[SEP]"

[6]: def chained_rnn_phi(t1, t2):
         return t1.leaves() + [SEP] + t2.leaves()

[7]: glove_lookup = utils.glove2dict(os.path.join(GLOVE_HOME, 'glove.840B.300d.txt'))

[8]: def fit_deep_rnn(X, y):
         vocab = utils.get_vocab(X)
         glove_embedding, glove_vocab = utils.create_pretrained_embedding(
             glove_lookup, vocab, required_tokens=('$UNK', SEP))
         mod = DeepRNNClassifier(
             glove_vocab,
             embedding=glove_embedding,
             embed_dim=300,
             hidden_dim=300,
             bidirectional=True,
             max_iter=8,
             eta=0.0004,
             l2_strength=0.00001,
             batch_size=16,
             warm_start=True)
         mod.fit(X, y)
         return mod

[9]: train_reader = nli.MultiNLITrainReader(MULTINLI_HOME)

[10]: dev_reader = nli.MultiNLIMatchedDevReader(MULTINLI_HOME)

[11]: basic_experiment = nli.experiment(
          train_reader,
          chained_rnn_phi,
          fit_deep_rnn,
          assess_reader=dev_reader,
          vectorize=False,
          verbose=True)
```

Inspired by
the BiLSTM of
Williams et al. 2018

The ROBERTa implementation

```
[1]: import nli, os
     import torch
     from sklearn.metrics import classification_report

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

[3]: model = torch.hub.load('pytorch/fairseq', 'roberta.large.mnli')
     _ = model.eval()

     Using cache found in /Users/cgpotts/.cache/torch/hub/pytorch_fairseq_master

[4]: dev = [((ex.sentence1, ex.sentence2), ex.gold_label)
            for ex in nli.MultiNLIMatchedDevReader(MULTINLI_HOME).read()]

[5]: X_dev_str, y_dev = zip(*dev)

[6]: X_dev = [model.encode(*ex) for ex in X_dev_str]

[7]: %time pred_indices = [model.predict('mnli', ex).argmax() for ex in X_dev]

     CPU times: user 1h 45min 44s, sys: 3min 44s, total: 1h 49min 28s
     Wall time: 27min 23s

[8]: to_str = {0: 'contradiction', 1: 'neutral', 2: 'entailment'}

[9]: preds = [to_str[c.item()] for c in pred_indices]

[10]: print(classification_report(y_dev, preds))
```

https://github.com/pytorch/fairseq/tree/master/examples/roberta

Dev-set score comparisons

Model	Precision	Recall	Macro-F1
Logistic regression	59.2	59.0	59.1
Chained LSTM with deep classifier	68.1	67.1	67.3
Fine-tuned ROBERTa	90.5	90.5	90.5

MultiNLI annotations

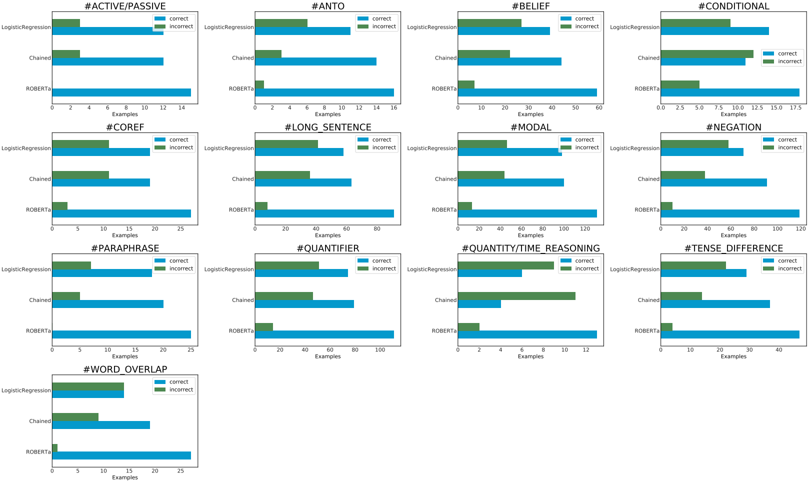
Annotations	Premise	Relation	Hypothesis
#MODAL, #COREF	Students of human misery can savor its underlying sadness and futility. entailment	entailment	Those who study human misery will savor the sadness and futility.
#NEGATION, #TENSE_ DIFFERENCE, #CONDITIONAL	oh really it wouldn't matter if we plant them 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.

Annotation-set score comparisons

Model	Precision	Recall	Macro-F1
Logistic regression	58.5	58.0	58.0
Chained LSTM with deep classifier	69.3	68.3	68.4
Fine-tuned ROBERTa	91.9	91.9	91.9

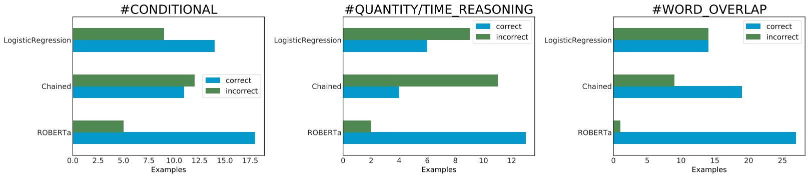
MultiNLI annotations: Results by category

All results



MultiNLI annotations: Results by category

Most challenging categories



Testing for specific patterns

Does your model know that negation is downward monotone?

Fido moved.	Fido did**n't** move.
↑	↓
Fido ran.	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.

↙ ↖

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

Does your model *systematically* capture such patterns?

Testing with adversarial test sets

Testing with adversarial test sets

One-word changes to SNLI hypotheses using structured resources; labels separately validated by crowdworkers.

	Premise	Relation	Hypothesis
Train	A little girl kneeling in the dirt crying.	entails	A little girl is very sad.
Adversarial		entails	A little girl is very unhappy.

Glockner et al. 2018

Testing with adversarial test sets

One-word changes to SNLI hypotheses using structured resources; labels separately validated by crowdworkers.

Contradiction	7,164
Entailment	982
Neutral	47
Total	8,193

Category	Examples
antonyms	1147
synonyms	894
cardinals	759
nationalities	755
drinks	731
antonyms_wordnet	706
colors	699
ordinals	663
countries	613
rooms	595
materials	397
vegetables	109
instruments	65
planets	60

Glockner et al. 2018

Testing with adversarial test sets

One-word changes to SNLI hypotheses using structured resources; labels separately validated by crowdworkers.

Model	Train set	SNLI test set	New test set	Δ
Decomposable Attention (Parikh et al., 2016)	SNLI	84.7%	51.9%	-32.8
	MultiNLI + SNLI	84.9%	65.8%	-19.1
	SciTail + SNLI	85.0%	49.0%	-36.0
ESIM (Chen et al., 2017)	SNLI	87.9%	65.6%	-22.3
	MultiNLI + SNLI	86.3%	74.9%	-11.4
	SciTail + SNLI	88.3%	67.7%	-20.6
Residual-Stacked-Encoder (Nie and Bansal, 2017)	SNLI	86.0%	62.2%	-23.8
	MultiNLI + SNLI	84.6%	68.2%	-16.8
	SciTail + SNLI	85.0%	60.1%	-24.9
WordNet Baseline	-	-	85.8%	-
KIM (Chen et al., 2018)	SNLI	88.6%	83.5%	-5.1

Table 3: Accuracy of various models trained on SNLI or a union of SNLI with another dataset (MultiNLI, SciTail), and tested on the original SNLI test set and the new test set.

Glockner et al. 2018

Testing with adversarial test sets

```
[1]: import nli, os, torch
     from sklearn.metrics import classification_report

[2]: # Available from https://github.com/BIU-NLP/Breaking_NLI:
     breaking_nli_src_filename = os.path.join("../new-data/data/dataset.jsonl")
     reader = nli.NLIReader(breaking_nli_src_filename)

[3]: exs = [((ex.sentence1, ex.sentence2), ex.gold_label) for ex in reader.read()]

[4]: X_test_str, y_test = zip(*exs)

[5]: model = torch.hub.load('pytorch/fairseq', 'roberta.large.mnli')
     _ = model.eval()

     Using cache found in /Users/cgpotts/.cache/torch/hub/pytorch_fairseq_master

[6]: X_test = [model.encode(*ex) for ex in X_test_str]

[7]: pred_indices = [model.predict('mnli', ex).argmax() for ex in X_test]

[8]: to_str = {0: 'contradiction', 1: 'neutral', 2: 'entailment'}

[9]: preds = [to_str[c.item()] for c in pred_indices]
```

https://github.com/pytorch/fairseq/tree/master/examples/roberta

Testing with adversarial test sets

```
[10]: print(classification_report(y_test, preds))

                   precision    recall  f1-score   support

    contradiction       0.99      0.97      0.98      7164
       entailment       0.86      1.00      0.92       982
          neutral       0.15      0.15      0.15        47

         accuracy                           0.97      8193
        macro avg       0.67      0.71      0.68      8193
     weighted avg       0.97      0.97      0.97      8193
```

https://github.com/pytorch/fairseq/tree/master/examples/roberta

References I

Yonatan Belinkov, Adam Poliak, Stuart Shieber, Benjamin Van Durme, and Alexander Rush. 2019. Don't take the premise for granted: Mitigating artifacts in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 877–891, Florence, Italy. Association for Computational Linguistics.

Johan Bos and 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*, pages 628–635, Stroudsburg, PA. ACL.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Stroudsburg, PA. Association for Computational Linguistics.

Long Chen, Xin Yan, Jun Xiao, Hanwang Zhang, Shiliang Pu, and Yueting Zhuang. 2020. Counterfactual samples synthesizing for robust visual question answering. ArXiv:2003.06576.

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

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL recognising textual entailment challenge. In *Machine Learning Challenges, Lecture Notes in Computer Science*, volume 3944, pages 177–190. Springer-Verlag.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking NLI systems with sentences that require simple lexical inferences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 650–655, Melbourne, Australia. Association for Computational Linguistics.

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

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and 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)*, pages 107–112, New Orleans, Louisiana. Association for Computational Linguistics.

Andrew Hickl and Jeremy Bensley. 2007. A discourse commitment-based framework for recognizing textual entailment. In *Proceedings of the Workshop on Textual Entailment and Paraphrasing*.

Kushal Kafle and Christopher Kanan. 2017. Visual question answering: Datasets, algorithms, and future challenges. *Computer Vision and Image Understanding*, 163:3–20.

Divyansh Kaushik and Zachary C. Lipton. 2018. How much reading does reading comprehension require? a critical investigation of popular benchmarks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5010–5015, Brussels, Belgium. Association for Computational Linguistics.

References II

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. ROBERTa: A robustly optimized BERT pretraining approach. ArXiv:1907.11692.

Thang Luong, Hieu Pham, and 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*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Bill MacCartney. 2009. *Natural Language Inference*. Ph.D. thesis, Stanford University.

Bill MacCartney and 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)*, pages 521–528, Manchester, UK. Coling 2008 Organizing Committee.

Bill MacCartney and Christopher D. Manning. 2009. An extended model of natural logic. In *Proceedings of the Eighth International Conference on Computational Semantics*, pages 140–156, Tilburg, The Netherlands. Association for Computational Linguistics.

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

Marie-Catherine de Marneffe, Bill MacCartney, Trond Grenager, Daniel Cer, Anna Rafferty, and Christopher D Manning. 2006. Learning to distinguish valid textual entailments. In *Proceedings of the 2nd Pascal RTE Challenge Workshop*.

Yixin Nie, Yicheng Wang, and Mohit Bansal. 2019a. Analyzing compositionality-sensitivity of NLI models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6867–6874.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2019b. Adversarial NLI: A new benchmark for natural language understanding. UNC CHapel Hill and Facebook AI Research.

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

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis only baselines in natural language inference. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 180–191, New Orleans, Louisiana. Association for Computational Linguistics.

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

Rachel Rudinger, Chandler May, and Benjamin Van Durme. 2017. Social bias in elicited natural language inferences. In *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*, pages 74–79, Valencia, Spain. Association for Computational Linguistics.

Benjamin Schiller, Johannes Daxenberger, and Iryna Gurevych. 2020. Stance detection benchmark: How robust is your stance detection? ArXiv:2001.01565.

Tal Schuster, Darsh J Shah, Yun Jie Serene Yeo, Daniel Filizzola, Enrico Santus, and Regina Barzilay. 2019. Towards debiasing fact verification models. ArXiv:1908.05267.

References III

Roy Schwartz, Maarten Sap, Ioannis Konstas, Leila Zilles, Yejin Choi, and Noah A. Smith. 2017. Story cloze task: UW NLP system. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, pages 52–55, Valencia, Spain. Association for Computational Linguistics.

Masatoshi Tsuchiya. 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.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

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

Adina Williams, Nikita Nangia, and 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)*, pages 1112–1122. Association for Computational Linguistics.

Peter Young, Alice Lai, Micah Hodosh, and 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.

Annie Zaenen, Lauri Karttunen, and 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*, pages 31–36, Ann Arbor, MI. Association for Computational Linguistics.