Posing Fair Generalization Tasks for Natural Language Inference

Atticus Geiger
Stanford Symbolic Systems Program
atticusg@stanford.edu

Ignacio Cases
Stanford Linguistics
cases@stanford.edu

Lauri Karttunen
Stanford Linguistics
laurik@stanford.edu

Christopher Potts
Stanford Linguistics
cgpotts@stanford.edu

Abstract

Deep learning models for semantics are generally evaluated using naturalistic corpora. Adversarial methods, in which models are evaluated on new examples with known semantic properties, have begun to reveal that good performance at these naturalistic tasks can hide serious shortcomings. However, we should insist that these evaluations be *fair* – that the models are given data sufficient to support the requisite kinds of generalization. In this paper, we define and motivate a formal notion of fairness in this sense. We then apply these ideas to natural language inference by constructing very challenging but provably fair artificial datasets and showing that standard neural models fail to generalize in the required ways; only task-specific models that jointly compose the premise and hypothesis are able to achieve high performance, and even these models do not solve the task perfectly.

1 Introduction

Evaluations of deep learning approaches to semantics generally rely on corpora of naturalistic examples, with quantitative metrics serving as a proxy for the underlying capacity of the models to learn rich meaning representations and find generalized solutions. From this perspective, when a model achieves human-level performance on a task according to a chosen metric, one might be tempted to say that the task is "solved". However, recent adversarial testing methods, in which models are evaluated on new examples with known semantic properties, have begun to reveal that even these state-of-the-art models often rely on brittle, local solutions that fail to generalize even to examples that are similar to those they saw in training. These findings indicate that we need a broad and deep range of evaluation methods to fully characterize the capacities of our models.

However, for any evaluation method, we should ask whether it is *fair*. Has the model been shown data sufficient to support the kind of generalization we are asking of it? Unless we can say "yes" with complete certainty, we can't be sure whether a failed evaluation traces to a model limitation or a data limitation that no model could overcome.

In this paper, we seek to address this issue by defining a formal notion of fairness for these evaluations. The definition is quite general and can be used to create fair evaluations for a wide range of tasks. We apply it to Natural Language Inference (NLI) by constructing very challenging but provably fair artificial datasets. We evaluate a number of different standard architectures (variants of LSTM sequence models with attention and tree-structured neural networks) as well as NLI-specific tree-structured neural networks that process aligned examples. Our central finding is that only task-specific models are able to achieve high performance, and even these models do not solve the task perfectly, calling into question the viability of the standard models for semantics.

2 Related Work

There is a growing literature that uses targeted generalization tasks to probe the capacity of learning models. We seek to build on this work by developing a formal framework in which one can ask whether one of these tasks is even possible.

In adversarial testing, training examples are systematically perturbed and then used for testing. In computer vision, it is common to adversarially train on artificially noisy examples to create a more robust model (Goodfellow et al., 2015; Szegedy et al., 2014). However, in the case of question answering, Jia and Liang (2017) show that training on one perturbation does not result in generalization to similar perturbations, revealing a

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 4485–4495,
Hong Kong, China, November 3–7, 2019.

need for models with stronger generalization capabilities. Similarly, adversarial testing has shown that strong models for the SNLI dataset (Bowman et al., 2015a) have significant holes in their knowledge of lexical and compositional semantics (Glockner et al., 2018; Naik et al., 2018; Nie et al., 2018; Yanaka et al., 2019; Dasgupta et al., 2018). In addition, a number of recent papers suggest that even top models exploit dataset artifacts to achieve good quantitative results (Poliak et al., 2018; Gururangan et al., 2018; Tsuchiya, 2018), which further emphasizes the need to go beyond naturalistic evaluations.

Artificially generated datasets have also been used extensively to gain analytic insights into what models are learning. These methods have the advantage that the complexity of individual examples can be precisely characterized without reference to the models being evaluated. Evans et al. (2018) assess the ability of neural models to learn propositional logic entailment. Bowman et al. (2015b) conduct similar experiments using natural logic, and Veldhoen and Zuidema (2018) analyze models trained on those same tasks, arguing that they fail to discover the kind of global solution we would expect if they had truly learned natural logic. Lake and Baroni (2017) apply similar methods to instruction following with an artificial language describing a simple domain.

These methods can provide powerful insights, but the issue of fairness looms large. For instance, Bowman (2013) poses generalization tasks in which entire reasoning patterns are held out for testing. Similarly, Veldhoen and Zuidema (2018) assess a model's ability to recognize De Morgan's laws without any exposure to this reasoning in training. These extremely difficult tasks break from standard evaluations in an attempt to expose model limitations. However, these tasks are not fair by our standards; brief formal arguments for these claims are given in Appendix A.

3 Compositionality and Generalization

Many problems can be solved by recursively composing intermediate representations with functions along a tree structure. In the case of arithmetic, the intermediate representations are numbers and the functions are operators such a plus or minus. In the case of evaluating the truth of propositional logic sentences, the intermediate representation are truth values and the functions are logical operators such as disjunction, negation, or the material conditional. We will soon see that, in the case of NLI, the intermediate representations are semantic relations between phrases and the functions are semantic operators such as quantifiers or negation. When tasked with learning some compositional problem, we intuitively would expect to be shown how every function operates on every intermediate value. Otherwise, some functions would be underdetermined. We now formalize the idea of recursive tree-structured composition and this intuitive notion of fairness.

Data: A composition tree $C = (T, \mathit{Dom}, \mathit{Func})$, a node $a \in N^T$, and an input $x \in \mathcal{I}_C$
Result: An output from $\mathit{Dom}(a)$

function $\texttt{compose}\,(C, a, x)$
  if $a \in N^T_{\mathit{leaf}}$ **then**
    $i \leftarrow \texttt{index}(a, T)$
    return x_i
  else
    $c_1, \ldots c_m \leftarrow \texttt{children}(a, T)$
    return $\mathit{Func}(a)($
      $\texttt{compose}(C, c_1, x), \ldots,$
      $\texttt{compose}(C, c_m, x))$
  end

Algorithm 1: Recursive composition up a tree. This algorithm uses helper functions $\texttt{children}(a, T)$, which returns the left-to-right ordered children of node a, and $\texttt{index}(a, T)$, which returns the index of a leaf according to left-to-right ordering.

We first define composition trees (Section 3.1) and show how these naturally determine baseline learning models (Section 3.2). These models implicitly define a property of fairness: a train/test split is fair if the baseline model learns the task perfectly (Section 3.3). This enables us to create provably fair NLI tasks in Section 4.

3.1 Composition Trees

A composition tree describes how to recursively compose elements from an input space up a tree structure to produce an element in an output space. Our baseline learning model will construct a composition tree using training data.

Definition 1. (Composition Tree) Let T be an ordered tree with nodes $N^T = N^T_{\text{leaf}} \cup N^T_{\text{non-leaf}}$, where N^T_{leaf} is the set of leaf nodes and $N^T_{\text{non-leaf}}$ is the set of non-leaf nodes for T. Let *Dom* be a map on N^T that assigns a set to each node, called the *domain* of the node. Let *Func* be a map on

Data: An ordered tree T and a set of training data $\mathcal{D}$ containing pairs (x, Y) where x is an input and Y is a function defined on $N^T_{\text{non-leaf}}$ providing labels at every node of T.
Result: A composition tree $(T, \mathit{Dom}, \mathit{Func})$

function `learn` $(T, \mathcal{D})$
  $\mathit{Dom}, \mathit{Func} \leftarrow$ `initialize`(T)
  for $(x, Y) \in \mathcal{D}$ **do**
    $\mathit{Dom}, \mathit{Func} \leftarrow$ `memorize`$(x, Y, T, \mathit{Dom}, \mathit{Func}, r)$
  end
  return $(T, \mathit{Dom}, \mathit{Func})$

function `memorize` *(x,Y,T,Dom,Func,a)*
  if $a \in N^T_{\mathit{leaf}}$ **then**
    $i \leftarrow$ `index`(a, T)
    $\mathit{Dom}[a] \leftarrow \mathit{Dom}[a] \cup \{x_i\}$
    return *Dom*, *Func*
  else
    $\mathit{Dom}[a] \leftarrow \mathit{Dom}[a] \cup \{Y(a)\}$
    $c_1, \ldots, c_m \leftarrow$ `children`(a, T)
    $\mathit{Func}[a][(Y(c_1), \ldots, Y(c_m))] \leftarrow Y(a)$
    for $k \leftarrow 1 \ldots m$ **do**
      $\mathit{Dom}, \mathit{Func} \leftarrow$ `memorize`$(x, Y, T, \mathit{Dom}, \mathit{Func}, c_k)$
    end
    return *Dom*, *Func*
  end

Algorithm 2: Given a tree and training data with labels for every node of the tree, this learning model constructs a composition tree. This algorithm uses helper functions `children`(a, T) and `index`(a, T), as in Algorithm 1, as well as `initialize`(T), which returns *Dom*, a dictionary mapping N^T to empty sets, and *Func*, a dictionary mapping $N^T_{\text{non-leaf}}$ to empty dictionaries.

$N^T_{\text{non-leaf}}$ that assigns a function to each non-leaf node satisfying the following property: For any $a \in N^T_{\text{non-leaf}}$ with left-to-right ordered children $c_1, \ldots, c_m$, we have that $\mathit{Func}(a) : \mathit{Dom}(c_1) \times \cdots \times \mathit{Dom}(c_m) \to \mathit{Dom}(a)$. We refer to the tuple $C = (T, \mathit{Dom}, \mathit{Func})$ as a *composition tree*. The *input space* of this composition tree is the cartesian product $\mathcal{I}_C = \mathit{Dom}(l_1) \times \cdots \times \mathit{Dom}(l_k)$, where $l_1, \ldots, l_k$ are the leaf nodes in left-to-right order, and the *output space* is $\mathcal{O}_C = \mathit{Dom}(r)$ where r is the root node.

A composition tree $C = (T, \mathit{Dom}, \mathit{Func})$ realizes a function $F : \mathcal{I}_C \to \mathcal{O}_C$ in the following way: For any input $x \in \mathcal{I}_C$, this function is given by $F(x) = \texttt{compose}(C, r, x)$, where r is the root node of T and `compose` is defined recursively in Algorithm 1. For a given $x \in \mathcal{I}_C$ and $a \in N^T_{\text{non-leaf}}$ with children $c_1, \ldots, c_m$, we say that the element of $\mathit{Dom}(c_1) \times \cdots \times \mathit{Dom}(c_m)$ that is input to $\mathit{Func}(a)$ during the computation of $F(x) = \texttt{compose}(C, r, x)$ is the input realized at $\mathit{Func}(a)$ on x and the element of $\mathit{Dom}(a)$ that is output by $\mathit{Func}(a)$ is the output realized at $\mathit{Func}(a)$ on x. At a high level, $\texttt{compose}(C, a, x)$ finds the output realized at a node a by computing node a's function $\mathit{Func}(a)$ with the outputs realized at node a's children as inputs. This recursion bottoms out when the components of x are provided as the outputs realized at leaf nodes.

3.2 A Baseline Learning Model

Algorithm 2 is our baseline learning model. It learns a function by constructing a composition tree. This is equivalent to learning the function that tree realizes, as once the composition tree is created, Algorithm 1 computes the realized function. Because this model constructs a composition tree, it has an inductive bias to recursively compute intermediate representations up a tree structure. At a high level, it constructs a full composition tree when provided with the tree structure and training data that provides a label at every node in the tree by looping through training data inputs and memorizing the output realized at each intermediate function for a given input. As such, any learning model we compare to this baseline model should be provided with the outputs realized at every node during training.

3.3 Fairness

We define a training dataset to be fair with respect to some function F if our baseline model perfectly learns the function F from that training data. The guiding idea behind fairness is that the training data must expose every intermediate function of a composition tree to every possible intermediate input, allowing the baseline model to learn a global solution:

Definition 2. (A Property Sufficient for Fairness) A property of a training dataset $\mathcal{D}$ and tree T that is sufficient for fairness with respect to a function F is that there exists a composition tree $C = (T, \mathit{Dom}, \mathit{Func})$ realizing F such that, for any $a \in N^T_{\text{non-leaf}}$ and for any input i to $\mathit{Func}(a)$, there exists $(x, Y) \in \mathcal{D}$ where i is the input realized at $\mathit{Func}(a)$ on x.

Not all fair datasets are challenging. For example, a scenario in which one trains and tests on the entire space of examples will be fair. The role of fairness is to ensure that, when we separately de-

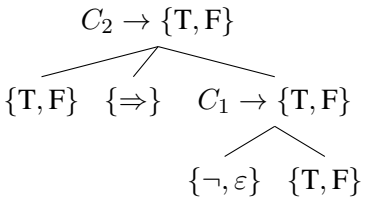

Figure 1: A composition tree that realizes a function evaluating propositional sentences. We define the functions $C_1(U, V_1) = U(V_1)$ and $C_2(V_1, \Rightarrow, V_2) = V_1 \Rightarrow V_2$ where $V_1, V_2 \in \{\mathrm{T}, \mathrm{F}\}$ and $U \in \{\neg, \varepsilon\}$.

Train	**Test**
T $\Rightarrow$ ε F	T $\Rightarrow$ $\neg$ T
T $\Rightarrow$ $\neg$ F	T $\Rightarrow$ ε T
F $\Rightarrow$ $\neg$ T	F $\Rightarrow$ $\neg$ F
F $\Rightarrow$ ε T	F $\Rightarrow$ ε F

Table 1: A fair train/test split for the evaluation problem defined by the composition tree in Figure 1. We give just the terminal nodes; the examples are full trees.

fine a challenging task, it is guaranteed to be possible. We noted in Section 2 that some challenging problems in the literature fail to meet this minimal requirement.

3.4 Fair Tasks for Propositional Evaluation

As a simple illustration of the above concepts, we consider the task of evaluating the truth of a sentence from propositional logic. We use the standard logical operators material conditional, $\Rightarrow$, and negation, $\neg$, as well as the unary operator ε, which we define to be the identity function on $\{\mathrm{T}, \mathrm{F}\}$. We consider a small set of eight propositional sentences, all which can be seen in Table 1. We illustrate a composition tree that realizes a function performing truth evaluation on these sentences in Figure 1, where a leaf node l is labeled with its domain $\mathit{Dom}(l)$ and a non-leaf node a is labeled with $\mathit{Func}(a) \rightarrow \mathit{Dom}(a)$.

A dataset for this problem is fair if and only if it has two specific properties. First, the binary operator $\Rightarrow$ must be exposed to all four inputs in $\{\mathrm{T}, \mathrm{F}\} \times \{\mathrm{T}, \mathrm{F}\}$ during training. Second, the unary operators $\neg$ and ε each must be exposed to both inputs in $\{\mathrm{T}, \mathrm{F}\}$. Jointly, these constraints ensure that a model will see all the possibilities for how our logical operators interact with their truth-value arguments. If either constraint is not met, then there is ambiguity about which operators the model is tasked with learning. An example fair train/test split is given in Table 1.

Crucial to our ability to create a fair training dataset using only four of the eight sentences is that $\Rightarrow$ operates on the intermediate representation of a truth value, abstracting away from the specific identity of its sentence arguments. Because there are two ways to realize T and F at the intermediate node, we can efficiently use only half of our sentences to satisfy our fairness property.

symbol	example	set theoretic definition
$x \equiv y$	couch $\equiv$ sofa	$x = y$
$x \sqsubset y$	crow $\sqsubset$ bird	$x \subset y$
$x \sqsupset y$	bird $\sqsupset$ crow	$x \supset y$
$x \mathbin{^\wedge} y$	human $^\wedge$ nonhuman	$x \cap y = \emptyset \wedge x \cup y = U$
$x \mid y$	cat $\mid$ dog	$x \cap y = \emptyset \wedge x \cup y \neq U$
$x \smile y$	animal $\smile$ nonhuman	$x \cap y \neq \emptyset \wedge x \cup y = U$
$x \# y$	hungry # hippo	(all other cases)

Table 2: The seven basic semantic relations of MacCartney and Manning (2009): $\mathcal{B}$ = {# = independence, $\sqsubset$ = entailment, $\sqsupset$ = reverse entailment, $\mid$ = alternation, $\smile$ = cover, $^\wedge$ = negation, $\equiv$ = equivalence}.

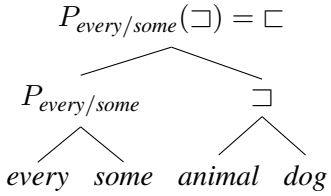

Figure 2: Natural logic inference cast as composition on aligned semantic parse trees. The joint projectivity signature $P_{every/some}$ operates on the semantic relation $\sqsupset$ determined by the aligned pair *animal/dog* to determine entailment ($\sqsubset$) for the whole. In contrast, if we reverse *every* and *some*, creating the example *some animal/every dog*, then the joint projectivity signature $P_{some/every}$ operates on $\sqsupset$, which determines reverse entailment ($\sqsupset$).

4 Fair Artificial NLI Datasets

Our central empirical question is whether current neural models can learn to do robust natural language inference if given fair datasets. We now present a method for addressing this question. To do this, we need to move beyond the simple propositional logic example explored above, to come closer to the true complexity of natural language. To do this, we adopt a variant of the *natural logic* developed by MacCartney and Manning (2007, 2009) (see also Sánchez-Valencia 1991; van Benthem 2008; Icard and Moss 2013). Natural logic

is a flexible approach to doing logical inference directly on natural language expressions. Thus, in this setting, we can work directly with natural language sentences while retaining complete control over all aspects of the generated dataset.

4.1 Natural Logic

We define natural logic reasoning over aligned semantic parse trees that represent both the premise and hypothesis as a single structure and allow us to calculate semantic relations for all phrases compositionally. The core components are *semantic relations*, which capture the direct inferential relationships between words and phrases, and *projectivity signatures*, which encode how semantic operators interact compositionally with their arguments. We employ the semantic relations of MacCartney and Manning (2009), as in Table 2. We use $\mathcal{B}$ to denote the set containing these seven semantic relations.

The essential concept for the material to come is that of *joint projectivity*: for a pair of semantic functions f and g and a pair of inputs X and Y that are in relation R, the joint projectivity signature $P_{f/g} : \mathcal{B} \to \mathcal{B}$ is a function such that the relation between $f(X)$ and $g(Y)$ is $P_{f/g}(R)$. Figure 2 illustrates this with the phrases *every animal* and *some dog*. We show the details of how the natural logic of MacCartney and Manning (2009), with a small extension, determines the joint projectivity signatures for our datasets in Appendix B.

4.2 A Fragment of Natural Language

Our fragment G consists of sentences of the form:

$$\mathrm{Q_S\ Adj_S\ N_S\ Neg\ Adv\ V\ Q_O\ Adj_O\ N_O}$$

where $\mathrm{N_S}$ and $\mathrm{N_O}$ are nouns, V is a verb, $\mathrm{Adj_S}$ and $\mathrm{Adj_O}$ are adjectives, and Adv is an adverb. Neg is *does not*, and $\mathrm{Q_S}$ and $\mathrm{Q_O}$ can be *every*, *not every*, *some*, or *no*; in each of the remaining categories, there are 100 words. Additionally, $\mathrm{Adj_S}$, $\mathrm{Adj_O}$, Adv, and Neg can be the empty string ε, which is represented in the data by a unique token. Semantic scope is fixed by surface order, with earlier elements scoping over later ones.

For NLI, we define the set of premise–hypothesis pairs $\mathcal{S} \subset G \times G$ such that $(s_p, s_h) \in \mathcal{S}$ iff the non-identical non-empty nouns, adjectives, verbs, and adverbs with identical positions in s_p and s_h are in the $\#$ relation. This constraint on $\mathcal{S}$ trivializes the task of determining the lexical relations between adjectives, nouns, adverbs, and verbs, since the relation is $\equiv$ where the two aligned elements are identical and otherwise $\#$. Furthermore, it follows that distinguishing contradictions from entailments is trivial. The only sources of contradictions are negation and the negative quantifiers *no* and *not every*. Consider $(s_p, s_h) \in \mathcal{S}$ and let C be the number of times negation or a negative quantifier occurs in s_p and s_h. If s_p contradicts s_h, then C is odd; if s_p entails s_h, then C is even.

We constrain the open-domain vocabulary to stress models with learning interactions between logically complex function words; we trivialize the task of lexical semantics to isolate the task of compositional semantics. We also do not have multiple morphological forms, use artificial tokens that do not correspond to English words, and collapse *do not* and *not every* to single tokens to further simplify the task and isolate a model's ability to perform compositional logical reasoning.

Our corpora use the three-way labeling scheme of *entailment*, *contradiction*, and *neutral*. To assign these labels, we translate each premise–hypothesis pair into first-order logic and use Prover9 (McCune, 2005–2010). We assume no expression is empty or universal and encode these assumptions as additional premises. This label generation process implicitly assumes the relation between unequal nouns, verbs, adjectives, and adverbs is independence.

When we generate training data for NLI corpora from some subset $\mathcal{S}_{train} \subset \mathcal{S}$, we perform the following balancing. For a given example, every adjective–noun and adverb–verb pair across the premise and hypothesis is equally likely to have the relation $\equiv$, $\sqsubset$, $\sqsupset$, or $\#$. Without this balancing, any given adjective–noun and adverb–verb pair across the premise and hypothesis has more than a 99% chance of being in the independence relation for values of $\mathcal{S}_{train}$ we consider. Even with this step, 98% of the sentence pairs are neutral, so we again sample to create corpora that are balanced across the three NLI labels. This balancing across our three NLI labels justifies our use of an accuracy metric rather than an F1 score.

4.3 Composition Trees for NLI

We provide a composition tree for inference on $\mathcal{S}$ in Figure 3. This is an *aligned* composition tree, as in Figure 2: it jointly composes lexical items from the premise and hypothesis. The leaf nodes come in sibling pairs where one sibling is a lexical item from the premise and the other is a lexical item

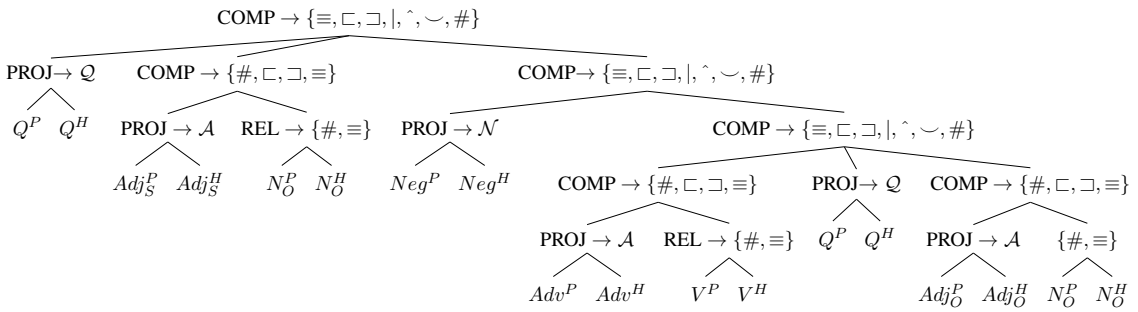

Figure 3: An aligned composition tree for inference on our set of examples $\mathcal{S}$. The superscripts P and H refer to premise and hypothesis. The semantic relations are defined in Table 2. The set Q is {*some*, *every*, *no*, *not*, *every*}. The set *Neg* is $\{\varepsilon, \textit{not}\}$. $\mathcal{Q}$ is the set of 16 joint projectivity signatures between the elements of Q. $\mathcal{N}$ is the set of 4 joint projectivity signatures between ε and *no*. $\mathcal{A}$ is the set of 4 joint projectivity signatures between ε and an intersective adjective or adverb. REL computes the semantic relations between lexical items, PROJ computes the joint projectivity between two semantic functions (Section 4.1 and Appendix B), and COMP applies semantic relations to joint projectivity signatures. This composition tree defines over 10^{26} distinct examples.

from the hypothesis. If both leaf nodes in a sibling pair have domains containing lexical items that are semantic functions, then their parent node domain contains the joint projectivity signatures between those semantic functions. Otherwise the parent node domain contains the semantic relations between the lexical items in the two sibling node domains. The root captures the overall semantic relation between the premise and the hypothesis, while the remaining non-leaf nodes represent intermediate phrasal relations.

The sets Adj_S, N_S, Adj_O, N_O, *Adv*, and *V* each have 100 of their respective open class lexical items with *Adv*, Adj_S, and Adj_O also containing the empty string ε. The set Q is {*some*, *every*, *no*, *not*, *every*} and the set *Neg* is $\{\varepsilon, \textit{not}\}$. $\mathcal{Q}$ is the set of 16 joint projectivity signatures between the quantifiers *some*, *every*, *no*, and *not every*, $\mathcal{N}$ is the set of 4 joint projectivity signatures between the empty string ε and *no*, and $\mathcal{A}$ is the set of 4 projectivity signatures between ε and an intersective adjective or adverb. These joint projectivity signatures were exhaustively determined by us by hand, using the projectivity signatures of negation and quantifiers provided by MacCartney and Manning (2009) as well as a small extension (details in Appendix B).

The function PROJ computes the joint projectivity signature between two semantic functions, REL computes the semantic relation between two lexical items, and COMP inputs semantic relations into a joint projectivity signature and outputs the result. We trimmed the domain of every node so that the function of every node is surjective. Pairs of subexpressions containing quantifiers can be in any of the seven basic semantic relations; even with the contributions of open-class lexical items trivialized, the level of complexity remains high, and all of it emerges from semantic composition, rather than from lexical relations.

4.4 A Difficult But Fair NLI Task

A fair training dataset exposes each local function to all possible inputs. Thus, a fair training dataset for NLI will have the following properties. First, all lexical semantic relations must be included in the training data, else the lexical targets could be underdetermined. Second, for any aligned semantic functions f and g with unknown joint projectivity signature $P_{f/g}$, and for any semantic relation R, there is some training example where $P_{f/g}$ is exposed to the semantic relation R. This ensures that the model has enough information to learn full joint projectivity signatures. Even with these constraints in place, the composition tree of Section 3.1 determines an enormous number of very challenging train/test splits. Appendix C fully defines the procedure for data generation.

We also experimentally verify that our baseline learns a perfect solution from the data we generate. The training set contains 500,000 examples randomly sampled from $\mathcal{S}_{train}$ and the test and development sets each contain 10,000 distinct examples randomly sampled from $\mathcal{S}_{train}$. All random sampling is balanced across adjective–noun and adverb–verb relations as well as across the three NLI labels, as described in Section 4.2.

5 Models

We consider six different model architectures:

CBoW Premise and hypothesis are represented by the average of their respective word embeddings (continuous bag of words).

LSTM Encoder Premise and hypothesis are processed as sequences of words using a recurrent neural network (RNN) with LSTM cells, and the final hidden state of each serves as its representation (Hochreiter and Schmidhuber, 1997; Elman, 1990; Bowman et al., 2015a).

TreeNN Premise and hypothesis are processed as trees, and the semantic composition function is a single-layer feed-forward network (Socher et al., 2011b,a). The value of the root node is the semantic representation in each case.

Attention LSTM An LSTM RNN with word-by-word attention (Rocktäschel et al., 2015).

CompTreeNN Premise and hypothesis are processed as a single aligned tree, following the structure of the composition tree in Figure 3. The semantic composition function is a single-layer feed-forward network (Socher et al., 2011b,a). The value of the root node is the semantic representation of the premise and hypothesis together.

CompTreeNTN Identical to the CompTreeNN, but with a neural tensor network as the composition function (Socher et al., 2013).

For the first three models, the premise and hypothesis representations are concatenated. For the CompTreeNN, CompTreeNTN, and Attention LSTM, there is just a single representation of the pair. In all cases, the premise–hypothesis representation is fed through two hidden layers and a softmax layer.

All models are initialized with random 100-dimensional word vectors and optimized using Adam (Kingma and Ba, 2014). It would not be possible to use pretrained word vectors, due to the artificial nature of our dataset. A grid hyperparameter search was run over dropout values of $\{0, 0.1, 0.2, 0.3\}$ on the output and keep layers of LSTM cells, learning rates of $\{1e{-}2, 3e{-}3, 1e{-}3, 3e{-}4\}$, L2 regularization values of $\{0, 1e{-}4, 1e{-}3, 1e{-}2\}$ on all weights, and activation functions ReLU and $\tanh$. Each hyperparameter setting was run for three epochs and parameters with the highest development set score were used for the complete training runs.

sentence	adverb-verb phrase
every tall kid ϵ happily kicks every ϵ rock *entailment* no tall kid does not ϵ kicks some large rock	happily kicks $\sqsubset$ ϵ kicks
negated verb phrases	adjective-noun phrase
ϵ happily kicks every ϵ rock | does not ϵ kicks some large rock	tall kid $\equiv$ tall kid
verb phrases	single words
happily kicks every ϵ rock $\sqsubset$ ϵ kicks some large rock	tall $\equiv$ tall kid $\equiv$ kid happily $\sqsubset$ ϵ
adjective-noun phrase	single words
ϵ rock $\sqsupset$ large rock	kicks $\equiv$ kicks ϵ $\sqsupset$ large rock $\equiv$ rock

Figure 4: For any example sentence pair (top left) the neural models are trained using the a weighted sum of the error on 12 prediction tasks shown above. The 12 errors are weighted to regularize the loss according to the length of the expressions being predicted on.

The training datasets for this generalization task are only fair if the outputs realized at every non-leaf node are provided during training just as they are in our baseline learning model. For our neural models, we accomplish this by predicting semantic relations for every subexpression pair in the scope of a node in the tree in Figure 3 and summing the loss of the predictions together. We do not do this for the nodes labeled PROJ $\rightarrow \mathcal{Q}$ or PROJ $\rightarrow \mathcal{N}$, as the function PROJ is a bijection at these nodes and no intermediate representations are created. For any example sentence pair the neural models are trained using the a weighted sum of the error on 12 prediction tasks shown in Figure 4. The 12 errors are weighted to regularize the loss according to the length of the expressions being predicted on.

The CompTreeNN and CompTreeNTN models are structured to create intermediate representations of these 11 aligned phrases and so intermediate predictions are implemented as in the sentiment models of Socher et al. (2013). The other models process each of the 11 pairs of aligned phrases separately. Different softmax layers are used depending on the number of classes, but otherwise the networks have identical parameters for all predictions.

6 Results and Analysis

Table 3 summarizes our findings on the hardest of our fair generalization tasks, where the training sets are minimal ones required for fairness.

Model	Train	Dev	Test
CBoW	88.04 ± 0.68	54.18 ± 0.17	53.99 ± 0.27
TreeNN	67.01 ± 12.71	54.01 ± 8.40	53.73 ± 8.36
LSTM encoder	98.43 ± 0.41	53.14 ± 2.45	52.51 ± 2.78
Attention LSTM	73.66 ± 9.97	47.52 ± 0.43	47.28 ± 0.95
CompTreeNN	99.65 ± 0.42	80.17 ± 7.53	80.21 ± 7.71
CompTreeNTN	99.92 ± 0.08	90.45 ± 2.48	90.32 ± 2.71

Table 3: Mean accuracy of 5 runs on our difficult but fair generalization task, with standard 95% confidence intervals. These models are trained on the intermediate predictions described in Section 5.

The four standard neural models fail the task completely. The CompTreeNN and CompTreeNTN, while better, are not able to solve the task perfectly either. However, it should be noted that the CompTreeNN outperforms our four standard neural models by $\approx 30\%$ and the CompTreeNTN improves on this by another $\approx 10\%$. This increase in performance leads us to believe there may be some other composition function that solves this task perfectly.

Both the CompTreeNN and CompTreeNTN have large 95% confidence intervals, indicating that the models are volatile and sensitive to random initialization. The TreeNN also has a large 95% interval. On one of the five runs, the TreeNN achieved a test accuracy of 65.76%, much higher than usual, indicating that this model may have more potential than the other three.

Figure 5, left panel, provides further insights into these results by tracking dev-set performance throughout training. It is evident here that the standard models never get traction on the problem. The volatility of the CompTreeNN and CompTreeNTN is also again evident. Notably, the CompTreeNN is the only model that doesn't peak in the first four training epochs, showing steady improvement throughout training.

We can also increase the number of training examples so that the training data redundantly encodes the information needed for fairness. As we do this, the learning problem becomes one of trivial memorization. Figure 5, right panel, tracks performance on this sequence of progressively more trivial problems. The CompTreeNN and CompTreeNTN both rapidly ascend to perfect performance. In contrast, the four standard models continue to have largely undistinguished performance for all but the most trivial problems. Finally, CBoW, while competitive with other neural models initially, falls behind in a permanent way; its inability to account for word order prevents it from even memorizing the training data.

The results in Figure 5 are for models trained to predict the semantic relations for every subexpression pair in the scope of a node in the tree in Figure 3 (as discussed in Section 5), but we also trained the models without intermediate predictions to quantify their impact.

All models fail on our difficult generalization task when these intermediate values are withheld. Without intermediate values this task is unfair by our standards, so this to be expected. In the hardest generalization setting the CBoW model is the only one of the four standard models to show statistically significant improvement when intermediate predictions are made. We hypothesize that the model is learning relations between open-class lexical items, which are more easily accessible in its sentence representations. As the generalization task approaches a memorization task, the four standard models benefit more and more from intermediate predictions. In the easiest generalization setting, the four standard models are unable to achieve a perfect solution without intermediate predictions, while the CompTreeNN and CompTreeNTN models achieve perfection with or without the intermediate values. Geiger et al. (2018) show that this is due to standard models being unable to learn the lexical relations between open class lexical items when not directly trained on them. Even with intermediate predictions, the standard models are only able to learn the base case of this recursive composition.

7 The Problem is Architecture

One might worry that these results represent a failure of model capacity. However, the systematic errors remain even for much larger networks;

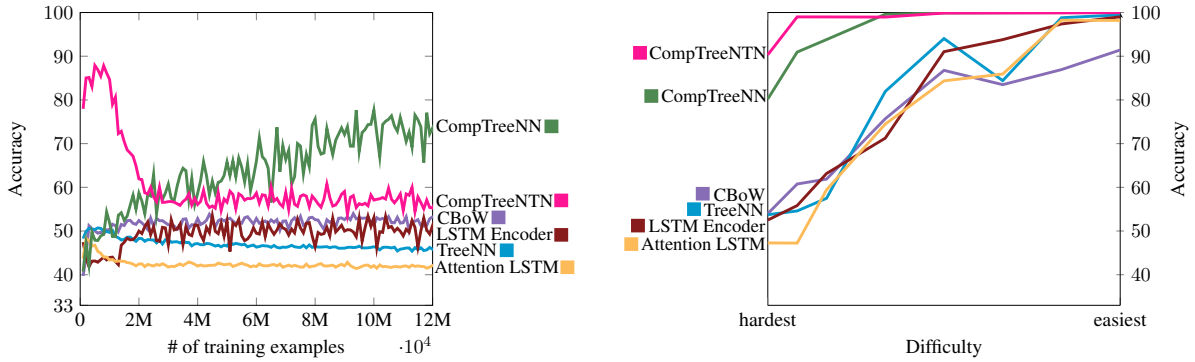

Figure 5: **Left**: Model performance on our difficult but fair generalization task throughout training. **Right**: Mean accuracy of 5 runs as we move from true generalization tasks ('hardest') to problems in which the training set contains so much redundant encoding of the test set that the task is essentially one of memorization ('easiest'). Only the task-specific CompTreeNN and CompTreeNTN are able to do well on true generalization tasks. The other neural models succeed only where memorization suffices, and the CBoW model never succeeds because it does not encode word order.

the trends by epoch and final results are virtually identical with 200-dimensional rather than 100-dimensions representations.

The reason these standard neural models fail to perform natural logic reasoning is their architecture. The CBoW, TreeNN, and LSTM Encoder models all separately bottleneck the premise and hypothesis sentences into two sentence vector embeddings, so the only place interactions between the two sentences can occur is in the two hidden layers before the softmax layer. However, the essence of natural logic reasoning is recursive composition up a tree structure where the premise and hypothesis are composed jointly, so this bottleneck proves extremely problematic. The Attention LSTM model has an architecture that can align and combine lexical items from the premise and hypothesis, but it cannot perform this process recursively and also fails. The CompTreeNN and CompTreeNTN have this recursive tree structure encoded as hard alignments in their architecture, resulting in higher performance. Perhaps in future work, a general purpose model will be developed that can learn to perform this recursive composition without a hard-coded aligned tree structure.

8 Conclusion and Future Work

It is vital that we stress-test our models of semantics using methods that go beyond standard naturalistic corpus evaluations. Recent experiments with artificial and adversarial example generation have yielded valuable insights here already, but it is vital that we ensure that these evaluations are fair in the sense that they provide our models with achievable, unambiguous learning targets. We must carefully and precisely navigate the border between meaningful difficulty and impossibility. To this end, we developed a formal notion of fairness for train/test splits.

This notion of fairness allowed us to rigorously pose the question of whether specific NLI models can learn to do robust natural logic reasoning. For our standard models, the answer is no. For our task-specific models, which align premise and hypothesis, the answer is more nuanced; they do not achieve perfect performance on our task, but they do much better than standard models. This helps us trace the problem to the information bottleneck formed by learning separate premise and hypothesis representations. This bottleneck prevents the meaningful interactions between the premise and hypothesis that are at the core of inferential reasoning with language. Our task-specific models are cumbersome for real-world tasks, but they do suggest that truly robust models of semantics will require much more compositional interaction than is typical in today's standard architectures.

Acknowledgments

We thank Adam Jaffe for help developing mathematical notation and Thomas Icard for valuable discussions. This research is based in part upon work supported by the Stanford Data Science Initiative and by the NSF under Grant No. BCS-1456077. This research is based in part upon work supported by a Stanford Undergraduate Academic Research Major Grant.

References

Johan van Benthem. 2008. A brief history of natural logic. In *Logic, Navya-Nyaya and Applications: Homage to Bimal Matilal*.

Samuel R. Bowman. 2013. Can recursive neural tensor networks learn logical reasoning? *CoRR*, abs/1312.6192.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015a. 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, Lisbon, Portugal. Association for Computational Linguistics.

Samuel R. Bowman, Christopher Potts, and 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*, pages 12–21. Association for Computational Linguistics.

Ishita Dasgupta, Demi Guo, Andreas Stuhlmüller, Samuel J. Gershman, and Noah D. Goodman. 2018. Evaluating compositionality in sentence embeddings. *CoRR*, abs/1802.04302.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science*, 14(2):179–211.

Richard Evans, David Saxton, David Amos, Pushmeet Kohli, and Edward Grefenstette. 2018. Can neural networks understand logical entailment? *CoRR*, abs/1802.08535.

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

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. Association for Computational Linguistics.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *ICLR*.

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. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Thomas F. Icard and Lawrence S. Moss. 2013. Recent progress on monotonicity. *Linguistic Issues in Language Technology*, 9(7):1–31.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, Copenhagen, Denmark. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Brenden M. Lake and Marco Baroni. 2017. Still not systematic after all these years: On the compositional skills of sequence-to-sequence recurrent networks. *CoRR*, abs/1711.00350.

Bill MacCartney and Christopher D. Manning. 2007. Natural logic for textual inference. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, RTE '07, pages 193–200, Stroudsburg, PA, USA. Association for Computational Linguistics.

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

W. McCune. 2005–2010. Prover9 and Mace4. `http://www.cs.unm.edu/~mccune/prover9/`.

Aakanksha Naik, Abhilasha Ravichander, Norman Sadeh, Carolyn Rose, and Graham Neubig. 2018. Stress test evaluation for natural language inference. *arXiv preprint arXiv:1806.00692*.

Yixin Nie, Yicheng Wang, and Mohit Bansal. 2018. Analyzing compositionality-sensitivity of NLI models. *CoRR*, abs/1811.07033.

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. Association for Computational Linguistics.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *CoRR*, abs/1509.06664.

V.M.S. Sánchez-Valencia. 1991. *Studies on Natural Logic and Categorial Grammar*. Universiteit van Amsterdam.

Richard Socher, Cliff Chiung-Yu Lin, Andrew Y. Ng, and Christopher D. Manning. 2011a. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 129–136, USA. Omnipress.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011b. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 151–161, Stroudsburg, PA, USA. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *International Conference on Learning Representations*.

Masatoshi Tsuchiya. 2018. Performance impact caused by hidden bias of training data for recognizing textual entailment. *CoRR*, abs/1804.08117.

Sara Veldhoen and Willem Zuidema. 2018. Can neural networks learn logical reasoning? In *Proceedings of the Conference on Logic and Machine Learning in Natural Language (LaML 2017)*, pages 35–41. Association for Computational Linguistics.

Hitomi Yanaka, Koji Mineshima, Daisuke Bekki, Kentaro Inui, Satoshi Sekine, Lasha Abzianidze, and Johan Bos. 2019. HELP: A dataset for identifying shortcomings of neural models in monotonicity reasoning. ArXiv:1904.12166.