
Recursive Neural Networks for Learning Logical Semantics

Samuel R. Bowman^{*†}
sbowman@stanford.edu

Christopher Potts^{*}
cgpotts@stanford.edu

Christopher D. Manning^{*†‡}
manning@stanford.edu

^{*}Stanford Linguistics

[†]Stanford NLP Group

[‡]Stanford Computer Science

Supervised recursive neural network models (RNNs) for sentence meaning have been successful in an array of sophisticated language tasks, but it remains an open question whether they can learn compositional semantic grammars that support logical deduction. We address this question directly by for the first time evaluating whether each of two classes of neural model — plain RNNs and recursive neural tensor networks (RNTNs) — can correctly learn relationships such as entailment and contradiction between pairs of sentences, where we have generated controlled data sets of sentences from a logical grammar. 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 model generalizes well in every setting and appears capable of learning suitable representations for natural language logical inference.

1 Introduction

Supervised recursive neural network models (RNNs) for sentence meaning have been successful in an 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 the core semantic concepts of quantification, entailment, and contradiction needed to identify the relationships between sentences like *every animal walks*, *every turtle moves*, and *most reptiles don't move*. 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, it is essential 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 [14] 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 [17, 18].

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 relations between complex recursive structures like *(a or b)*

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

Table 1: The seven natural logic relations of [14]. \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.

and *not(not a and not b)*, and our third experiment (Section 5) involves relations between quantified statements like *every reptile walks* and *no turtle moves*. We find that the plain RNN achieves only mixed results in all three experiments, whereas the stronger RNTN model generalizes well in every case, suggesting that it has in fact learned, or at least learned to simulate, our target logical concepts. These experiments differentiate the increased power of RNTNs better than previous work and provide the most convincing demonstration to date of the ability of neural networks to model semantic inferences in complex natural language sentences.

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 composition functions [19, 20]. In our distributed setting, word meanings are vectors of length n . The composition function maps pairs of them to single vectors of length n , which can then be merged again to represent more complex phrases. Once the entire sentence-level representation has been derived, it serves as a fixed-length input for some subsequent function.

We use the model architecture proposed in [16] and depicted in Figure ?? . 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 (2) proposed in Chen et al. [21]. A sigmoid nonlinearity (element-wise \tanh) is applied to the output of either layer function, following [2].

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

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

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 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. 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 bias vector \vec{b} .

The comparison layer uses the same kind of function template 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 follow [16] in using a rectified linear function. In particular, we use the leaky rectified linear function [22]: $f(\vec{x}) = \max(\vec{x}, 0) + 0.01 \min(\vec{x}, 0)$, applied element-wise.

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 using stochastic gradient descent (SGD) with learning rates computed using AdaGrad [23]. The parameters (including the vocabulary) are initialized randomly using a uniform distribution over $[-0.1, 0.1]$. Because the seven classes are not balanced in general, we report performance using both accuracy and macroaveraged F1, since the latter emphasizes performance on infrequent classes. We compute macroaveraged F1 as the harmonic mean of mean precision and mean recall, both computed for all classes for which there is test data, setting precision to 0 where it is not defined. Source code and generated data will be released after the review period.

3 Reasoning about semantic relations

If a model is to learn the behavior of a relational logic like the one presented here from a finite amount of data, it must 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 $b \wedge c$, it follows that $a \mid c$. The full set of sound inferences of this form is summarized in Table ??; cells containing a dot correspond to pairs of relations for which no valid inference can be drawn in our logic.

Experiments To test our models’ ability to learn these inferential patterns, we create small boolean structures for our logic in which terms denote sets of entities from a small domain. Figure 1 depicts a structure of this form. The lattice 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.

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 formulae. 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 , thereby 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 99.4% (94.3 macroaveraged F1) of the provably derivable test examples, and 95.4% (92.2 macro-F1) 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 89.0% (80.7 macro-F1) and 62.5% (40.7 macro-F1), respectively. Training accuracy was 99.8% for the RNTN and 95.4% for the RNN.

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 achieve 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.

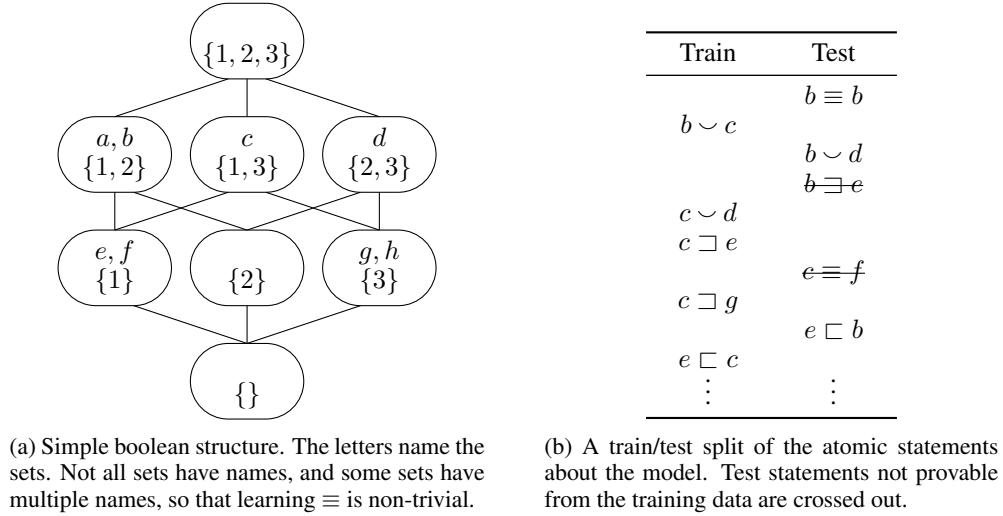


Figure 1: Small example structure and data for learning relation composition.

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 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 Section 3, we test this phenomenon within the framework of natural logic, but we now replace the unanalyzed symbols from that experiment with complex formulae. These formulae represent a truth-functionally complete classical propositional logic: each atomic variable denotes either \top or \bot , and the only operators are truth-functional ones. Table ?? defines this logic, and Table ?? 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 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.

Socher et al. [24] show that a matrix-vector RNN model somewhat similar to our RNTN can learn a logic, but the logic discussed here is a much better approximation of the kind of expressivity 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 unknown truth values of propositions. As opposed to the two-way contrasts seen in [24], 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 if we are to learn this logical system in its full generality.

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.

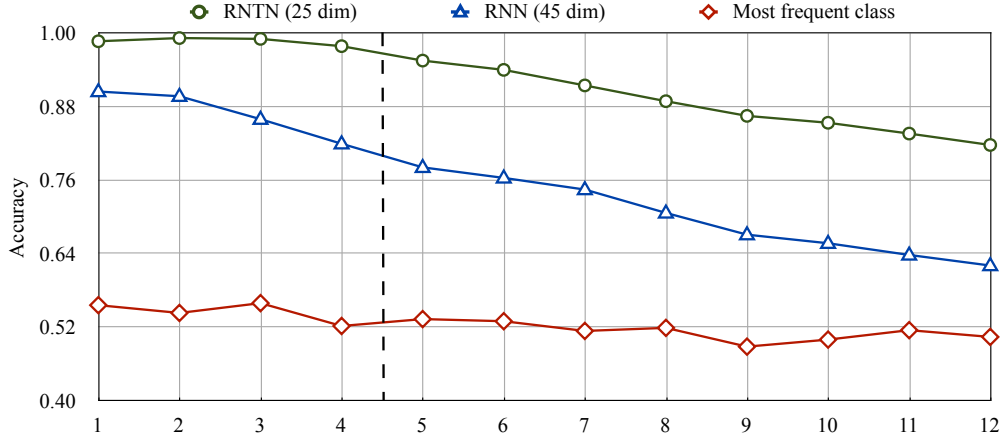


Figure 2: Model performance on propositional logic, by expression size. The dashed line indicates that only expressions of size four or less appeared in the training data.

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.

Results We trained both the RNN and RNTN models on data with up to 4 connectives (65k pairs) and tested it on examples of up to 12 connectives (44k pairs). We initialized the model parameters randomly, including the vector representations of the six variables. The results are shown in Figure 2. 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 98.7% on formulae below length four (macro-F1 was above 95.5). Starting at size four, performance gradually falls with increasing size. The RNN model did not perform well, reaching only 90.5% (81.4 micro-F1) accuracy on the smallest test examples, and declining from there to near-baseline performance on formulae of length 12. Training accuracy was 99.6% for the RNTN and 87.6% for the RNN.

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 gradually dropping performance. In contrast, the RNN model did not appear to be able to learn a correct approximation of the logic for statements of any length, but it appears that the faulty logic that the RNN did learn decayed similarly with expression size. This is not necessarily a significant flaw in the model, since it remains possible that different training regimes would allow it to learn accurate representations of the larger expressions.

In an effort to better understand why the RNTN outperformed the plain RNN, we evaluated both models on pairs of long formulae involving binary connectives, to assess the extent to which they achieve substantially different representations for semantically distinct formulae and substantially similar representations for synonymous formulae. We found that both models are able to identify synonymous pairs like $(a \text{ and } a)$ and $(a \text{ and } (a \text{ and } (a \text{ and } a)))$, with both *and* and *or*, but only the RNTN is able to learn substantially different representations for pairs of differing formulae like $(a \text{ and } (a \text{ and } a))$ and $(a \text{ and } (a \text{ and } b))$ when the difference between the two is embedded under multiple operators. Figure ?? summarizes this result.

5 Reasoning with natural language quantifiers and negation

We have seen that the RNTN can learn an approximation of propositional logic. However, natural languages can express functional meanings of considerably greater complexity than this. As a first step towards investigating whether our models can capture this complexity, we now attempt to directly measure the degree to which they 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 [18].

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 learn, yielding results that we believe to be substantially more straightforward to interpret.

Experiments Our experimental data consist 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 ?? . As in previous sections, the goal of learning is then to assign these relational labels accurately to unseen pairs of sentences.

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 | (no turtle) move
- (4) (all lizard) reptile \sqsubset (some lizard) animal
- (5) (most turtle) reptile | (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 of the two quantifiers in the pair from the way that it interacts with the other nine quantifiers with which it is seen in training. 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 pairs of quantifiers, so 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 in the test data. The fourth is a self-pair (*no/no*), meant to test that the model correctly handles equality.

Results The results for these experiments are shown in Figure ?? . We compare the results to a most frequent class baseline, which reflects the frequency in the test data of the most frequent class in the training data, #. After some cross-validation, we chose 16- and 20-dimensional representations for the RNN and RNTN, respectively, and 75-dimensional feature vectors for the classifier. Training accuracy was near 80% for each of the five RNN experiments, and near 98% for each of the five RNTN experiments.

Data	Most frequent class		16 dim RNN		20 dim RNTN	
ALL SPLIT	35.4	(7.5)	67.4	(56.5)	100.0	(100.0)
PAIR TWO/LESS-THAN-TWO	59.8	(18.4)	77.2	(57.4)	100.0	(100.0)
PAIR NOT-ALL/NOT-MOST	0	(0)	66.0	(54.3)	93.8	(91.5)
PAIR ALL/SOME	0	(0)	62.7	(65.9)	78.4	(84.7)
PAIR NO/NO	30.8	(10.2)	67.8	(61.8)	99.9	(99.7)

Table 2: Performance on the quantifier experiments. Results are reported as accuracy scores followed by macroaveraged F1 scores in parentheses.

The RNN performed poorly at this task, even though the sentences used in these examples are short enough to avoid the pathology shown in Figure ?? . However, the perfect performance by the RNTN on the ALL SPLIT and PAIR TWO/LESS-THAN-TWO experiments, and its strong performance on the PAIR NO/NO experiment, suggest that this stronger model is able to handle quantifiers correctly given sufficient training data, but the weaker results on two of the training settings suggest that it may not be able to generalize from this kind of restricted training data in general. However, though the question of how much data is necessary to accurately capture quantifier behavior in a naïve model remains open, the fact that this model is able to perform so well is promising.

6 General discussion

This paper evaluated two recursive models on 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 showing 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. 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.

References

- [1] Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of EMNLP*, pages 151–161, 2011.
- [2] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*, pages 1631–1642, 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 Pennington, Christopher D. Manning, and Andrew Y. Ng. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Proceedings of NIPS*, 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 AAAI*, volume 2, pages 1050–1055, 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 21st 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 Sadzadeh, Stephen Clark, Bob Coecke, and Stephen Pulman. Concrete sentence spaces for compositional distributional models of meaning. In *Proceedings of IWCS*, 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 *Proceedings of EACL*, pages 23–32, 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 Textual Entailment*, pages 177–190. Springer, 2006.
- [14] Bill MacCartney and Christopher D. Manning. An extended model of natural logic. In *Proceedings of IWCS*, pages 140–156, 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 *Proceedings of 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] 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, 2013.
- [18] Thomas F. Icard and Lawrence S. Moss. Recent progress on monotonicity. *Linguistic Issues in Language Technology*, 9(7):1–31, 2013.
- [19] Barbara H. Partee. Compositionality. In Fred Landman and Frank Veltman, editors, *Varieties of Formal Semantics*, pages 281–311. Foris, 1984.
- [20] 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, 1997.
- [21] 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 *Proceedings of ICLR*, 2013.
- [22] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of ICML*, 2013.
- [23] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, pages 2121–2159, 2011.
- [24] Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of EMNLP*, pages 1201–1211, 2012.