DRAFT: Learning to represent abstract structures for natural language reasoning

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 they can achieve the same results as compositional semantic grammars based in logical forms. We address this question directly by using a logical grammar to generate controlled data sets encoding the relationships (entailment, synonymy, 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 confirms that both models can learn the basic algebra of logical relations involved. Our second and third experiments expand on this result to cover complex recursive structures and sentences involving quantification. We find that the plain RNN achieves only mixed results on the second and third experiments, but that the stronger RNTN models generalized well in every case.

1 Introduction

Deep learning methods in NLP which learn vector representations for words have seen successful uses in recent years on increasingly sophisticated tasks [1, 2, 3, 4]. Given the still modest performance of semantically rich NLP systems in many domains—question answering and machine translation, for instance—it is worth exploring the degree to which learned vectors can serve as general purpose semantic representations. Much of the work to date analyzing vector representations for words (see [5]) has focused on lexical semantic behaviors—like the similarity between words like *Paris* and *France*. Good similarity functions are valuable for many NLP tasks, but there are real use cases for which it is necessary to know how words relate to one another or to some extrinsic representation, and to model the ways in which word meanings combine to form phrase, sentence, or document meanings. This paper explores the ability of linguistic representations developed using supervised deep learning techniques to support interpretation and reasoning.

There are two broad family of tasks that could be used to test the ability of a model to develop general purpose meaning representations. In an interpretation task, sentences are mapped onto some denotation, such as *true* or *false* for statements, or a factual answer for questions. There has been preliminary work in developing distributed models for interpretation [6, 7], but developing a representation of world knowledge that corresponds accurately to the content expressed in language introduces considerable design challenges. I approach the problem by way of an inference task instead. Inferring the truth of one sentence from another does not require any preexisting knowledge representations, but nonetheless requires a precise representation of sentence meaning. I borrow the structure of the task from MacCartney and Manning [8]. In it, the model is presented with a pair of sentences, and made to label the logical relation between the sentences as equivalence, entailment, or any of five other classes, as here:

name	symbol	set-theoretic definition	example
entailment	$x \sqsubset y$	$x \subset y$	*crow, bird*
reverse entailment	$x \sqsupset y$	$x \supset y$	*Asian, Thai*
equivalence	$x \equiv y$	$x = y$	*couch, sofa*
alteration	$x \mid y$	$x \cap y = \emptyset \wedge x \cup y \neq \mathcal{D}$	*cat, dog*
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-ape*
independence	$x \# y$	(else)	*hungry, hippo*

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.

1. Many smartphone users avoid high bills overseas by disabling data service.
2. Not everyone uses their smartphones for email when traveling abroad.

$\Rightarrow$ Entailment

In this paper, we test the ability of recursive models to on three simple tasks, each of which is meant to capture a property that is necessary in representing natural language meaning in the setting of inference. I begin with an overview of MacCartney and Manning's [8] framework for inference, and of the recursive neural networks that I study. by showing that these models can learn to correctly represent entailment representations between sentences. I then show that these models can capture the meanings of recursive structurers accurately up to a sufficient depth. I finally close with a brief demonstration of the ability of these models to reason over short natural language sentences involving quantifiers.

1.1 The task: natural language inference

In the standard formulation of the inference task (and the one used in the RTE datasets [9]), the goal is to determine whether a reasonable human would infer a hypothesis from a premise. MacCartney formalizes a method of inferring entailment relations, and moves past two way entailment/non-entailment classification, proposing the set $\mathfrak{B}$ of seven labels meant to describe all of the possible non-trivial relations that might hold between pairs of statements, shown in Table 1.

In order to minimize this possibility, I define the task of strict unambiguous NLI (SU-NLI). In this task, only entailments that are licensed by a strictly literal interpretation of the provided sentences are considered valid, and several constraints are applied to the language to minimize ambiguity:

- A small, unambiguous vocabulary is used.
- All strings are given an explicit unlabeled tree structure parse.
- Statements involving the hard-to-formalize generic senses of nouns—i.e. *dogs bark* as opposed to the non-generic *all dogs bark*—are excluded.
- The sentences do not contain context dependent elements. This includes any reference to time or any tense morphology, and all pronouns.
- The morphology is dramatically simplified: the copula is not used (*some puppy is French* is simplified to *some puppy French*, to make it more directly comparable to sentences like *some puppy bark*), and agreement marking (*they walk* vs. *she walks*) is omitted.

The key to success at this task is to learn a set of representations and functions that can mimic the logical structure underlying the data. There is limited precedent that deterministic logical behavior can be learned in supervised deep learning models: Socher et al. [10] show in an aside that a Boolean logic with negation and conjunction can be learned in a minimal recursive neural network model with one-dimensional (scalar) representations for words. Modeling the logical behavior that underlies linguistic reasoning, though, is a substantially more difficult challenge, even in modular hand-built models.

The natural logic engine at the core of MacCartney's [11] NLI system requires a complex set of linguistic knowledge, much of which takes the form of what he calls projectivity signatures. These signatures are tables showing the entailment relation that must hold between two strings that differ in a given way (such as the substitution of the argument of some quantifier), and are explicitly provided to the model for dozens of different cases of insertion, deletion and substitution of different types of lexical item. For example in judging the inference *no animals bark* $|$ *some dogs bark* it would first used stored knowledge to compute the relations introduced by each of the two differences between the sentences. Here, the substitution of *no* for *some* yields $^\wedge$ and the substitution of *dogs* for *animals* yields $\sqsupset$. It would then use an additional store of knowledge about relations to resolve the resulting series of relations into one ($|$) that expresses the relation between the two sentences being compared:

1. *no animals bark* $^\wedge$ ***some*** *animals bark*
2. *some animals bark* $\sqsupset$ *some* ***dogs*** *bark*
3. *no animals bark* $[^\wedge \bowtie \sqsupset = |]$ *some dogs bark*

This study is the first that I am aware of to attempt to build an inference engine based on learned vector representations, but two recent projects have attempted to introduce vector representations into inference systems in other ways: Baroni et al. [12] have achieved limited success in building a classifier to judge entailments between one- and two-word phrases (including some with quantifiers), though their vector representations were crucially based on distributional statistics and were not learned for the task. In another line of research, Garrette et al. [13] propose a way to improve standard discrete NLI with vector representations. They propose a deterministic inference engine (similar to MacCartney's) which is augmented by a probabilistic component that evaluates individual lexical substitution steps in the derivation using vector representations, though again these representations are not learned, and no evaluations of this system have been published to date.

2 Methods

The model that I study is a recursive neural network model Socher et al. [14], and is thus centered on a recursively applied composition function which is meant to mimic the recursive construction of meanings in formal models of semantics. In this scheme, pairs of words are merged into phrase representations by a function that maps from representations of length $2N$ to representations of length N. These phrase representations are then further merged with other words and phrases until the entire phrase or sentence being evaluated is represented in a single vector. This vector is then used as the input to a classifier and used in a supervised learning task.

Borrowing a model directly from the existing literature for this task is impossible since none has been proposed to detect asymmetric relations between phrases. Instead, I build a combination model, 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. Then, the two phrase 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.

For a composition layer, I experiment with both a plain neural network layer function (eq. 2) and the RNTN layer function proposed in Chen et al. [4] (eq. 2). A sigmoid nonlinearity (elementwise $\tanh$) is applied to the output of either layer function, following Socher et al.

$$(1) \qquad y_{RNN} = f(M[\vec{x}^{(l)};\vec{x}^{(r)}] + b_i) \qquad y^i_{RNTN} = f(\vec{x}^{(l)T}\mathbf{A}^{[\mathbf{i}]}\vec{x}^{(r)} + \vec{B}_{i,:}[\vec{x}^{(l)};\vec{x}^{(r)}] + c_i)$$

The comparison layer uses the same type of function with different parameters and a different nonlinearity function wrapped around it. Rather than use a $\tanh$ nonlinearity here, I found a substantial improvement in performance by using a rectified linear function for f_b. In particular, I use the leaky rectified linear function [15]: $f_b(\vec{x}) = \max(\vec{x}, 0) + 0.01\min(\vec{x}, 0)$, applied elementwise.

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.

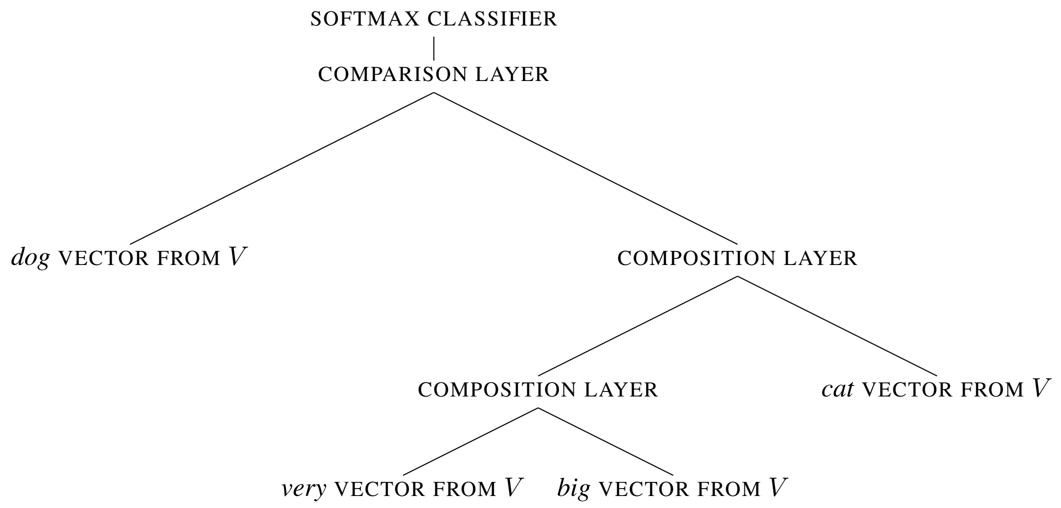

Figure 1: The model structure used to compare *dog* and *(very big) cat*.

$$
\begin{array}{l|ll}
a := \{1,3,4,5,6\} & a \equiv a & a \# c \\
b := \{0,3,5,6\} & e \sqsupset c & b \smile c \\
c := \{1,2,3,4\} & d \smile e & b \# d \\
d := \{0,4\} & a \sqsubset e & d \equiv d \\
e := \{1,2,3,4,5,6\} & a \# b & \ldots
\end{array}
$$

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

The model is trained using backpropagation through structure [16], wherein the negative log of the probability assigned to the correct label is taken as a cost function, and the gradient of each parameter with respect to that cost function is computed at each node, with information passing down the tree and into both the function parameters and the vocabulary matrix. Gradients for different instances of the composition function or different instances of a word in the same tree are summed to produce at most a single gradient update per parameter.

Optimization: I train the model with stochastic gradient descent (SGD), with gradients pooled from randomly chosen minibatches of 32 training examples, and learning rates computed using AdaGrad [17] from a starting rate of 0.2. The parameters (including the vocabulary) are initialized randomly using a uniform distribution over $[-0.1, 0.1]$.

Model size: The dimensionalities of the model were tuned extensively. The performance of the model is approximately optimal with the dimensionality of the word vectors set to 16, and the dimensionality of the feature vector produced by the comparison layer set to 45.

Source code and data will be released after the conclusion of the review period.

3 Relation composition

In order for a model to be able to learn to mimic 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. The simplest such deductions involve facts about the relations themselves that do not involve considering the internal structure of the things being compared. For example, given that $a \sqsupset b$ and $b \sqsupset c$ one can conclude that $a \sqsupset c$ by the transitivity of $\sqsupset$, even without understanding a, b, or c. These seven relations support more than just transitivity: MacCartney and Manning's [8] join table defines 32 valid inferences that can be made on the basis of pairs of relations of the form aRb and $bR'c$, including several less intuitive ones such as that if $a \wedge b$ and $b \mid c$ then $a \sqsupset c$.

$$\begin{array}{ll}
a \equiv a & (c\ (and\ (not\ d)))\ \#\ f \\
b\ \#\ c & (not\ (c\ (or\ b)))\ \sqsubset\ (not\ c) \\
d\ ^\wedge\ (not\ d) & f\ \#\ ((c\ (or\ (not\ d)))\ (and\ a)) \\
(c\ (and\ d)) \sqsubset d & d \sqsupset ((d\ (or\ d))\ (and\ (not\ b)))
\end{array}$$

Figure 3: Some sample randomly generated pairs of propositional logic statements.

I test the model's ability to learn this behavior by creating an artificial dataset of terms which represent sets of numbers. Since MacCartney and Manning's set of relations hold between sets as well as between sentences, I can use the underlying set structure to generate the all of the relations that hold between any pair of these terms, as in Figure 2. I train the model defined above on a subset of these relations, but rather then presenting the model with a pair of tree-structured sentences as inputs, simply present it 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.

I generate 80 randomly generated sets drawing from the same seven elements, and create a dataset consisting of the relations between every pair of sets, yielding 6400 pairs. 3200 of these pairs were then chosen as a test dataset, and that test dataset was further split into the 2960 examples that can be provably derived from the test data using MacCartney and Manning's join table (or by the symmetry of the relations in about half of the cases) and the 240 that cannot.

A TODO-dimensional RNTN model tuned for the task was able to correctly label TODO % of the derivable test examples, and TODO % of the remaining examples. The simpler RNN model with the same number of dimensions was able to derive only TODO % and TODO %, respectively.

4 Recursive structure

Recursive structure is a prominent feature of natural language. Consider, for example, *Alice said hello*, *Bob said that Alice said hello*, and *Carl thinks that Bob said that Alice said hello*. Overt recursion of this kind is easy to find, and theoretical accounts of natural language syntax and semantics rely heavily on recursive structures. In order for a model to be able to accurately learn natural language meanings, then, we expect that it would need to be able to learn to represent the meanings of function words in a such a way that they are able to behave correctly when taking their own outputs as isput.

We again test this phenomenon within the framework of MacCartney and Manning-style entailment reasoning, but we replace the unanalyzed symbols from the previous experiment with expressions that involve recursive structure. To define these expressions, we turn to propositional logic, a relatively simple logic in which each variable represents either *true* or *false*. We generate data of the form seen in Figure 3: strings of arbitrary length consisting of six elementary proposition symbols and the operators *and*, *or*, and *not*, arranged in pairs with the logical relations between them specified. It should be noted that we preserve the same The data are generated randomly and deduplicated, and consist of about 248k training examples and 44k test examples. Each of the six symbols and each of the three operators is treated as a word for the purposes of our model, and is represented by a randomly initialized vector representation.

Socher et al. [10] have previously demonstrated 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 ouptuts 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 [10], this logic distinguishes between 64 (2^6) possible assignments of truth values, and expressions of this logic define arbitrary conditions on these possible assignments, for a total of 2^{64} ($\approx 10^{20}$) possible statements that the recursive model needs to be able to distinguish.

(some x) mobile $\sqsupset$ (some y) mobile [s.t. $x \sqsupset y$]	**(all x) French $\sqsubset$ (most x) European**
(some dog) mobile $\sqsupset$ (some puppy) mobile	(all puppy) French $\sqsubset$ (most puppy) European
(some animal) mobile $\sqsupset$ (some cat) mobile	(all cat) French $\sqsubset$ (most cat) European
(some Asian) mobile $\sqsupset$ (some Thai) mobile	(all hippo) French $\sqsubset$ (most hippo) European
...	...

Table 2: Sample datasets from two different templates.

...

train up to depth 5 test up to depth 12

5 Reasoning with natural language quantifiers

Interest in NLI in the NLP community is ongoing, and a version of it has been included in the 2014 SemEval challenge specially targeted towards the evaluation of distributional models. Neither this dataset, nor any other existing RTE and NLI datasets are appropriate for the task at hand however. I intend for the present experiment to evaluate the ability of a class of models to learn certain types of inference behavior, and need a dataset that precisely tests these phenomena. With existing datasets that use unrestricted natural language, there is the risk that a model could successfully capture monotonicity inference, but fail to accurately label test data due to problems with, for example, lexical ambiguity, syntactic ambiguity, coreference resolution, or pragmatic enrichment.

5.1 Data

The dataset is built on a small purpose-built vocabulary, which is intended to be large and diverse enough to permit a wide variety of experiments on different semantic phenomena beyond those shown here, but which is still small enough to allow the relations between the lexical items themselves to be exhaustively manually labeled when needed. The vocabulary contains 41 predicates (nouns, adjectives, and intransitive verbs; see Appendix B for more on the design of the wordlist), six quantifiers, the logical conjunctions *and* and *or*, and the unary operator *not*.

The data take the form of 12k pairs of short sentences, each annotated with a relation label, which I divide into about 200 smaller datasets. These datasets contain tightly constrained variants of a specific valid reasoning pattern, each differing in at most one lexical item, and all sharing the same relation label. The remainder of this section describes the composition of these 200 datasets. Examples from four of the datasets are provided in Table 2.

Basic monotonicity: This set of datasets contains reasoning patterns like those seen in example 1 in Table 2, where one of the predicates on one side entails the predicate in the corresponding position on the other side. In some of the datasets (as in 1), this alternation is in the first argument position, in some the second argument position, and in some both. For the alternating first argument subclasses, I have every lexical entailment pair in the data in the first argument position, the terms *bark*, *mobile*, and *European* in the second argument position, and every quantifier. For alternating second argument datasets, I have all predicates in the first argument position, and the pairs *bark–animate*, *French–European*, and *Parisian–French* in the second argument position. The datasets in which there is an alternation in both positions fall into two categories. In some, the entailment relations between the arguments work in the same direction (*some dog French* $\sqsubset$ *some animal European*), in others, they work in opposite directions (*some dog European # some animal French*).

Quantifier substitution: These datasets, exemplified in 2 above, show the interactions between each possible pair of different quantifiers, with the same predicates on both sides. Datasets exist with *bark*, *mobile*, and *European* in the second argument position, and within each dataset every possible predicate is shown in first argument position.

Monotonicity with quantifier substitution: These datasets show the more complex monotonicity interactions that emerge when reasoning between phrases with differing arguments and differing quantifiers, as in 3. Exhaustively creating datasets of this kind involves too large an amount of data to readily verify manually, so I sampled from the extensive set of possible combinations of the six quantifiers and the fixed argument fillers used in the monotonicity datasets. Each possible relation

Target dataset	Data evaluated	SUBCL.-OUT	PAIR-OUT
(most x) bark | *(no x)* bark	target dataset only	100%	93.6%
(two x) bark # (all x) bark	target dataset only	100%	94.7%
(some x) bark ^ (no x) bark	target dataset only	0%	0%

Table 3: Test set accuracy.

but '$\equiv$' (which does not apply sentences like these unless the predicates on both sides are identical) is expressed, as is every possible pair of quantifiers.

Negation: To show the interaction of negation and monotonicity, I included variants of many of the datasets described above in which one of the four argument positions is systematically negated, as demonstrated in example 4.

6 Experiments and results

In the simplest experimental setting, which I label ALL-SPLIT, I randomly sample 85% the data—making sure to sample 85% of each of the individual datasets—train the model on that portion, and evaluate on the remaining 15% of the data. This setting is meant to test whether the model is able to correctly generalize the individual reasoning patterns represented by each of the datasets.

Performance on this setting is perfect: the model quickly converges to 100% accuracy on the test data, showing that it is capable of accurately learning to capture all of the reasoning patterns in the data. The remaining experimental settings serve to determine whether what is learned captures the underlying logical structure of the data in a way that allows it to accurately label unseen kinds of reasoning pattern. In each of them, I choose one of three arbitrarily chosen target datasets, all involving quantifier substitution, to test on. I then then hold out that dataset and—depending on the experimental setting—other similar datasets from the training data in an attempt to discover just how different a test example can be from anything seen in training and still be classified accurately. Table **??** shows what information is included in the training data for each of the four settings for one of the three target datasets.

SET-OUT: In these experiments, I hold out the target dataset, training on none of it and testing on all of it, and additionally split each remaining dataset as in ALL-SPLIT. This setting is meant to test whether the model can generalize a broader reasoning pattern from one dataset to another—the model will still have seen other similar quantifier substitution datasets that differ from the target dataset only in which filler word is placed in the second argument position, as in row 5 in Table **??** above.

PAIR-OUT: In these experiments, I hold out all of the datasets that demonstrate the interaction between a particular pair of quantifiers. For the target dataset *(most x) bark* | *(no x) bark* in this setting, all examples with *most* on one side and *no* on the other, in either order, with or without negation or differing predicates, are held out from training. The remaining datasets are split as above.

These last three experiments have multiple sources of test data: some from the 15% test portions of the datasets that were split, some from the target datasets, and some from any other similar datasets that were held out. In Table 3, I report results for each experiment on the entire test dataset, on the single target dataset, and on all of the held out datasets (i.e., the test data excluding the 15% portions from the training datasets) in aggregate. This third figure is identical to the second for the SET-OUT experiments, since the only held out dataset is the target dataset.

Performance was perfect for at least some held out datasets in both the SET-OUT and SUBCLASS-OUT settings, and near perfect for some in PAIR-OUT. Performance for one of the three target datasets—*(some x) bark ^ (no x) bark*—was consistently poor across training settings. This poor performance is consistent across random initializations, and should be a target of further investigation.

6.1 Discussion

The model learns to generalize well to novel data in most but not all of the training configurations. This inconsistent performance suggests that there is room to improve the optimization techniques used on the model, but the fact that it is able to perform well in these settings even some of the time suggests that the structure of the model is basically capable of learning meaning representations that support inference.

The results in the ALL-SPLIT condition show two important behaviors. The model is able to learn to identify the difference between two unseen sentences and consistently return the label that corresponds to that difference. In addition, the model can learn lexical relations from the training data, such as *dog* $\sqsubset$ *animal* from *(no dog) bark* $\sqsupset$ *(no animal) bark*, and it can then use these learned lexical relations to compute the relation for a sentence pair like *(some dog) bark* $\sqsubset$ *(all animal) bark*. The results from the other three experimental settings show that the model is able to learn general purpose representations for quantifiers which, at least in many cases, allow it to perform inference when a crucial difference between two sentences—the substitution of one specific quantifier for another—has not been seen. These results serve to confirm that the representations learned are capturing some of the underlying logic, rather than than just supporting the memorization of fixed reasoning patterns.

These results leave open the question of how much information is minimally needed to learn general purpose representations for quantifiers in this setting. There are two lines of experimental work that could help to clarify this. Including longer sentences and constructions involving conjunctions (i.e. *and, or*), transitive verbs (i.e. *eats, kicks*) or other constructions in the training and test data could further test what kinds of behavior can be learned from a small training set, as could further experiments on this data involving even smaller training sets than those shown here or differently structured configurations of train and test sets.

6.2 General discussion

1 Para

- capable of a geometric encoding of fundamentals of logic

- capable of capturing recursive structure up to a scale comparable to what's used in language, and of true recursion

- capable of learning functional word meanings given adequate data, frontiers: more complex cases

Acknowledgments

Withheld for review.

Bibliography

[1] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12: 2493–2537, 2011.

[2] 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*. ACL, 2011.

[3] 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*. EMNLP, 2013.

[4] 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.

[5] Marco Baroni, Raffaella Bernardi, and Roberto Zamparelli. Frege in space: A program for compositional distributional semantics. *Submitted, draft at http://clic. cimec. unitn. it/composes*, 2013.

[6] Edward Grefenstette. Towards a formal distributional semantics: Simulating logical calculi with tensors. *arXiv preprint 1304.5823*, 2013.

[7] Tim Rocktäschel, Matko Bosnjak, Sameer Singh, and Sebastian Riedel. Low-dimensional embeddings of logic. *Proceedings of the ACL 2014 Workshop on Semantic Parsing*, 2014.

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

[9] 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.

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

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

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

[13] Dan Garrette, Katrin Erk, and Raymond Mooney. A formal approach to linking logical form and vector-space lexical semantics. *Computing Meaning*, 4, 2013.

[14] Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. *NIPS*, 24, 2011.

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

[16] Christoph Goller and Andreas Kuchler. Learning task-dependent distributed representations by backpropagation through structure. In *Proc. IEEE International Conference on Neural Networks*, volume 1. IEEE, 1996.

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