Recursive Neural Networks for Learning Logical Semantics

Anonymous Author(s)
Affiliation
Address
email

Abstract

Supervised recursive neural network models (RNNs) for sentence meaning have been successful in a wide array of sophisticated language tasks, but it remains an open question whether succeed in learning compositional semantic grammars that allow logical deduction. We address this question directly by using a logical grammar to generate controlled data sets encoding the relationships (e.g., entailment, contradiction) between pairs of expressions and evaluating whether each of two classes of neural model — plain RNNs and recursive neural tensor networks (RNTNs) — can learn those relationships correctly. Our first experiment evaluates whether these models can learn the basic algebra of logical relations involved. Our second and third experiments extend this evaluation to complex recursive structures and sentences involving quantification. We find that the plain RNN achieves only mixed results on all three experiments, whereas the stronger RNTN models generalizes well in every setting. This serves as compelling evidence that RNTNs provide a satisfactory representation of meaning for natural language reasoning.

1 Introduction

Supervised recursive neural network models (RNNs) for sentence meaning have been successful in a wide array of sophisticated language tasks, including sentiment analysis [1, 2], image description [3], and paraphrase detection [4]. These results are encouraging for the ability of these models to learn compositional semantic grammars, but it remains an open question whether they can achieve the same results as grammars based in logical forms [5, 6, 7, 8] when it comes to core semantic concepts like quantification, entailment, and contradiction. To date, experimental investigations of these concepts using distributed representations have been largely confined to short phrases [9, 10, 11, 12]. For robust natural language understanding, we must be able to model these phenomena in their full generality on complex linguistic structures.

We address this question in the context of *natural language inference* (also known as *recognizing textual entailment*; [13]), in which the goal is to determine the core inferential relationship between two sentences. Much of the theoretical work on this task (and some successful implemented models [14, 15]) 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. Following [16], we use the natural logic developed by [17] as our formal model. This logic defines seven core 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 and inferential strength of this system are now well-understood [18, 19].

In our experiments, we use this pre-specified logical grammar to generate controlled data sets encoding the semantic relationships between pairs of expressions and evaluate whether each of two

classes of neural model — plain RNNs and recursive neural tensor networks (RNTNs, [2]) — can learn those relationships correctly. In our first experiment (Section 3), we evaluate the ability of these models to learn the core relational algebra of natural logic from data. Our second experiment (Section 4) extends this evaluation to cover relations between complex recursive structures like $(a \textit{ or } b)$ and *not*$(\textit{not } a \textit{ and not } b)$, and our third experiment (Section 5) involves relations between quantified statements like *every reptile walks* and *every turtle moves*. We find that the plain RNN achieves only mixed results in all three experiments, whereas the stronger RNTN models generalized well in every case, suggesting that it has in fact learned, or at least learned to simulate, our target logical concepts.

Name	Symbol	Set-theoretic definition	Example
entailment	$x \sqsubset y$	$x \subset y$	*turtle, reptile*
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	$x \mathbin{^\wedge} y$	$x \cap y = \emptyset \wedge x \cup y = \mathcal{D}$	*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 entailment relations in $\mathfrak{B}$. $\mathcal{D}$ is the universe of possible objects of the same type as those being compared, and the relation $\#$ applies whenever none of the other six do, including when there is insufficient knowledge to choose a label.

2 Recursive neural network models

We study neural models that adhere to the linguistic *principle of compositionality*, which says that the meanings for complex expressions are derived from the meanings of their constituent parts via specific combination functions [20, 21]. In our distributed setting, word meanings are vectors of length N. The combination function concatenates pairs of them to form vectors of length $2N$ and maps those larger vectors back into length N vectors, which can then be merged again to create more complex phrases. Once the entire sentence-level representation has been derived, it is fed to a classifier trained in a supervised manner.

We use the model architecture proposed in [16] and depicted in Figure 1. The two phrases being compared are built up separately on each side of the tree, using the same composition function, until they have each been reduced to single vectors. 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 in turn produces a hypothesized distribution over the seven relations represented in Table 1.

For a composition layer, we evaluate models with both the plain neural network layer function (1) and the RNTN layer function proposed in Chen et al. [22] (2). A sigmoid nonlinearity (element-wise $\tanh$) is applied to the output of either layer function, following [2].

$$\vec{y}_{RNN} = f(\mathbf{M}[\vec{x}^{(l)}; \vec{x}^{(r)}] + \vec{b}) \tag{1}$$

$$\vec{y}_{RNTN} = f(\vec{x}^{(l)T}\mathbf{T}^{[1\ldots N]}\vec{x}^{(r)} + \mathbf{M}[\vec{x}^{(l)}; \vec{x}^{(r)}] + \vec{b}) \tag{2}$$

Here, the $\vec{x}^{(l)}$ 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 RNN concatenates them, multiplies them by an $N \times 2N$ matrix of learned weights, and applies the element-wise non-linearity to the resulting vector (plus a bias term). The RNTN has the same basic structure, but with the addition of a learned third-order tensor $\mathbf{T}$, dimension $N \times N \times N$, modeling multiplicative interactions between the child vectors. Both models include a learned bias vector, represented $\vec{b}$.

The comparison layer uses the same kind of function template as the composition function (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 a substantial improvement in performance by using a rectified linear function for f_b. In particular, we use the leaky rectified linear function [23]: $f_b(\vec{x}) = \max(\vec{x}, 0) + 0.01 \min(\vec{x}, 0)$, applied element-wise.

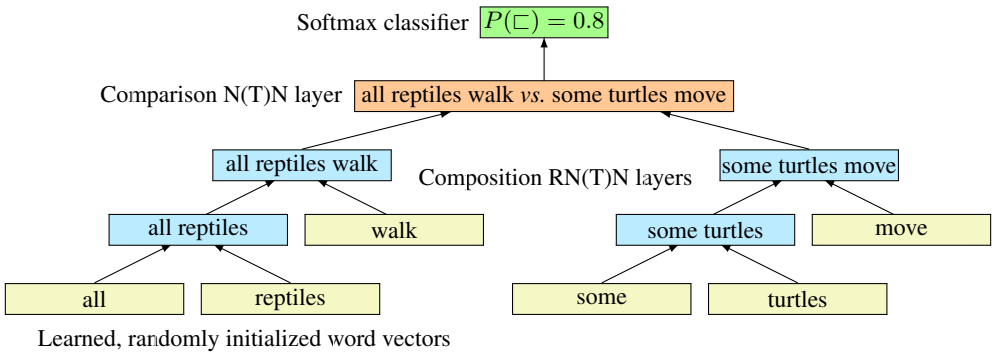


Figure 1: The model structure used to compare *((all reptiles) walk)* and *((some turtles) move)*. The same structure is used for both the RNN and RNTN layer functions.

To run the model forward and label a pair of phrases, the structure of the lower layers of the network is assembled so as to mirror the tree structures provided for each phrase. The word vectors are then looked up from the vocabulary matrix V (one of the model parameters), and the composition and comparison functions are used to pass information up the tree and into the classifier at the top. For an objective function, we use the negative log of the probability assigned to the correct label.

We train the model with minibatch stochastic gradient descent (SGD) with learning rates computed using AdaGrad [24] from a starting rate of 0.2. The parameters (including the vocabulary) are initialized randomly using a uniform distribution over $[-0.1, 0.1]$. Source code and generated data will be released after the conclusion of the review period.

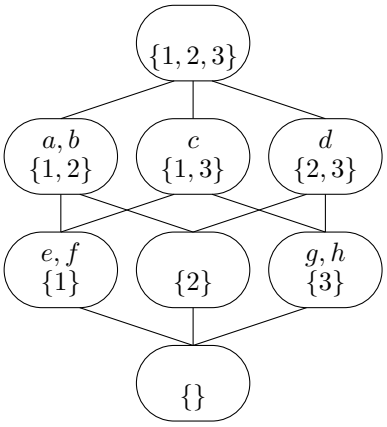
3 Relation composition

If a model is to learn the behavior of a relational logic like the one presented here from a finite amount data, it must be able to learn to deduce new relations from seen relations in a sound manner. The simplest such deductions involve atomic statements using the relations in Table 1. For instance, given that $a \sqsupset b$ and $b \sqsupset c$, one can conclude that $a \sqsupset c$, by basic set-theoretic reasoning (transitivity of $\sqsupset$). Similarly, from $a \sqsubset b$ and $b \mathbin{\wedge} c$, it follows that $a \mid c$. The full set of sound inferences of this form is summarized in Table 2; cells containing a dot correspond to pairs of relations for which no valid inference can be drawn in our logic.

	$\equiv$	$\sqsubset$	$\sqsupset$	$\wedge$	$\mid$	$\smile$	$\#$
$\equiv$	$\equiv$	$\sqsubset$	$\sqsupset$	$\wedge$	$\mid$	$\smile$	$\#$
$\sqsubset$	$\sqsubset$	$\sqsubset$	$\cdot$	$\mid$	$\mid$	$\cdot$	$\cdot$
$\sqsupset$	$\sqsupset$	$\cdot$	$\sqsupset$	$\smile$	$\cdot$	$\smile$	$\cdot$
$\wedge$	$\wedge$	$\smile$	$\mid$	$\equiv$	$\sqsupset$	$\sqsubset$	$\#$
$\mid$	$\mid$	$\cdot$	$\mid$	$\sqsubset$	$\cdot$	$\sqsubset$	$\cdot$
$\smile$	$\smile$	$\smile$	$\cdot$	$\sqsupset$	$\sqsupset$	$\cdot$	$\cdot$
$\#$	$\#$	$\cdot$	$\cdot$	$\#$	$\cdot$	$\cdot$	$\cdot$

Table 2: Inference path from premises aRS (row) and bSc (column) to the relation that holds between a and c, if any. These inferences are based on basic set-theoretic truths about the meanings of the underlying relations as described in Table 1. We assess our model's ability to reproduce such inferential paths.

To test our systems's ability to learn this relational structure, we create small models for our logic in which terms denote sets of entities from a single domain of seven entities (integers). Figure 2 depicts a small model of this form. The lattice structure gives the full model, for which all the statements on the right are valid. We divide these statements evenly into training and test sets, and remove from the test set those statements which cannot be proven from the training statements using the logic depicted in Figure 2.



(a) Simple boolean model. The letters name the sets. Not all sets have names, and some sets have multiple names, so that learning $\equiv$ is non-trivial.

Train	Test
	$b \equiv b$
$b \smile c$	
	$b \smile d$
	~~$b \sqsupset e$~~
$c \smile d$	
$c \sqsupset e$	
	~~$e \equiv f$~~
$c \sqsupset g$	
	$e \sqsubset b$
$e \sqsubset c$	
$\vdots$	$\vdots$

(b) An example train/test split from the full set of statements one can make about the model. Statements not provable from the test data are crossed out.

Figure 2: Some sample randomly generated sets, and some of the relations defined between them.

In our experiments, we create 80 randomly generated sets drawing from a domain of seven elements. This yields a data set consisting of 6400 statements about pairs of entities. 3200 of these pairs are chosen as a test set, and that test set is further reduced to the 2960 examples that can be provably derived from the training data.

We trained versions of both the RNN model and the RNTN model on these data sets. In both cases, the models were implemented exactly as described in Section 2, but since the items being compared are single terms rather than full tree structures, the composition layer was not involved, and the two models differed only in which layer function was used for the comparison layer. We simply present the models with two single terms, each of which corresponds to a single vector in the (randomly initialized) vocabulary matrix V, ensuring that the model has no information about the terms being compared except the relations between them.

Results. We found that the RNTN model worked best with 11-dimensional vector representations for the 80 sets and a 90-dimensional feature vector for the classifier, though the performance of the model was not highly sensitive to either dimensionality parameter in any of the experiments discussed here. This model was able to correctly label 99.3% of the provably derivable test examples, and 99.1% of the remaining test examples. The simpler RNN model worked best with 11 and 75 dimensions, respectively, but was able to achieve accuracies of only 90.0% and 87.0%, respectively.

These results are fairly straightforward to interpret. The RNTN model was able to accurately encode the relations between the terms in the geometric relations between their vectors, and was able to then use that information to recover relations that were not overtly included in the training data. In contrast, the RNN model was able to approximate this behavior only incompletely. It is possible but not likely that it could be made to find a good solution with further optimization on different learning algorithms, or that it would do better on a larger universe of sets for which there would be a larger set of training data to learn from, but the RNTN is readily able to achieve these effects in the setting discussed here.

4 Recursive structure in propositional logic

Recursive structure is a prominent feature of natural languages and an important part of their expressivity, as it allows people to use and interpret complex structures they have never encountered before. Theoretical accounts of natural language syntax and semantics therefore rely heavily on recursive structures, and we expect that accurate computational models will have to come to grips with them as well. The present section pursues this question for our two classes of RNN. In evaluating the model, we take advantage of the fact that recursive structures of this kind define potentially

infinite languages, by testing the model on strings that are longer and more complex than any seen in training.

As in Section 3, we test this phenomenon within the framework of natural logic, but we now replace the unanalyzed symbols from that experiment with expressions that involve recursive structure. These expressions represent a truth-functionally complete classical propositional logic: each atomic variable denotes either T or F, and the only operators are truth-functional ones. Table 3a defines this logic, and Table 3b gives some examples of relational statements that we can make in these terms. To compute these relations between statements, we exhaustively enumerate the sets of assignments of truth values to proposition variables that would satisfy each of the statements and then 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.

Formula	Interpretation
a, b, c, d, e, f	$[\![x]\!] \in \{\mathtt{T}, \mathtt{F}\}$
not φ	T iff $[\![\varphi]\!] = \mathtt{F}$
(φ *and* ψ)	T iff $\mathtt{F} \notin \{[\![\varphi]\!], [\![\psi]\!]\}$
(φ *or* ψ)	T iff $\mathtt{T} \in \{[\![\varphi]\!], [\![\psi]\!]\}$

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

not a	$\wedge$	a
not not a	$\equiv$	a
a	$\sqsubset$	(a *or* b)
a	$\sqsupset$	(a *and* b)
($a \sqsubset b$)	$\equiv$	($b \sqsupset a$)
not(*not* a *and not* b)	$\equiv$	(a *or* b)

(b) Examples of statements about relations between well-formed formulae, defined in terms of sets of satisfying interpretation functions $[\![\cdot]\!]$.

Table 3: Propositional logic with natural logic relations.

Socher et al. [25] demonstrate the learning of a logic in a matrix-vector RNN model somewhat similar to our own, but the logic discussed here is substantially stronger, and a much better approximation of the kind of structure that is needed for natural language. The logic learned in that experiment is boolean, wherein the atomic symbols are simply the values 0 and 1, rather than variables over those values. While learning the operators of that logic is not trivial, the outputs of each operator can be represented accurately by a single bit. The statements of propositional logic learned here describe conditions on the truth values of propositions where those truth values are not known. As opposed to the two-way contrasts seen in [25], this logic distinguishes between $2^6 = 64$ possible assignments of truth values, and expressions of this logic define arbitrary conditions on these possible assignments, for a total of 2^{64} possible statements that the intermediate vector representations need to be able to distinguish.

For our experiments, we randomly generate a large set of unique pairs of formulae and compute the relation that holds for each pair. We discard pairs in which either statement is a tautology or contradiction (a statement that is true of either all or no possible assignments), for which none of the seven relation labels in Table 1 can hold. The resulting set of formula pairs is then partitioned into bins based on the number of logical operators in the longer of the two formulae. We then randomly sample 15% 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, the generated data consists in large part of statements in which the two formulae refer to largely separate subsets of the six variables, and to which we will nearly always assign the # relation. 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 proposition variables.

Results. We trained both the RNN and RNTN models on the data of size four or less (65k pairs), and tested it on examples of up to size 12 (44k pairs). We initialized the model parameters randomly, including the vector representations of the six variables. The results are shown in Figure 3. In tuning, we found that the RNN model was approximately optimal with 45-dimensional vector representations, and the RNTN model was approximately optimal with 25 dimensions. We fixed the size of the feature vector for the classifier at 75 dimensions. We found that the RNTN model was able to perform almost perfectly on unseen small test examples, with accuracy above 99% below size four. After depth four, performance gradually falls with increasing size. The RNN model did

not perform well, reaching only 88.2% accuracy on the smallest test examples, and declining from there to near-baseline performance at size 12.

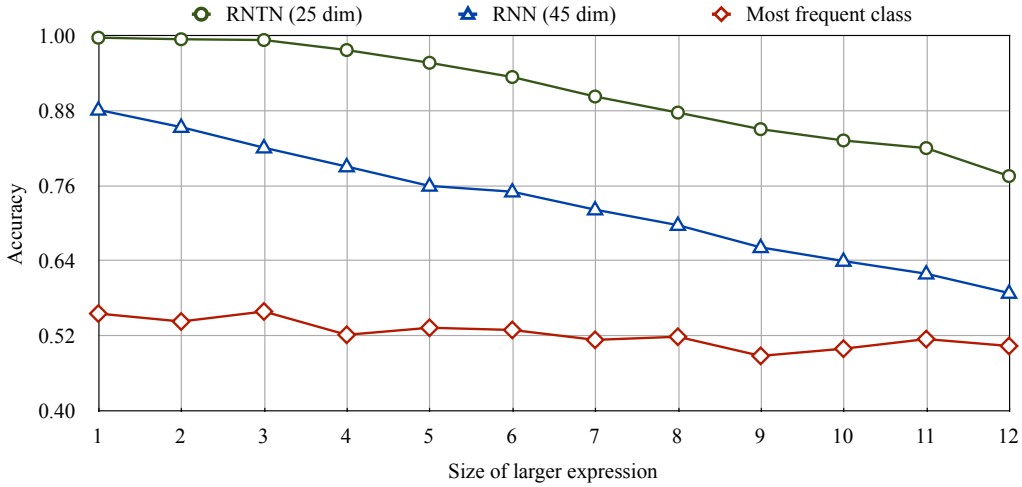


Figure 3: Model performance on propositional logic, by expression size.

The performance of the RNTN model on small unseen test examples indicates that it learned a correct approximation of the underlying logic. It appears that this approximation is accurate enough to yield correct answers when the composition layer is only applied a small number of times, but that the error in the approximation grows with increasing depth (and with the increasing complexity of the expressions), resulting in the gradually dropping performance. This is not necessarily a significant flaw in the model, since it remains possible that a less lossy approximation of the same logic would do no better on the training objective, since that objective only considers short strings and includes a regularization term. Whether a more accurate encoding could be learned from a larger set of training data remains a question for future work. In contrast, the RNN model did not appear to be able to learn a correct approximation of the logic.

5 Reasoning with natural language quantifiers and negation

Even to the extent that RNTN models can handle functional meanings in the form of the operators of propositional logic, this is not a guarantee that these models can handle functional meanings of the form seen in natural language. As a first step towards investigating this issue, we attempt to directly measure the degree to which RNNs are able to develop suitable representations for the semantics of natural language quantifiers like *some* and *all*. Quantification is far from the only place in natural language where complex functional meanings are found, but it is a natural starting point, since it can be tested in sentences whose structures are otherwise quite simple, and since it has formed a standard case study in prior formal work on natural language inference.

This experiment replicates similar work described in [16], which found that RNTNs can learn to reason well with quantifier meanings given sufficient training data. This paper replaces the partially manually annotated data in that paper with data that is generated directly using the logical system that we hope to model, yielding results that we believe to be substantially more straightforward to interpret.

Our data consists of pairs of sentences generated from a small artificial grammar. Each sentence contains a quantifier, a noun which may be negated, and an intransitive verb which may be negated. We use the basic quantifiers *some*, *most*, *all*, *two*, and *three*, and their negations *no*, *not-all*, *not-most*, *less-than-two*, and *less-than-three*. We 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 between each noun–noun pair, each verb–verb pair, and each quantifier–quantifier pair. The grammar accepts aligned pairs of sentences of this form and calculates the natural logic relationship between them. Some examples of these data are provided

in Table 4. As in previous sections, the goal of learning is then to assign these relational labels accurately to unseen pairs of sentences.

(most warthogs) walk	$\wedge$	(not-most warthogs) walk
(most mammals) move	$\#$	(not-most (not turtles)) move
(most (not pets)) (not swim)	$\sqsupset$	(not-most (not pets)) move
(no turtles) (not growl)	$\mid$	(no turtles) (not swim)
(no warthogs) swim	$\sqsupset$	(no warthogs) move
(no warthogs) move	$\sqsubset$	(no (not reptiles)) swim

Table 4: Sample data involving two different quantifier pairs.

We evaluate the model using two experimental settings. In the simpler setting, ALL SPLIT, we randomly sample 85% of the data and evaluate on the remaining 15%. In this setting, the model is being asked to learn a complete reasoning system for the limited language and logic presented in the training data, but it is not being asked to generalize to test examples that are substantially different from those it was trained on. Crucially though, to succeed on this task, the model must be able to recognize all of the lexical relations between the nouns, verbs, and quantifiers and how they interact. For instance, it might see (3) and (4) in training and, from that information, determine (5).

(3) (most turtle) swim $\mid$ (no turtle) move

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

(5) (most turtle) reptile $\mid$ (no turtle) animal

While our primary interest is in discovering the extent to which our models can learn to encode the logic given an arbitrary amount of data, we are also interested in the degree to which they can infer a correct representation for the logic from more constrained training data. To this end, we segment the sentence pairs according to which quantifiers appear in each pair, and then hold out one such pair for testing. We hypothesize that a model that can efficiently learn to represent a logic should be able to construct an accurate representation of each held out quantifier from the way that it interacts with the other nine quantifiers which are not held out. Since running this experiment requires choosing a pair of quantifiers to hold out before training, the resource demands of training prevent us from testing each of the 55 possible possible pairs of quantifiers, and we choose only four pairs to test on. Three of these (*two*/*less-than-two*, *not-all*/*not-most*, and *all*/*some*) were chosen because they allow for the most different class labels at in the training data. The fourth is a self-pair (*no*/*no*), meant to test that the model correctly handles equality.

Data	Most frequent class	TODO dim RNN (%)	16 dim RNTN (%)
ALL SPLIT	35.4	63.9	**93.4**
PAIR TWO/LESS-THAN-TWO	59.8	79.5	**98.3**
PAIR NOT-ALL/NOT-MOST	0	62.2	**83.3**
PAIR ALL/SOME	0	63.6	**81.0**
PAIR NO/NO	30.8	66.0	**100**

Table 5: Quantifier experiment performance.

Results. The results for these experiments are shown in Figure 5. We compare the results to a most frequent class baseline, which reflects the frequency in the test data of most frequent class in the training data, #. The RNN model was approximately optimal with N dimensional word representations and an M dimensional comparison layer. The RNTN was approximately optimal with N and M dimensions, respectively.

The perfect performance by the RNTN on the PAIR NO/NO experiment and its strong performance on PAIR TWO/LESS-THAN-TWO suggests that it is at least plausible that this model is able to handle quantifiers correctly given sufficient training data, but the sub 90% results on two of the training settings suggest that the model may not be able to generalize from data this impoverished in general.

However, the fact that both models perform far above baseline is promising, and the question of how much data is necessary to accurately capture quantifier behavior in a naïve model remains open.

6 General discussion

This paper evaluated two RNNs in a series of three increasingly challenging interpretive tasks involving natural language inference: the core relational algebra of natural logic with entailment and exclusion; recursive propositional logic structures; and statements involving quantification and negation. The results suggest that RNTNs, but not plain RNNs, have the capacity to meet the challenges of these tasks with reasonably-sized training sets. These positive results are promising for the future of learned representation models in the applied modeling of compositional semantics.

Of course, challenges remain. In terms of our experimental data, even the RNTN falls short of perfection in our more complex tasks, with performance falling off steadily as the depth of recursion grows. It remains to be seen whether these deficiencies can be overcome with improvements to the model, the optimization procedures, or the linguistic representations [3, 12]. In addition, there remain subtle questions about how to fairly assess whether these models have truly generalized in the way we want them to. There is a constant tension between giving the models training data that gives them a chance to learn the target logical functions and revealing the answer to them in a way that leads to overfitting. The underlying logical theories provide only limited guidance on this point, and the fact that there is a finite universe of possible expressions makes this an unavoidable issue. Finally, we have only scratched the surface of the logical complexity of natural language; in future experiments, we hope to test sentences with embedded quantifiers, multiple interacting quantifiers, relative clauses, and other kinds of recursive structure. Nonetheless, the rapid progress the field has made with these models in recent years provides ample reason to be optimistic that they can be trained to meet the challenges of natural language semantics.

Bibliography

[1] Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proc. EMNLP*, 2011.

[2] Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. EMNLP*, 2013.

[3] Richard Socher, Quoc V Le, Christopher D Manning, and Andrew Y Ng. Grounded compositional semantics for finding and describing images with sentences. *TACL*, 2014.

[4] Richard Socher, Eric H. Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y. Ng. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In John Shawe-Taylor, Richard S. Zemel, Peter L. Bartlett, Fernando Pereira, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 801–809, 2011.

[5] David H. D. Warren and Fernando C. N. Pereira. An efficient easily adaptable system for interpreting natural language queries. *American Journal of Computational Linguistics*, 8(3–4):110–122, 1982.

[6] John M. Zelle and Raymond J. Mooney. Learning to parse database queries using inductive logic programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence – Volume 2*, pages 1050–1055. AAAI Press, 1996.

[7] Luke S. Zettlemoyer and Michael Collins. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Twenty First Conference on Uncertainty in Artificial Intelligence*, 2005.

[8] Percy Liang, Michael I. Jordan, and Dan Klein. Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2):389–446, 2013.

[9] Jeff Mitchell and Mirella Lapata. Composition in distributional models of semantics. *Cognitive Science*, 34(8):1388–1429, 2010.

[10] Edward Grefenstette, Mehrnoosh Sadrzadeh, Stephen Clark, Bob Coecke, and Stephen Pulman. Concrete sentence spaces for compositional distributional models of meaning. In *Proceedings of the 9th International Conference on Computational Semantics (IWCS 2011)*, pages 125–134, 2011.

[11] Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. Entailment above the word level in distributional semantics. In *Proc. EACL*, 2012.

[12] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

[13] Ido Dagan, Oren Glickman, and Bernardo Magnini. The PASCAL Recognising Textual Entailment Challenge. In *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, pages 177–190. Springer, 2006.

[14] Bill MacCartney. *Natural language inference*. PhD thesis, Stanford University, 2009.

[15] Yotaro Watanabe, Junta Mizuno, Eric Nichols, Naoaki Okazaki, and Kentaro Inui. A latent discriminative model for compositional entailment relation recognition using natural logic. In *COLING*, pages 2805–2820, 2012.

[16] Samuel R. Bowman. Can recursive neural tensor networks learn logical reasoning? *arXiv preprint arXiv:1312.6192. Presented at ICLR*, 2014.

[17] B. MacCartney and C.D. Manning. An extended model of natural logic. In *Proc. International Conference on Computational Semantics*, 2009.

[18] Thomas F. Icard and Lawrence S. Moss. A complete calculus of monotone and antitone higher-order functions. In Nikolaos Galatos, Alexander Kurz, and Constantine Tsinakis, editors, *Proceedings of Topology, Algebra, and Categories in Logic*, pages 96–99, July 2013.

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

[20] Barbara H. Partee. Compositionality. In Fred Landman and Frank Veltman, editors, *Varieties of Formal Semantics*, pages 281–311. Foris, Dordrecht, 1984.

[21] Theo M. V. Janssen. Compositionality. In Johan van Benthem and Alice ter Meulen, editors, *Handbook of Logic and Language*, pages 417–473. MIT Press and North-Holland, Cambridge, MA and Amsterdam, 1997.

[22] Danqi Chen, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. Learning new facts from knowledge bases with neural tensor networks and semantic word vectors. In *Proc. ICLR*, 2013.

[23] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML 30*, 2013.

[24] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 2011.

[25] Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. Semantic compositionality through recursive matrix-vector spaces. In *Proc. EMNLP*, 2012.