Recursive Neural Networks Can Learn Logical Semantics

Samuel R. Bowman[*†] **Christopher Potts**[*] **Christopher D. Manning**[*†‡]
sbowman@stanford.edu cgpotts@stanford.edu manning@stanford.edu

{[*]Dept. of Linguistics, [†]NLP Group, [‡]Dept. of Computer Science}
Stanford University
Stanford, CA 94305, USA

Abstract

Tree-structured recursive neural networks (TreeRNNs) for sentence meaning have been successful for many applications, but it remains an open question whether the fixed-length representations that they learn can support tasks as demanding as logical deduction. We pursue this question by evaluating whether two such models—plain TreeRNNs and tree-structured neural tensor networks (TreeRNTNs)—can correctly learn to identify logical relationships such as entailment and contradiction using these representations. In our first set of experiments, we generate artificial data from a logical grammar and use it to evaluate the models' ability to learn to handle basic relational reasoning, recursive structures, and quantification. We then evaluate the models on the more natural SICK challenge data. Both models perform competitively on the SICK data and generalize well in all three experiments on simulated data, suggesting that they can learn suitable representations for logical inference in natural language.

1 Introduction

Tree-structured recursive neural network models (TreeRNNs; Goller and Kuchler 1996; Socher et al. 2011b) for sentence meaning have been successful in an array of sophisticated language tasks, including sentiment analysis (Socher et al., 2011b; Irsoy and Cardie, 2014), image description (Socher et al., 2014), and paraphrase detection (Socher et al., 2011a). These results are encouraging for the ability of these models to learn to produce and use strong semantic representations for sentences. However, it remains an open question whether any such fully learned model can achieve the kind of high-fidelity distributed representations proposed in recent algebraic work on vector space modeling (Coecke et al., 2011; Grefenstette, 2013; Hermann et al., 2013; Rocktäschel et al., 2014), and whether any such model can match the performance of grammars based in logical forms in their ability to model core semantic phenomena like quantification, entailment, and contradiction (Warren and Pereira, 1982; Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Liang et al., 2013).

Recent work on the algebraic approach of Coecke et al. (2011) has yielded rich frameworks for computing the meanings of fragments of natural language compositionally from vector or tensor representations, but has not yet yielded effective methods for learning these representations from data in typical machine learning settings. Past experimental work on reasoning with distributed representations have been largely confined to short phrases (Mitchell and Lapata, 2010; Grefenstette et al., 2011; Baroni et al., 2012). However, for robust natural language understanding, it is essential to model these phenomena in their full generality on complex linguistic structures.

This paper describes four machine learning experiments that directly evaluate the abilities of these models to learn representations that support specific semantic behaviors. These tasks follow the format of *natural language inference* (also known as *recognizing textual entailment*; Dagan et al. 2006), in which the goal is to determine the core inferential relationship between two sentences. We introduce a novel NN architecture for natural language inference which independently computes vector representations for each of two sentences using standard TreeRNN or TreeRNTN (Socher et al., 2013) models, and produces a judgment for the pair using only those representations. This allows us to gauge the abilities of these two models to represent all of the necessary semantic

Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality (CVSC), pages 12–21,
Beijing, China, July 26-31, 2015.

information in the sentence vectors.

Much of the theoretical work on natural language inference (and some successful implemented models; MacCartney and Manning 2009; Watanabe et al. 2012) involves *natural logics*, which are formal systems that define rules of inference between natural language words, phrases, and sentences without the need of intermediate representations in an artificial logical language. In our first three experiments, we test our models' ability to learn the foundations of natural language inference by training them to reproduce the behavior of the natural logic of MacCartney and Manning (2009) on artificial data. This logic defines seven mutually-exclusive relations of synonymy, entailment, contradiction, and mutual consistency, as summarized in Table 1, and it provides rules of semantic combination for projecting these relations from the lexicon up to complex phrases. The formal properties of this system are now well-understood (Icard and Moss, 2013a; Icard and Moss, 2013b). The first experiment using this logic covers reasoning with the bare logical relations (§3), the second extends this to reasoning with statements constructed compositionally from recursive functions (§4), and the third covers the additional complexity that results from quantification (§5). Though the performance of the plain TreeRNN model is somewhat poor in our first experiment, we find that the stronger TreeRNTN model generalizes well in every case, suggesting that it has learned to simulate our target logical concepts.

The experiments with simulated data provide a convincing demonstration of the ability of neural networks to learn to build and use semantic representations for complex natural language sentences from reasonably-sized training sets. However, we are also interested in the more practical question of whether they can learn these representations from naturalistic text. To address this question, we apply our models to the SICK entailment challenge data in §6. The small size of this corpus puts data-hungry NN models like ours at a disadvantage, but we are nonetheless able to achieve competitive performance on it, surpassing several submitted models with significant hand-engineered task-specific features and our own NN baseline. This suggests that the representational abilities that we observe in the previous sections are not limited to carefully circumscribed tasks. We conclude that

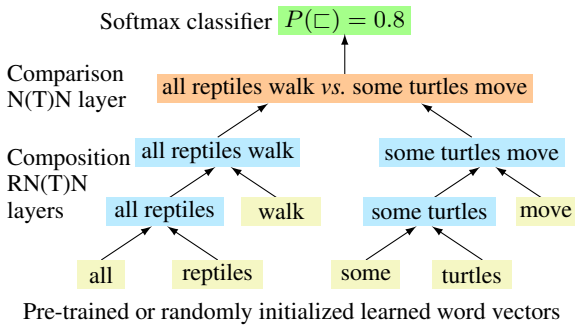


Figure 1: In our model, two separate tree-structured networks build up vector representations for each of two sentences using either NN or NTN layer functions. A comparison layer then uses the resulting vectors to produce features for a classifier.

TreeRNTN models are adequate for typical cases of natural language inference, and that there is not yet any clear level of inferential complexity for which other approaches work and NN models fail.

2 Tree-structured neural networks

We limit the scope of our experiments in this paper to neural network models that adhere to the linguistic *principle of compositionality*, which says that the meanings for complex expressions are derived from the meanings of their parts via specific composition functions (Partee, 1984; Janssen, 1997). In our distributed setting, word meanings are embedding vectors of dimension N. A learned composition function maps pairs of them to single phrase vectors of dimension N, which can then be merged again to represent more complex phrases, forming a tree structure. Once the entire sentence-level representation has been derived at the top of the tree, it serves as a fixed-dimensional input for some subsequent layer function.

To apply these recursive models to our task, we propose the tree pair model architecture depicted in Fig. 1. In it, the two phrases being compared are processed separately using a pair of tree-structured networks that share a single set of parameters. The resulting vectors are fed into a separate comparison layer that is meant to generate a feature vector capturing the relation between the two phrases. The output of this layer is then given to a softmax classifier, which produces a distribution over the seven relations represented in Table 1.

For the sentence embedding portions of the network, we evaluate both TreeRNN models with the

Name	Symbol	Set-theoretic definition	Example
(strict) entailment	$x \sqsubset y$	$x \subset y$	*turtle, reptile*
(strict) reverse entailment	$x \sqsupset y$	$x \supset y$	*reptile, turtle*
equivalence	$x \equiv y$	$x = y$	*couch, sofa*
alternation	$x \mid y$	$x \cap y = \emptyset \wedge x \cup y \neq \mathcal{D}$	*turtle, warthog*
negation	$x \wedge y$	$x \cap y = \emptyset \wedge x \cup y = \mathcal{D}$	*able, unable*
cover	$x \smile y$	$x \cap y \neq \emptyset \wedge x \cup y = \mathcal{D}$	*animal, non-turtle*
independence	$x \# y$	(else)	*turtle, pet*

Table 1: The seven relations of MacCartney and Manning (2009)'s logic are defined abstractly on pairs of sets drawing from the universe $\mathcal{D}$, but can be straightforwardly applied to any pair of natural language words, phrases, or sentences. The relations are defined so as to be mutually exclusive.

standard NN layer function (1) and those with the more powerful neural tensor network layer function (2) proposed in Chen et al. (2013). The nonlinearity $f(x) = \tanh(x)$ is applied elementwise to the output of either layer function.

$$(1) \quad \vec{y}_{\textit{TreeRNN}} = f(\mathbf{M} \begin{bmatrix} \vec{x}^{(l)} \\ \vec{x}^{(r)} \end{bmatrix} + \vec{b})$$

$$(2) \quad \vec{y}_{\textit{TreeRNTN}} = \vec{y}_{\textit{TreeRNN}} + f(\vec{x}^{(l)T} \mathbf{T}^{[1\ldots n]} \vec{x}^{(r)})$$

Here, $\vec{x}^{(l)}$ and $\vec{x}^{(r)}$ are the column vector representations for the left and right children of the node, and $\vec{y}$ is the node's output. The TreeRNN concatenates them, multiplies them by an $N \times 2N$ matrix of learned weights, and adds a bias $\vec{b}$. The TreeRNTN adds a learned full rank third-order tensor $\mathbf{T}$, of dimension $N \times N \times N$, modeling multiplicative interactions between the child vectors. The comparison layer uses the same layer function as the composition layers (either an NN layer or an NTN layer) with independently learned parameters and a separate nonlinearity function. Rather than use a $\tanh$ nonlinearity here, we found better results with the leaky rectified linear function (Maas et al., 2013): $f(x) = \max(x, 0) + 0.01 \min(x, 0)$.

Other strong tree-structured models have been proposed in past work (Socher et al., 2014; Irsoy and Cardie, 2014; Tai et al., 2015), but we believe that these two provide a valuable case study, and that positive results on here are likely to generalize well to stronger models.

To run the model forward, we assemble the two tree-structured networks so as to match the structures provided for each phrase, which are either included in the source data or given by a parser. The word vectors are then looked up from the vocabulary embedding matrix V (one of the learned model parameters), and the composition and comparison functions are used to pass information up the tree and into the classifier. For an objective function, we use the negative log likelihood of the correct label with tuned L2 regularization.

We initialize parameters uniformly, using the range $(-0.05, 0.05)$ for layer parameters and $(-0.01, 0.01)$ for embeddings, and train the model using stochastic gradient descent (SGD) with learning rates computed using AdaDelta (Zeiler, 2012). The classifier feature vector is fixed at 75 dimensions and the dimensions of the recursive layers are tuned manually. Training times on CPUs vary from hours to days across experiments. On the experiments which use artificial data, we report mean results over five fold cross-validation, where variance across runs is typically no more than two percentage points. In addition, because the classes are not necessarily balanced, we report both accuracy and macroaveraged F1.[1] Source code and generated data can be downloaded from `http://stanford.edu/~sbowman/`.

3 Reasoning about semantic relations

The simplest kinds of deduction in natural logic involve atomic statements using the relations in Table 1. For instance, from the relation $p_1 \sqsupset p_2$ between two propositions, one can infer the relation $p_2 \sqsubset p_1$ by applying the definitions of the relations directly. If one is also given the relation $p_2 \sqsupset p_3$ one can conclude that $p_1 \sqsupset p_3$, by basic set-theoretic reasoning (transitivity of $\sqsupset$). The full set of sound such inferences on pairs of premise relations is depicted in Table 2. Though these basic inferences do not involve compositional sentence representations, any successful reasoning using compositional representations will rely on the ability to perform sound inferences of this kind

[1] We compute macroaveraged F1 as the harmonic mean of average precision and average recall, both computed for all classes for which there is test data, setting precision to 0 where it is not defined.

	$\equiv$	$\sqsubset$	$\sqsupset$	$^\wedge$	$\mid$	$\smile$	$\#$
$\equiv$	$\equiv$	$\sqsubset$	$\sqsupset$	$^\wedge$	$\mid$	$\smile$	$\#$
$\sqsubset$	$\sqsubset$	$\sqsubset$	·	$\mid$	$\mid$	·	·
$\sqsupset$	$\sqsupset$	·	$\sqsupset$	$\smile$	·	$\smile$	·
$^\wedge$	$^\wedge$	$\smile$	$\mid$	$\equiv$	$\sqsupset$	$\sqsubset$	$\#$
$\mid$	$\mid$	·	$\mid$	$\sqsubset$	·	$\sqsubset$	·
$\smile$	$\smile$	$\smile$	·	$\sqsupset$	$\sqsupset$	·	·
$\#$	$\#$	·	·	$\#$	·	·	·

Table 2: In §3, we assess our models' ability to learn to do inference over pairs of relations using the rules represented here, which are derived from the definitions of the relations in Table 1. As an example, given that $p_1 \sqsubset p_2$ and $p_2 \,^\wedge p_3$, the entry in the $\sqsubset$ row and the $^\wedge$ column lets us conclude that $p_1 \mid p_3$. Cells containing a dot correspond to situations for which no valid inference can be drawn.

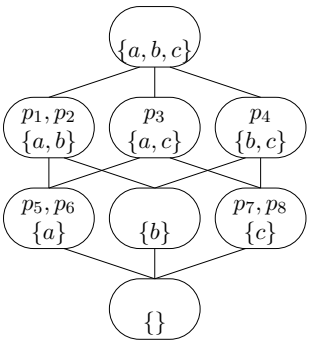
	Train	Test
# only	53.8 (10.5)	53.8 (10.5)
15d NN	99.8 (99.0)	94.0 (87.0)
15d NTN	**100 (100)**	**99.6 (95.5)**

Table 3: Performance on the semantic relation experiments. These results and all other results on artificial data are reported as mean accuracy scores over five runs followed by mean macroaveraged F1 scores in parentheses. The "# only" entries reflect the frequency of the most frequent class.

in order to be able to use unseen relational facts within larger derivations. Our first experiment studies how well each model can learn to perform them them in isolation.

Experiments We begin by creating a world model on which we will base the statements in the train and test sets. This takes the form of a small Boolean structure in which terms denote sets of entities from a small domain. Fig. 2a depicts a structure of this form with three entities (a, b, and c) and eight proposition terms (p_1–p_8). We then generate a relational statement for each pair of terms in the model, as shown in Fig. 2b. We divide these statements evenly into train and test sets, and delete the test set examples which cannot be proven from the train examples, for which there is not enough information for even an ideal system to choose a correct label. In each experimental run, we create a model with 80 terms over a domain of 7 elements, yielding a training set of 3200 examples and a test set of 2960 examples.

We trained models with both the NN and NTN



(a) Example boolean structure, shown with edges idicating inclusion. The terms p_1–p_8 name the sets. Not all sets have names, and some sets have multiple names, so that learning $\equiv$ is non-trivial.

Train	Test
$p_1 \equiv p_2$	$p_2 \,^\wedge p_7$
$p_1 \sqsupset p_5$	$p_2 \sqsupset p_5$
$p_4 \sqsupset p_8$	~~$p_5 \equiv p_6$~~
$p_5 \mid p_7$	~~$p_7 \sqsubset p_4$~~
$p_7 \,^\wedge p_1$	$p_8 \sqsubset p_4$

(b) A few examples of atomic statements about the model depicted above. Test statements that are not provable from the training data shown are crossed out.

Figure 2: Small example structure and data for learning relation composition.

comparison functions on these data sets.[2] In both cases, the models are implemented as described in §2, but since the items being compared are single terms rather than full tree structures, the composition layer is not used, and the two models are not recursive. We simply present the models with the (randomly initialized) embedding vectors for each of two terms, ensuring that the model has no information about the terms being compared except for the relations between them that appear in training.

Results The results (Table 3) show that NTN is able to accurately encode the relations between the terms in the geometric relations between their vectors, and is able to then use that information to recover relations that are not overtly included in the training data. The NN also generalizes fairly well, but makes enough errors that it remains an open question whether it is capable of learning representations with these properties. It is not possible for us to rule out the possibility that different optimization techniques or finer-grained hyperparameter tuning could lead an NN model to succeed.

As an example from our test data, both mod-

[2]Since this task relies crucially on the learning of a pair of vectors, no simpler version of our model is a viable baseline.

els correctly labeled $p_1 \sqsubset p_3$, potentially learning from the training examples $\{p_1 \sqsubset p_{51},\ p_3 \sqsupset p_{51}\}$ or $\{p_1 \sqsubset p_{65},\ p_3 \sqsupset p_{65}\}$. On another example involving comparably frequent relations, the NTN correctly labeled $p_6 \sqsupset p_{24}$, likely on the basis of the training examples $\{p_6 \smile p_{28},\ p_{28} \frown p_{24}\}$, while the NN incorrectly assigned it #.

4 Recursive structure

A successful natural language inference system must reason about relations not just over familiar atomic symbols, but also over novel structures built up recursively from these symbols. This section shows that our models can learn a compositional semantics over such structures. In our evaluations, we exploit the fact that our logical language is infinite by testing on strings that are longer and more complex than any seen in training.

Experiments As in §3, we generate artificial data from a formal system, but we now replace the unanalyzed symbols from that experiment with complex formulae. These formulae represent a complete classical propositional logic: each atomic symbol is a variable over the domain {T, F}, and the only operators are truth-functional ones. Table 4a defines this logic, and Table 4b gives some short examples of relational statements from our data. To compute these relations between statements, we exhaustively enumerate the sets of assignments of truth values to propositional variables that would satisfy each of the statements, and then we convert the set-theoretic relation between those assignments into one of the seven relations in Table 1. As a result, each relational statement represents a valid theorem of the propositional logic, and to succeed, the models must learn to reproduce the behavior of a theorem prover.[3]

In our experiments, we randomly generate unique pairs of formulae containing up to 12 instances of logical operators each and compute the relation that holds for each pair. We discard pairs in which either statement is either a tautology or a contradiction, for which the seven relations in Table 1 are undefined. The resulting set of formula pairs is then partitioned into 12 bins according the number of operators in the larger of the two formulae. We then sample 20% of each bin for a held-out test set. If we do not implement any constraint that the two statements being compared are similar in any way, then the generated data are dominated by statements in which the two formulae refer to largely separate subsets of the six variables, which means that the # relation is almost always correct. In an effort to balance the distribution of relation labels without departing from the basic task of modeling propositional logic, we disallow individual pairs of statements from referring to more than four of the six propositional variables.

In order to test the model's generalization to unseen structures, we discard training examples with more than 4 logical operators, yielding 60k short training examples, and 21k test examples across all 12 bins. In addition to the two tree models, we also train a summing NN baseline which is largely identical to the TreeRNN, except that instead of using a learned composition function, it simply sums the term vectors in each expression to compose them before passing them to the comparison layer. Unlike the two tree models, this baseline does not use word order, and is as such guaranteed to ignore some information that it would need in order to succeed perfectly.

Results Fig. 3 shows the relationship between test accuracy and statement size. While the summing baseline model performed poorly across the

Formula	Interpretation
$p_1, p_2, p_3, p_4, p_5, p_6$	$[\![x]\!] \in \{\mathtt{T}, \mathtt{F}\}$
not φ	T iff $[\![\varphi]\!] = \mathtt{F}$
$(\varphi$ *and* $\psi)$	T iff $\mathtt{F} \notin \{[\![\varphi]\!], [\![\psi]\!]\}$
$(\varphi$ *or* $\psi)$	T iff $\mathtt{T} \in \{[\![\varphi]\!], [\![\psi]\!]\}$

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

not p_3	$\frown$	p_3
not not p_6	$\equiv$	p_6
p_3	$\sqsubset$	$(p_3$ *or* $p_2)$
$(p_1$ *or* $(p_2$ *or* $p_4))$	$\sqsupset$	$(p_2$ *and not* $p_4)$
not $(p_1$ *and not* $p_2)$	$\equiv$	$(p_1$ *or* $p_2)$

(b) Short examples of the type of statements used for training and testing. These are relations between well-formed formulae, computed in terms of sets of satisfying interpretation functions $[\![\cdot]\!]$.

Table 4: Natural logic relations over sentences of propositional logic.

[3] Socher et al. (2012) show that a matrix-vector TreeRNN model somewhat similar to our TreeRNTN can learn boolean logic, a logic where the atomic symbols are simply the values T and F. While learning the operators of that logic is not trivial, the outputs of each operator can be represented accurately by a single bit. In the much more demanding task presented here, the atomic symbols are variables over these values, and the sentence vectors must thus be able to distinguish up to 2^{2^6} distinct conditions on valuations.

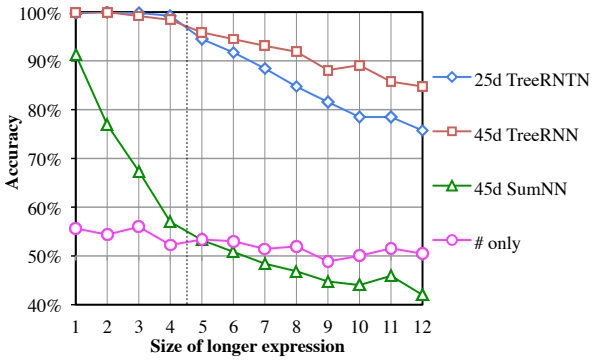


Figure 3: Results on recursive structure. The vertical dotted line marks the size of the longest training examples.

board, we found that both recursive models were able to perform well on unseen small test examples, with TreeRNN accuracy above 98% and TreeRNTN accuracy above 99% on formulae below length five, indicating that they learned correct approximations of the underlying logic. Training accuracy was 66.6% for the SumNN, 99.4% for the TreeRNN, and 99.8% for the TreeRNTN.

After the size four training cutoff, performance gradually decays with expression size for both tree models, suggesting that the learned approximations were accurate but lossy. Despite the TreeRNTN's stronger performance on short sentences, its performance decayed more quickly than the TreeRNN's. This suggests to us that it learned to interpret many specific fixed-size tree structures directly, allowing it to get away without learning as robust generalizations about how to compose terms in the general case. Two factors may have contributed to the learning of these narrower generalizations: even with the lower dimension, the TreeRNTN composition function has about eight times as many parameters as the TreeRNN, and the TreeRNTN worked best with weaker L2 regularization than the TreeRNN ($\lambda = 0.0003$ vs. 0.001). However, even in the most complex set of test examples, the TreeRNTN classifies true examples of every class but $\equiv$ (which is rare in long examples, and occurs only once here) correctly the majority of the time, and the performance of both models on those examples indicates that both have learned reasonable approximations of the underlying theorem proving task over recursive structure.

5 Reasoning with quantifiers and negation

We have seen that recursive models can learn an approximation of propositional logic. However, natural languages can express functional meanings of considerably greater complexity than this. As a key test of whether our models can capture this complexity, we now study the degree to which they are able to develop suitable representations for the semantics of natural language quantifiers like *most* and *all* as they interact with negation and lexical entailments. Quantification and negation are far from the only place in natural language where complex functional meanings are found, but they are natural focus, since they have formed a standard case study in prior formal work on natural language inference (Icard and Moss, 2013b).

Experiments Our data consist of pairs of sentences generated from a grammar for a simple English-like artificial language. Each sentence contains a quantifier, a noun which may be negated, and an intransitive verb which may be negated. We use the quantifiers *some*, *most*, *all*, *two*, and *three*, and their negations *no*, *not-all*, *not-most*, *less-than-two*, and *less-than-three*, and also include five nouns, four intransitive verbs, and the negation symbol *not*. In order to be able to define relations between sentences with differing lexical items, we define the lexical relations for each noun–noun pair, each verb–verb pair, and each quantifier–quantifier pair. The grammar then generates pairs of sentences and calculates the relations between them. For instance, our models might then see pairs like (3) and (4) in training and be required to then label (5).

(3) (most turtle) swim | (no turtle) move

(4) (all lizard) reptile $\sqsubset$ (some lizard) animal

(5) (most turtle) reptile | (all turtle) (not animal)

In each run, we randomly partition the set of valid *single sentences* into train and test, and then label all of the pairs from within each set to generate a training set of 27k pairs and a test set of 7k pairs. Because the model doesn't see the test sentences at training time, it cannot directly use the kind of reasoning described in §3 at the sentence level (by treating sentences as unanalyzed symbols), and must instead jointly learn the word-level relations and a complete reasoning system over them for our logic.

	Train	Test
# only	35.4 (7.5)	35.4 (7.5)
25d SumNN	96.9 (97.7)	93.9 (95.0)
25d TreeRNN	99.6 (99.6)	99.2 (99.3)
25d TreeRNTN	**100** (**100**)	**99.7** (**99.5**)

Table 5: Performance on the quantifier experiments, given as % correct and macroaveraged F1.

We use the same summing baseline as in §4. The highly consistent sentence structure in this experiment means that this model is not as disadvantaged by the lack of word order information as it is in the previous experiment, but the variable placement of *not* nonetheless introduces potential uncertainty in the 58.8% of examples that contain a sentence with a single token of it.

Results The results (Table 5) show that both tree models are able to learn to generalize the underlying logic almost perfectly. The baseline summing model can largely memorize the training data, but does not generalize as well. We do not find any consistent pattern in the handful of errors made by either tree model, and no errors were consistent across model restarts, suggesting that there is no fundamental obstacle to learning a perfect model for this problem.

6 The SICK textual entailment challenge

The specific model architecture that we use is novel, and though the underlying tree structure approach has been validated elsewhere, our experiments so far do not guarantee that it viable model for handling inference over real natural language data. To investigate our models' ability to handle the noisy labels and the diverse range of linguistic structures seen in typical natural language data, we use the SICK textual entailment challenge corpus (Marelli et al., 2014b). The corpus consists of about 10k natural language sentence pairs, labeled with *entailment*, *contradiction*, or *neutral*. At only a few thousand distinct sentences (many of them variants on an even smaller set of template sentences), the corpus is not large enough to train a high quality learned model of general natural language, but it is the largest human-labeled entailment corpus that we are aware of, and our results nonetheless show that tree-structured NN models can learn to approximate natural logic-style inference in the real world.

Adapting to this task requires us to make a few additions to the techniques discussed in §2. In order to better handle rare words, we initialized our word embeddings using 200 dimensional vectors trained with GloVe (Pennington et al., 2014) on data from Wikipedia. Since 200 dimensional vectors are too large to be practical in an TreeRNTN on a small dataset, a new embedding transformation layer is needed. Before any embedding is used as an input to a recursive layer, it is passed through an additional $\tanh$ neural network layer with the same output dimension as the recursive layer. This new layer allows the model to choose which aspects of the 200 dimensional representations from the unsupervised source it most values, rather than relying on GloVe—which is has no knowledge of the task—to do so, as would be the case were GloVe asked to directly produce vectors of the lower dimensionality. An identical layer is added to the SumNN between the word vectors and the comparison layer.

We also supplemented the SICK training data[4] (4500 examples) with 600k examples of approximate entailment data from the Denotation Graph project (DG, Hodosh et al. 2014, also used by the winning SICK submission), a corpus of noisy automatically labeled entailment examples over image captions, the same genre of text from which SICK was drawn. We trained a single model on data from both sources, but used a separate set of softmax parameters for classifying into the labels from each source, and forced the model to sample SICK examples and DG examples about equally often during training.

We parsed the data from both sources with the Stanford PCFG Parser v. 3.3.1 (Klein and Manning, 2003). We also found that we were able to train a working model much more quickly with an additional technique: we collapse subtrees that were identical across both sentences in a pair by replacing them with a single head word. The training and test data on which we report performance are collapsed in this way, and both collapsed and uncollapsed copies of the training data are used in training. Finally, in order to improve regularization on the noisier data, we used dropout (Srivastava et al., 2014) at the input to the comparison layer (10%) and at the output from the embedding

[4] We tuned the model using performance on a held out development set, but report performance here for a version of the model trained on both the training and development data and tested on the 4,928 example SICK test set. We also report training accuracy on a small sample from each data source.

The patient is being helped by the doctor	*entailment*	The doctor is helping the patient (PASSIVE)
A little girl is playing the violin on a beach	*contradiction*	There is no girl playing the violin on a beach (NEG)
The yellow dog is drinking water from a bottle	*contradiction*	The yellow dog is drinking water from a pot (SUBST)
A woman is breaking two eggs in a bowl	*neutral*	A man is mixing a few ingredients in a bowl (MULTIED)
Dough is being spread by a man	*neutral*	A woman is slicing meat with a knife (DIFF)

Table 6: Examples of each category used in error analysis from the SICK test data.

	neutral only	30d SumNN	30d TrRNN	50d TrRNTN
DG Train	50.0	68.0	67.0	**74.0**
SICK Train	56.7	96.6	95.4	**97.8**
SICK Test	56.7	73.4	74.9	**76.9**
PASSIVE (4%)	0	76	68	**88**
NEG (7%)	0	96	**100**	**100**
SUBST (24%)	28	**72**	64	**72**
MULTIED (39%)	**68**	61	66	64
DIFF (26%)	**96**	68	79	**96**
SHORT (47%)	50.0	73.9	73.5	**77.3**

Table 7: Classification accuracy, including a category breakdown for SICK test data. Categories are shown with their frequencies.

transform layer (25%).

Results Despite the small amount of high quality training data available and the lack of resources for learning lexical relationships, the results (Table 7) show that our tree-structured models perform competitively on textual entailment, beating a strong baseline. Neither model reached the performance of the winning system (84.6%), but the TreeRNTN did exceed that of eight out of 18 submitted systems, including several which used sophisticated hand-engineered features and lexical resources specific to the version of the entailment task at hand.

To better understand our results, we manually annotated a fraction of the SICK test set, using mutually exclusive categories for passive/active alternation pairs (PASSIVE), pairs differing only by the presence of negation (NEG), pairs differing by a single word or phrase substitution (SUBST), pairs differing by multiple edits (MULTIED), and pairs with little or no content word overlap (DIFF). Examples of each are in Table 6. We annotated 100 random examples to judge the frequency of each category, and continued selectively annotating until each category contained at least 25. We also use the category SHORT for pairs in which neither sentence contains more than ten words.

The results (Table 7) show that the TreeRNTN performs especially strongly in the two categories which pick out specific syntactic configurations, PASSIVE and NEG, suggesting that that model has learned to encode the relevant structures well. It also performs fairly on SUBST, which most closely parallels the lexical entailment inferences addressed in §5. In addition, none of the models perform dramatically better on the SHORT pairs than on the rest of the data, suggesting that the performance decay observed in §4 may not impact models trained on typical natural language text.

It is known that a model can perform well on SICK (like other natural language inference corpora) without taking advantage of compositional syntactic or semantic structure (Marelli et al., 2014a), and our summing baseline model is powerful enough to do this. Our tree models nonetheless perform substantially better, and we remain confident that given sufficient data, it should be possible for the tree models, and not the summing model, to learn a truly high-quality solution.

7 Discussion and conclusion

This paper first evaluates two recursive models on three natural language inference tasks over clean artificial data, covering the core relational algebra of natural logic with entailment and exclusion, recursive structure, and quantification. We then show that the same models can learn to perform an entailment task on natural language. The results suggest that TreeRNTNs, and potentially also TreeRNNs, can learn to faithfully reproduce logical inference behaviors from reasonably-sized training sets. These positive results are promising for the future of learned representation models in the applied modeling of compositional semantics.

Some questions about the abilities of these models remain open. Even the TreeRNTN falls short of perfection in the recursion experiment, with performance falling off steadily as the size of the expressions grows. It remains to be seen whether these deficiencies are limiting in practice, and whether they can be overcome with stronger models or learning techniques. In addition, interesting analytical questions remain about *how* these mod-

els encode the underlying logics. Neither the underlying logical theories, nor any straightforward parameter inspection technique provides much insight on this point, but we hope that further experiments may reveal structure in the learned parameters or the representations they produce.

Our SICK experiments similarly only begin to reveal the potential of these models to learn to perform complex semantic inferences from corpora, and there is ample room to develop our understanding using new and larger sources of natural language data. Nonetheless, the rapid progress the field has made with these models in recent years provides ample reason to be optimistic that learned representation models can be trained to meet all the challenges of natural language semantics.

Acknowledgments

We thank Jeffrey Pennington and Richard Socher, as well as Neha Nayak for developing the SICK collapsing technique.

We also gratefully acknowledge support from a Google Faculty Research Award, a gift from Bloomberg L.P., the Defense Advanced Research Projects Agency (DARPA) Deep Exploration and Filtering of Text (DEFT) Program under Air Force Research Laboratory (AFRL) contract no. FA8750-13-2-0040, the National Science Foundation under grant no. IIS 1159679, and the Department of the Navy, Office of Naval Research, under grant no. N00014-10-1-0109. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of Google, Bloomberg L.P., DARPA, AFRL NSF, ONR, or the US government.

References

M. Baroni, R. Bernardi, N.Q. Do, and C.C. Shan. 2012. Entailment above the word level in distributional semantics. In *Proc. EACL*.

D. Chen, R. Socher, C.D. Manning, and A.Y. Ng. 2013. Learning new facts from knowledge bases with neural tensor networks and semantic word vectors. In *Proc. ICLR*.

B. Coecke, M. Sadrzadeh, and S. Clark. 2011. Mathematical foundations for a compositional distributed model of meaning. *Linguistic Analysis*, 36(1–4).

I. Dagan, O. Glickman, and B. Magnini. 2006. The PASCAL Recognising Textual Entailment Challenge. In *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*. Springer.

C. Goller and A. Kuchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *Proc. IEEE International Conference on Neural Networks*.

E. Grefenstette, M. Sadrzadeh, S. Clark, B. Coecke, and S. Pulman. 2011. Concrete sentence spaces for compositional distributional models of meaning. In *Proc. IWCS*.

E. Grefenstette. 2013. Towards a formal distributional semantics: Simulating logical calculi with tensors. In *Proc. *SEM*.

K.M. Hermann, E. Grefenstette, and P. Blunsom. 2013. "Not not bad" is not "bad": A distributional account of negation. In *Proc. of the 2013 Workshop on Continuous Vector Space Models and their Compositionality*.

M. Hodosh, P. Young, A. Lai, and J. Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *TACL*.

T.F. Icard and L.S. Moss. 2013a. A complete calculus of monotone and antitone higher-order functions. In N. Galatos, A. Kurz, and C. Tsinakis, editors, *Proc. Topology, Algebra, and Categories in Logic*.

T.F. Icard and L.S. Moss. 2013b. Recent progress on monotonicity. *LILT*, 9(7).

O. Irsoy and C. Cardie. 2014. Deep recursive neural networks for compositionality in language. In *Proc. NIPS*.

T.M.V. Janssen. 1997. Compositionality. In J. van Benthem and A. ter Meulen, editors, *Handbook of Logic and Language*. MIT Press and North-Holland.

D. Klein and C.D. Manning. 2003. Accurate unlexicalized parsing. In *Proc. ACL*.

P. Liang, M.I. Jordan, and D. Klein. 2013. Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2).

A.L. Maas, A.Y. Hannun, and A.Y. Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*.

B. MacCartney and C.D. Manning. 2009. An extended model of natural logic. In *Proc. IWCS*.

M. Marelli, L. Bentivogli, M. Baroni, R. Bernardi, S. Menini, and R. Zamparelli. 2014a. SemEval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. *SemEval-2014*.

M. Marelli, S. Menini, M. Baroni, L. Bentivogli, R. Bernardi, and R. Zamparelli. 2014b. A SICK cure for the evaluation of compositional distributional semantic models. In *Proc. LREC*.

J. Mitchell and M. Lapata. 2010. Composition in distributional models of semantics. *Cognitive Science*, 34(8).

B.H. Partee. 1984. Compositionality. In Fred Landman and Frank Veltman, editors, *Varieties of Formal Semantics*. Foris.

J. Pennington, R. Socher, and C.D. Manning. 2014. GloVe: Global vectors for word representation. In *Proc. EMNLP*.

T. Rocktäschel, M. Bosnjak, S. Singh, and S. Riedel. 2014. Low-dimensional embeddings of logic. In *Proc. the ACL 2014 Workshop on Semantic Parsing*.

R. Socher, E.H. Huang, J. Pennington, C.D. Manning, and A.Y. Ng. 2011a. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Proc. NIPS*.

R. Socher, J. Pennington, E.H. Huang, A.Y. Ng, and C.D. Manning. 2011b. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proc. EMNLP*.

R. Socher, B. Huval, C.D. Manning, and A.Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proc. EMNLP*.

R. Socher, A. Perelygin, J. Wu, J. Chuang, C.D. Manning, A.Y. Ng, and C. Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. EMNLP*.

R. Socher, A. Karpathy, Q.V. Le, C.D. Manning, and A.Y. Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *TACL*.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 15(1).

K.S. Tai, R. Socher, and C.D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proc. ACL*.

D.H.D. Warren and F.C.N. Pereira. 1982. An efficient easily adaptable system for interpreting natural language queries. *American Journal of Computational Linguistics*.

Y. Watanabe, J. Mizuno, E. Nichols, N. Okazaki, and K. Inui. 2012. A latent discriminative model for compositional entailment relation recognition using natural logic. In *Proc. COLING*.

M.D. Zeiler. 2012. ADADELTA: An adaptive learning rate method. arXiv:1212.5701.

J.M. Zelle and R.J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proc. AAAI*.

L.S. Zettlemoyer and M. Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proc. of the 21st Conference on Uncertainty in Artificial Intelligence*.