Inside-Outside Semantics: A Framework for Neural Models of Semantic Composition

Le, P.; Zuidema, W.

Published in:
NIPS 2014 Workshop on Deep Learning and Representation Learning

Inside-Outside Semantics: A Framework for Neural Models of Semantic Composition

Phong Le and **Willem Zuidema**
Institute for Logic, Language, and Computation
University of Amsterdam, the Netherlands
{p.le,zuidema}@uva.nl

Abstract

The Recursive Neural Network (RNN) model and its extensions have been shown to be powerful tools for semantic composition with successes in many natural language processing (NLP) tasks. However, in this paper, we argue that the RNN model is restricted to a subset of NLP tasks where semantic compositionality plays a role. We propose an extension called Inside-Outside Semantics. In our framework every node in a parse tree is associated with a pair of representations, the inner representation for representing the content under the node, and the outer representation for representing its surrounding context. We demonstrate how this allows us to develop neural models for a much broader class of NLP tasks and for supervised as well as unsupervised learning. Our neural-net model, Inside-Outside Recursive Neural Network, performs on par with or better than the state-of-the-art (neural) models in word prediction, phrase-similarity judgements and semantic role labelling.

1 Introduction

For several decades, the most successful approaches for modelling word meaning, on the one hand, and those for modelling semantic composition (needed to compute the meaning of phrases and sentences), on the other hand, seemed incompatible. Word meanings can be well described with numerical vectors, often reflecting co-occurrence frequencies with other words in corpora. A rich and diverse literature has emerge on *distributional semantics*, with many successes in tasks like similarity judgements, word sense disambiguation and information retrieval (see [1] for a review).

Semantic composition, on the other hand, has since Montague [2] been modelled using the lambda-calculus, which is straightforwardly applied to symbolic logics of various sorts, but seems incompatible with vector representations. As recently as 2008, Lenci [1] wrote in a review of distributional semantics that the issue of how to do semantic composition with vectorial word representations remained unsolved. Since then, however, a series of papers have appeared proposing a range of technical solutions for this problem and creating much excitement about the prospects of finally bridging the two fields. One of the most successful of these new approaches, the Recurrent Neural Network (RNN) of Socher et al. [3, 4, 5] seems to lend itself well for neural network-based technology for language processing, and perhaps even to use as a model of how the hierarchical structure of language might be processed in the human brain.

In this paper, we discuss the published RNN models in some detail, and show that, at a closer look, they leave some major challenges unsolved. One such challenge concerns the bottom-up definition of composition, which reduces the RNN's usefulness as a framework for NLP. We propose a solution to this challenge, consisting of an extension, which we call Inside-Outside Semantics. We show that this allows information to flow top-down as well as bottom-up, and thus to successfully apply the model to a wide range of NLP tasks. We report results on a word prediction task, a phrase similarity

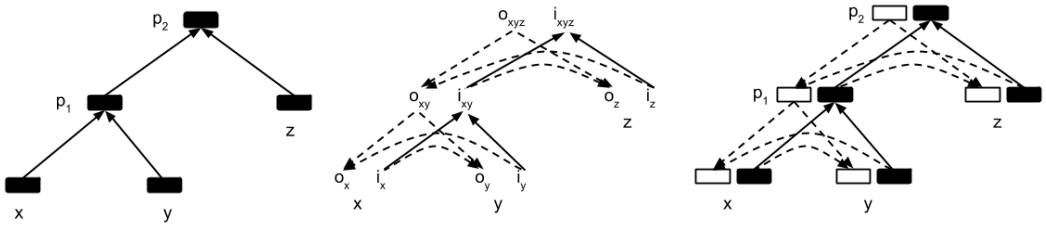

Figure 1: (Left) Recursive Neural Network. (Middle) Inside-Outside framework. (Right) Inside-Outside Recursive Neural Network. Black rectangles correspond to inner representations, white rectangles correspond to outer representations.

judgement task, and a semantic role labelling task, and show in each task that our new model is on par with the best n-gram or neural network approach for that specific domain.

2 Recursive Neural Networks

Words in natural language combine according to the principle of compositionality: *"The meaning of a whole is a function of the meanings of the parts and of the way they are syntactically combined"* [6]. In formal semantics, meaning representations are typically computed in a bottom-up manner: given a constituent xy and a grammar rule R that combines the two linguistic items x and y, then $\mathbf{xy} = f_R(\mathbf{x}, \mathbf{y})$ where f_R is a composition function. There are different ways to instantiate meaning representations and composition functions. In formal semantics, meaning representations are typically λ-expressions and the composition function is function application [7]. In recent models of compositional vector-based semantics, the words meanings are vectors, matrices or tensors, and a variety of algebraic operators (e.g. vector addition) have been proposed for composition.

Socher and colleagues [3] proposed to learn the composition functions from data, rather than handcraft them, and to implement them using feedforward neural networks. The basic idea of their *Recursive Neural Networks* is simple. Assume that there is a parse tree $(p_2\ (p_1\ x\ y)\ z)$ (Figure 1-left), and that $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{R}^n$ are vectorial representations of the three words x, y and z. We can then use a standard feedforward neural network which consists of a weight matrix $\mathbf{W}_1 \in \mathbb{R}^{n \times n}$ for left children and a weight matrix $\mathbf{W}_2 \in \mathbb{R}^{n \times n}$ for right children to compute the vector for a parent node in a bottom up manner. The RNN is *recursive* in the sense that the same weight matrix is used recursively to compute vector representations for nodes higher up in the tree, until we reach the root node. Thus, we compute representations for p_1 $\mathbf{p}_1 = f(\mathbf{W}_1\mathbf{x} + \mathbf{W}_2\mathbf{y} + \mathbf{b})$ and for p_2 $\mathbf{p}_2 = f(\mathbf{W}_1\mathbf{p}_1 + \mathbf{W}_2\mathbf{z} + \mathbf{b})$ where $\mathbf{b}$ is a bias vector and f is an activation function (e.g., *tanh*).

In the classic setup [8], the sequence of computations the RNN performs is guided by an external, symbolic control structure that is provided with a parse tree of the sentence. Socher et al. integrate the RNN's vector computations with the (greedy) search for the best tree structure for a sentence, but still need a symbolic control structure to guide the applications of the neural network. Using Goller & Kuechler's backpropagation through structure algorithm [8], and training on the syntactic annotations of the Penn WSJ treebank, Socher et al. obtain promising results on the syntactic parsing task for short sentences. A straightforward variant of the model, called compositional vector grammars [9], introduces separate weight matrices for grammar rules, and obtains excellent parsing scores on the Penn WSJ benchmark test.

Some extensions of the RNN have been introduced to further improve the accuracy of the composition functions. The matrix-vector model (MV-RNN) [4] associates with every node in the tree not only a length n vector, but also an $n \times n$ matrix. When computing a representation for a composition xy, the model then multiplies the matrix associated with x with the vector associated with y, and the matrix associated with y with the vector associated with x, and combines the two resulting vectors as in the standard RNN. This way, the MV-RNN can exploit the fact that some words/constituents, e.g. adjectives, play roles as modifiers. One weak point of the MV-RNN is that it requires a large number of parameters; this problem is addressed with a further extension called neural tensor networks [10].

Thanks to the successes in NLP tasks such as constituent parsing [9], sentiment analysis [10], RNN models are now firmly established as a powerful tool for semantic (and syntactic) composition.

However, the RNN framework can still only be applied to a specific subset of natural language processing problems that involve compositional structure, because of (i) the need for supervision, i.e. syntactic and semantic annotations needed to compute a training signal, and (ii) the bottom-up definition of semantic composition. A partial solution to the first problem is provided by an elegant extension called Recursive Auto-encoders (RAE) [5], which replace the traditional feedforward neural networks by auto-encoders. The underlying principle of the RAE is that *composition is compression*, such that an input is able to be recovered from the output by a decoding function. In this paper, however, we present an alternative extension of RNN's for unsupervised learning that outperforms the RAE on the popular phrase-similarity judgement task by a large margin.

The second problem, that information is allowed to flow bottom-up only, turns out to be greatly limiting the RNN models' usefulness to a number of important domains in NLP. First, the RNN models are not able to be used in ways that parallel major parsing strategies, such as Earley parsing [11] and left-corner parsing [12], that make use of top-down predictions. Second, the RNN's cannot be used either in ways that parallel important classes of top-down generative probability models such as Eisner's dependency models [13] and Collins' phrase-structure models [14]. Third, the RNN models do not compute representations for the contexts of constituents, even though contexts are essential for many tasks, such as (i) next or missing word prediction based on preceding or surrounding context, or (ii) semantic role labelling, where the goal is to assign semantic roles to constituents based on properties of both the slot (context) and the filler (content).

3 The Inside-Outside Semantics Framework

Believing that a natural solution for the difficulty mentioned above is to be found in the way information 'flows' in linguistic structures such as parse trees, we propose a framework, *Inside-Outside Semantics*. The crucial innovation of our framework is to introduce an architecture that allows information to flow in parse trees not only bottom-up, as in traditional compositional semantics frameworks, but also top-down. We accomplish this by adding a second vector to every node that represents the *context* surrounding that node (rather than the *content* under that node). These context representations can, like content representations, be computed compositionally. But where content representations are computed bottom-up from a node's children, context representations are computed top-down from a node's parent and siblings. The interaction between content and context representations in turn makes it possible to devise a new unsupervised learning scheme for estimating composition functions from raw texts. The interaction, moreover, can be formed based on training signals in supervised learning tasks such as semantic role labelling, named entity recognition.

We denote with $\mathbf{i}_{xy}$ and $\mathbf{o}_{xy}$ the representations of xy's content and context. Henceforth, we will call $\mathbf{i}$ the *inner representation*, and $\mathbf{o}$ the *outer representation*. Similarly to the traditional semantic composition framework, we have $\mathbf{i}_{xy} = f_R^i(\mathbf{i}_x, \mathbf{i}_y)$ where f_R^i is a composition function for inner representation. To compute outer representations, we use the following formulas

$$\mathbf{o}_x = f_R^{o1}(\mathbf{o}_{xy}, \mathbf{i}_y) \; ; \; \mathbf{o}_y = f_R^{o2}(\mathbf{o}_{xy}, \mathbf{i}_x) \tag{1}$$

where f_R^{oj} is the composition function for the outer representation of the j-th child. The rationale for these equations is that the context of the constituent x consists of the context of the constituent xy and the constituent y; the outer representation of x is thus a function of the outer representation of xy and the inner representation of y.

The whole process is depicted graphically in Figure 1-middle, where an arrow indicates that the representation at its head is a function of the representation at its tail, and solid arrows are for computing inner representations whereas dotted arrows are for outer representations. Looking at the graph, we can see that there are two information flows in the tree structure: one from bottom up and the other from top down. The interaction between these two flows may occur at any node and depends on the purpose of usage.

Our framework shares some properties with Irsoy and Cardie's recently proposed *bidirectional* RNN model [15], which also adds a second representation, the 'down vector', to every node and allows for information to flow top-down. However, because these down vectors can only be computed after

all 'up vectors' (corresponding to our inner representations) for a tree have been computed first, we believe that the model is not able to solve any of the three problems mentioned in Section 2. In our framework, in contrast, inner and outer representations are strictly complementary: the outer representation for a node x can be computed without knowing the contents under that node.

There are different ways to instantiate representations $\mathbf{i}, \mathbf{o}$ and composition functions f_R^i, f_R^{oj}. In the next section, we describe our IORNN [16], a neural-network-based instance of the framework. [1]

4 The Inside-Outside Recursive Neural Network

We explain the operation of the IORNN in terms of the same example parse tree $(p_2\ (p_1\ x\ y)\ z)$ as we used before (see Figure 1-right). For the present paper we assume that an independently given parser assigns a syntactic structure to an input string. Each node u in the syntactic tree carries two vectors $\mathbf{o}_u$ and $\mathbf{i}_u$, the outer representation and inner representation of the constituent that is covered by the node.

Inner representations are computed from the bottom up. We assume for every word w an inner representation $\mathbf{i}_w \in \mathbb{R}^n$. The inner representation of a non-terminal node, say p_1, is given by $\mathbf{i}_{p_1} = f(\mathbf{W}_1^i \mathbf{i}_x + \mathbf{W}_2^i \mathbf{i}_y + \mathbf{b}^i)$ where $\mathbf{W}_1^i, \mathbf{W}_2^i$ are $n \times n$ real matrices, $\mathbf{b}^i$ is a bias vector, and f is an activation function, e.g. $tanh$. The inner representation of a parent node is thus a function of the inner representations of its children.

Outer representations are computed from the top down. For a node which is not the root, say p_1, the outer representation is given by $\mathbf{o}_{p_1} = g(\mathbf{W}_1^o \mathbf{o}_{p_2} + \mathbf{W}_2^o \mathbf{i}_z + \mathbf{b}^o)$ where $\mathbf{W}_1^o, \mathbf{W}_2^o$ are $n \times n$ real matrices, $\mathbf{b}^o$ is a bias vector, and g is an activation function. The outer representation of a node (i.e., the associative profile of its context) is a function of the outer representation of its parent and the inner representations of its sisters. If there is information about the external context of the utterance that is being processed, this information determines the outer representation of the root node $\mathbf{o}_{root}$. In our first experiments reported here, no such information was assumed to be available. In this case, a random value $\mathbf{o}_\emptyset$ is chosen and assigned to the root nodes of all utterances. Note that this value is adjusted by the learning process discussed below.

Training the IORNN is to minimise an objective function $J(\theta)$ which depends on the purpose of usage (see Section 5, 6 and [16]) where θ is the set of parameters. To do so, we compute the gradient $\partial J / \partial \theta$ and apply the gradient descent method. The gradient is effectively computed thanks to back-propagation through structure [8]. We use AdaGrad [17] to update the parameters.

Incorporating Grammatical Information The IORNN above uses just four matrices (left/right for inside, parent/sister for outside) for combining any two nodes. However, Socher et al. [9] and Hermann & Blunsom [18] find that using different matrices for different grammar rules/categories greatly improves results. For most of our experiments, we follow this suggestion, using one of the following types of syntactic representations: [CFG-k], context-free grammar (CFG) derivation with the k most frequent rules (according to the Penn Treebank section 2-21), the other rules are replaced by abstract rules $X \rightarrow X\ ...\ X$; and [CCG], combinatory categorial grammar (CCG) derivation with all combinatory rules.

CCG [19] is a widely used grammar formalism for semantic applications thanks to its transparent interface between syntax and semantics [7]. In CCG, a word or constituent is assigned to a category, which could be basic (e.g., NP, S) or complex (e.g., NP/NP, $S\backslash NP$). A complex category defines selectional and directional combination. For instance, NP/NP implies that any constituent or word in this category can combine with a noun phrase to it right-hand side in order to form a new noun phrase. Standard CCG specifies five basic rules for combination: forward application (fa), backward application (ba), compo-

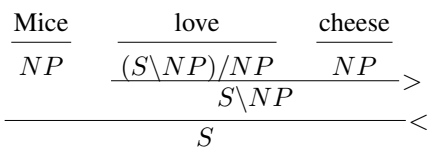

Figure 2: A CCG derivation for the sentence "Mice love cheese".

[1] See Appendix A for using λ-expressions, as in formal semantics, within our framework.

sition (comp), coordination (conj), and type raising (tr). Figure 2 shows a CCG derivation for the sentence "Mice love cheese".

5 Unsupervised Learning with the IORNN: word prediction and phrase similarity

In this section, we use the IORNN to learn from data that contain no annotations for the semantics of compounds. In that sense, the learning is 'unsupervised', but note that we will make use of syntactic structure, both at train time and at test time (using a third party parser).

To demonstrate that the capabilities of the IORNN beyond those of a standard bottom-up RNN, we apply it to the task of predicting missing words in a sentence. We reason that any fluent speaker of a language can guess the meaning of an unknown word by making use of the meaning of its context, and that when s/he correctly predicts the meaning of the unknown word, we can thus assume that s/he comprehends the meaning of the context. Following this reasoning, we base the composition function learning task on the word prediction task, which opens up a novel (and much needed) scheme for unsupervised training of semantic composition models. We therefore evaluate the IORNN both on the quality of the missing word predictions and on the quality of the semantic composition functions that it learned for the task (by measuring the correlation of semantic similarity judgements of compounds to those of human subjects).

Using a method proposed by [20], we train the network such that it assigns the correct target word a score higher than the scores given to other words. The score $s(x, \mathbf{o}_w)$ given to a candidate word x for a specific context $\mathbf{o}_w$ is computed by

$$u(x, \mathbf{o}_w) = f(\mathbf{W}_1^u \mathbf{o}_w + \mathbf{W}_2^u \mathbf{i}_x + \mathbf{b}^u) \; ; s(x, \mathbf{o}_w) = \mathbf{W}^s u(x, \mathbf{o}_w) \tag{2}$$

where $\mathbf{W}_1^u, \mathbf{W}_2^u$ are $n \times k$ real matrices, $\mathbf{W}^s$ is a $k \times 1$ matrix, and $\mathbf{b}^u$ is a bias vector. (We fix $k = 2n$.) Now, the objective function is the ranking criterion with respect to θ:

$$J(\theta) = \sum_{s \in D} \sum_{w \in s} \sum_{x \in V} \max\{0, 1 - s(w, \mathbf{o}_w) + s(x, \mathbf{o}_w)\} \tag{3}$$

where the first sum is over all sentences s in a corpus D, the second sum is over all observed words w in a sentence, and the third sum is over all candidate words x in the vocabulary V. The final term will give a value higher than 1 if the score of the candidate is higher than that of the observed word, and smaller than 1 (but at least 0) if the observed word's score is higher. To minimize this objective function, we randomly pick words in the vocabulary as 'corrupted' examples and follow the method given in Section 4.

5.1 Experiments

We built three IORNNs for three types of derivation [CFG-0], [CFG-200], and [CCG] (see Section 4). All of them were initialised with the 50-dim word embeddings[2] from [20] and trained on around 3M sentences longer than 10 words[3] in the British National Corpus (BNC), parsed by the C&C parser [21] and by the Berkeley Parser [22].

Table 1: Results in the word prediction task. (Smaller is better.)

	NAR
5-gram	1.8%
IORNN [CFG-0]	1.6%
IORNN [CFG-200]	**0.8%**

Word Prediction We run the trained networks on the Penn Treebank, section 22, and compare their performances with a baseline formed by the Kneser-Ney smoothed 5-gram model[4] trained on 300M-word text from the Wikipedia[5] (3 times larger than the BNC corpus). It is worth noting that this model is state-of-the-art in the language modelling area, outperforms syntax-based language models and thus is a strong baseline to compare

[2] `http://ronan.collobert.com/senna/`

[3] Long contexts are needed for successfully guessing missing words.

[4] We use the BerkeleyLM at `http://code.google.com/p/berkeleylm/`.

[5] `http://nlp.stanford.edu/data/WestburyLab.wikicorp.201004.txt.bz2`

Table 2: Items in the dataset from [23]. The ratings range from 1 (very low similarity) to 7 (very high similarity).

type	phrase 1	phrase 2	rating
v-obj	remember name	pass time	3
adj-n	dark eye	left arm	5
n-n	county council	town hall	4

Table 3: Spearman's correlation coefficients of model predictions for the phrase similarity task.

model	adj-n	n-n	v-obj
RAE	0.20	0.18	0.14
IORNN [CFG-0]	**0.38**	**0.36**	**0.32**
CCAE-B	0.38	**0.44**	0.34
IORNN [CCG]	**0.39**	0.40	**0.37**

against (even if it, unlike the IORNN's, does not make use of the syntactic structure of sentences at either train or test time). Predicting the actual word at any location in the sentence is often next to impossible. We therefore follow standard practice in evaluating these models, and base the evaluation on the rank of the correct word in the list of candidate words (all 130k words in the vocabulary) ranked by each of the models (Equation 2).

Table 1 shows normalized averaged ranks (NAR, i.e., averaged-rank$/|V|$, where $|V|$ is the size of the vocabulary) of target/gold-standard words. Both IORNNs score considerably lower NARs than the 5-gram model, showing that the IORNN is capable of judging very well how different words fit the context created by a surrounding utterance.

Phrase Similarity To evaluate the composition functions that are learned in the word prediction task, we make use of the phrase similarity judgement task. In this task the system considers pairs of (short) phrases and estimates the semantic similarity within each pair. Its estimations are compared with human judgements. We used the dataset from [23] which contains 5832 human judgements on 108 phrase pairs.[6] The pairs include noun-noun, verb-object, and adjective-noun combinations; the ratings range from 1 (very low similarity) to 7 (very high similarity) (see Table 2). The cosine distance was used to measure semantic similarity $d(a, b) = \cos(\mathbf{i}_a, \mathbf{i}_b)$ and the Spearman's correlation coefficient $\rho \in [-1, 1]$ was used to measure the difference between model scores and human judgements.

We compared our IORNNs against two models: the Recursive Auto-encoder (RAE) replicated by [24], and the Combinatory Categorial Autoencoders (CCAE)-B model proposed by [18]. The CCAE-B model is similar to the RAE but uses separate weight matrices for each CCG rule. Table 3 shows results. The IORNN [CFG-0] (which, like the basic RAE, does not make use of syntactic category information) outperforms the RAE by a large margin. When syntactic category information is used, in the IORNN [CCG] and CCAE-B, we can observe improvements across the board, with the IORNN [CCG] outperforming all other models on verb-obj and adj-noun compounds.

6 Supervised Learning with the IORNN: Semantic Role Labelling

Differing from the previous section, in this section we focus on a supervised learning task, namely semantic role labelling, where we learn from data that does contain semantic annotations. The interaction between the outer and inner representations at a node is now determined by the semantic role assigned to the constituent the node covers.

Our benchmark is the CoNLL2005 shared task [25]. In this task, given a sentence and a verb in this sentence, the system is required to classify every constituent into one of 54 classes, corresponding to 53 roles or class '-NONE-' (which means the constituent is not an argument). Following the PropBank formalism, roles A0-5 are assigned to words that are arguments of verbs (e.g., A0 for agent, experiencer; A1 for patient, theme; and so on.) In addition, there are several modifier tags such as AM-LOC (locational) and AM-TMP (temporal). The given dataset consists of a train set (sections 2-21 of the Penn WSJ treebank), a development set (section 24), and a test set (section 23 plus three sections from the Brown corpus). All the sentences were parsed by the Charniak parser [26]. The task is evaluated by computing the F1 measure.

[6] `http://homepages.inf.ed.ac.uk/s0453356/share`

The new IORNN is very similar to the one proposed in Section 4 except two changes. First, we add to the inner representation of any word one more bit, indicating whether this word is the target verb or not. Second, the inner representation of the head word of a constituent is used to compute the constituent's inner representation. At the same time, the inner representation of the target verb is used to compute the outer representation of every constituent[7]. Therefore, we compute inner and outer representations as follows

$$\mathbf{i}_{p_1} = f(\mathbf{W}_1^i \mathbf{i}_x + \mathbf{W}_2^i \mathbf{i}_y + \mathbf{W}_h \mathbf{i}_h + \mathbf{b}^i) \; ; \mathbf{o}_{p_1} = g(\mathbf{W}_1^o \mathbf{o}_{p_2} + \mathbf{W}_2^o \mathbf{i}_z + \mathbf{W}_v \mathbf{i}_v + \mathbf{b}^o) \quad (4)$$

where $\mathbf{W}_h, \mathbf{W}_v$ are two $n \times n$ real matrices, $\mathbf{i}_h, \mathbf{i}_v$ are the inner representations of the head word of the constituent p_1 and of the target verb, respectively. Finally, we use the *softmax* function to compute the probability that a constituent p is assigned to a class c

$$P(c|p) = \frac{e^{u(p,c)}}{\sum_{c' \in C} e^{u(p,c')}} \text{ where } [u(p,c_1), ..., u(p,c_{|C|})]^T = \mathbf{W}^o \mathbf{o}_p + \mathbf{W}^i \mathbf{i}_p + \mathbf{b} \quad (5)$$

C is the set of all classes, $\mathbf{W}^o, \mathbf{W}^i$ are $|C| \times n$ real matrices, $\mathbf{b}$ is a $|C|$-dimensional vector. We use the method presented in Section 4 to train the network, where the objective function is the cross-entropy over all constituents, i.e., $J(\theta) = -\sum_{T \in D} \sum_{p \in T} \sum_{i=1}^{|C|} t_i \times log(P(c_i|p))$ where $t_i = 1$ if c_i is the correct class of p, $t_i = 0$ otherwise; and D is the set of training parse trees.

IORNN-mixture Combining neural networks is a simple way to increase the general performance [27]. To combine m IORNNs, we simply average the probabilities that the IORNNs assign to a class c by $P(c|p) = \frac{1}{m} \sum_{i=1}^{m} P_i(c|p)$ where P_i is the probability given by the i-th IORNN.

6.1 Experiment: CoNLL 2005 Shared Task

We built an IORNN [CFG-200], initialised with the Collobert & Weston word-embeddings [20]. Because the number of constituents in the class '-NONE-' was significantly larger than the numbers of constituents in other classes, we used the simple pruning algorithm proposed by [28] to eliminate those constituents that are very unlikely arguments. We also built an IORNN-mixture which is a combination of ten IORNNs that have the same setting.

Table 4: Performance on the SRL task of IORNN compared with SENNA. IORNN+Charniak.RS is the IORNN evaluated on the sentences parsed by the self-trained re-ranked Charniak parser.

Approach	F1-valid	F1-test
SENNA (w.o. syntax)	72.29	74.15
SENNA	–	75.49
IORNN	74.04	73.88
IORNN-mixture	–	75.36
IORNN+Charniak.RS	75.78	75.14
IORNN-mixture+Charniak.RS	–	**77.14**

In this experiment, we compared the IORNN with SENNA [20], a state-of-the-art neural net system for POS tagging, chunking, named entity recognising and semantic role labelling. Differing from our IORNN, SENNA's neural net uses a convolution layer to compose representations for the words in a sentence given the target verb position. In addition, SENNA is biased to produce valid tag sequences, by a mechanism that combines network scores with transition scores. Finally, SENNA may also make use of syntactic constraints (computed by its own network). In contrast, in our IORNN, semantic tags are predicted solely on the basis of the computed inner and outer representations for a given node.

Table 4 reports results. We can see that SENNA without using syntax performed comparably with IORNN; when syntactic information was included, SENNA outperformed the (single) IORNN. We conjecture that this is because SENNA is relatively insensitive to errors introduced by automatic parsers, thanks to its convolution layer and transition matrix. Our IORNN, on the other hand, is very sensitive to such errors because it relies on the principle of compositionality. The performance of our system, however, was boosted (by 1.48%) by combining ten IORNNs, yielding comparable performance as SENNA. Furthermore, by using a better parser (we used the state-of-the-art self-trained and discriminatively re-ranked Charniak-Johnson parser; [29, 30]) we increased performance even further (1.26% for IORNN and 1.78% for IORNN-mixture). This confirms the importance of accurate syntactic parsing in semantic role labelling [31].

[7] Making use of head words and target verbs is consistent with the head-driven parsing theory of [14].

7 Top-down prediction with the IORNN: Dependency parsing

In this section we briefly consider the use of the IORNN in top-down prediction tasks, such as needed for Earley-style parsing and left-corner parsing, but also for defining neural models to replace top-down generative probability models such as those of Collins [14] (generating phrase-structure trees) and Eisner [13] (generating dependency structures). We limit ourselves here to Eisner's generative model, but similar observations hold for the other applications.

Eisner's generative model is simple. Every sentence is assumed to have a single head word, generated from ROOT. This head word then first generates 0 or more dependents to its left, and then generates 0 or more dependents to its right. Each of the dependents recursively generates left and right dependents as well, until no more words are generated. Eisner considered a number of variants of this generative story, and defined ways to estimate the probabilities of each step in generating a full dependency structure from counts in a training corpus, smoothed using back-off. At each of these steps we have a partial dependency tree (the "conditioning context", representing the results of the generation process up to this point), and a representation of the dependent we are about to generate (the "event").

Can we define a neural network model to estimate these probabilities, to replace the counting and back-off? This requires a neural representation of the conditioning context, and a neural representation of the event. In a standard RNN the latter is available (the vector representation of the word or subtree), but the former is not. In the IORNN, on the other hand, we can use the outer representation as a representation of the conditioning context, and train a softmax classifier to compute the required conditional probabilities. In [16] we describe the technical details of such a model, the approximations needed to make it efficient, and state-of-the-art results in dependency parsing using the IORNN as a reranker on the output of the third-party MST-parser [32]. For the purposes of this paper, where we discuss the range of applications that the Inside-Outside Semantics opens up for neural models, it suffices to note that the IORNN can successfully implement top-down prediction.

8 Conclusion

We have proposed the Inside-Outside Semantics framework in which every node in a parse tree is associated with a pair of representations, the inner representation for representing the contents under the node, and the outer representation for representing its surrounding context. Thanks to adding the context representations, information can flow not only bottom-up but also top-down in parse trees.

The Inside-Outside Recursive Neural Network was presented as an instance of the framework, where both representations are instantiated as high-dimensional vectors and composition functions are feedforward networks trained on large datasets. By exploiting the interaction between context and content representations, we have shown that the IORNN is capable of performing a wide range of NLP tasks that the RNN and its extensions are not.

We demonstrated how the IORNN can be used in the word prediction, learning composition functions from raw texts, and semantic role labelling tasks. Our IORNN outperformed the Kneser-Ney smoothed 5-gram model, which is state-of-the-art in the language modelling area, in the word prediction task. It also achieved better results than the RAE by a large margin in the phrase similarity evaluation. In the semantic role labelling task, our IORNN performed comparably with the state-of-the-art neural-net-based system for NLP, named SENNA. This empirical results establish that Inside-Outside Semantics offers a promising framework for applying RNN to NLP tasks that require paying attention to the hierarchical structure of sentences and *predictions* based on context.

References

[1] Alessandro Lenci. Distributional semantics in linguistic and cognitive research. *Italian journal of linguistics*, 20(1), 2008.

[2] Richard Montague. English as a formal language. *Linguaggi nella societae nella tecnica*, 1970.

[3] Richard Socher, Christopher D. Manning, and Andrew Y. Ng. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*, 2010.

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

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

[6] Barbara Partee. Lexical semantics and compositionality. In Lila R. Gleitman and Mark Liberman, editors, *An Invitation to Cognitive Science. Vol.1: Language*. MIT Press, Cambridge, MA, 1995.

[7] Johan Bos, Stephen Clark, Mark Steedman, James R Curran, and Julia Hockenmaier. Wide-coverage semantic representations from a ccg parser. In *COLING*, 2004.

[8] Christoph Goller and Andreas Küchler. Learning task-dependent distributed representations by backpropagation through structure. In *International Conference on Neural Networks*. IEEE, 1996.

[9] Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. Parsing with compositional vector grammars. In *ACL*, 2013.

[10] 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 *EMNLP*, 2013.

[11] Andreas Stolcke. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational linguistics*, 21(2), 1995.

[12] Christopher D Manning and Bob Carpenter. Probabilistic parsing using left corner language models. In *Advances in probabilistic and other parsing technologies*. Springer, 2000.

[13] Jason M Eisner. Three new probabilistic models for dependency parsing: An exploration. In *COLING*. Association for Computational Linguistics, 1996.

[14] M. Collins. Three generative, lexicalised models for statistical parsing. In *EACL*, 1997.

[15] Ozan Irsoy and Claire Cardie. Bidirectional recursive neural networks for token-level labeling with structure. In *NIPS Workshop on Deep Learning*, 2013.

[16] Phong Le and Willem Zuidema. The inside-outside recursive neural network model for dependency parsing. In *EMNLP*, 2014 (to appear).

[17] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 2011.

[18] Karl Moritz Hermann and Phil Blunsom. The role of syntax in vector space models of compositional semantics. In *ACL*, 2013.

[19] Mark Steedman. *The syntactic process*, volume 35. MIT Press, 2000.

[20] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *JMLR*, 12, 2011.

[21] J.R. Curran, S. Clark, and J. Bos. Linguistically motivated large-scale NLP with C&C and Boxer. In *ACL, Interactive Poster and Demonstration Sessions*, 2007.

[22] Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. Learning accurate, compact, and interpretable tree annotation. In *COLING-ACL*. Association for Computational Linguistics, 2006.

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

[24] William Blacoe and Mirella Lapata. A comparison of vector-based representations for semantic composition. In *EMNLP-CoNLL*, volume 70, 2012.

[25] Xavier Carreras and Lluís Màrquez. Introduction to the conll-2005 shared task: Semantic role labeling. In *CoNLL*. Association for Computational Linguistics, 2005.

[26] Eugene Charniak. Immediate-head parsing for language models. In *ACL*, 2001.

[27] Tomas Mikolov, Anoop Deoras, Stefan Kombrink, Lukas Burget, and Jan Cernockỳ. Empirical evaluation and combination of advanced language modeling techniques. In *INTERSPEECH*, 2011.

[28] Nianwen Xue and Martha Palmer. Calibrating features for semantic role labeling. In *EMNLP*, 2004.

[29] Eugene Charniak and Mark Johnson. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *ACL*, 2005.

[30] David McClosky, Eugene Charniak, and Mark Johnson. Effective self-training for parsing. In *HLT-NAACL*, 2006.

[31] Vasin Punyakanok, Dan Roth, and Wen tau Yih. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287, 2008.

[32] Ryan McDonald, Kevin Lerman, and Fernando Pereira. Multilingual dependency analysis with a two-stage discriminative parser. In *CoNLL*. Association for Computational Linguistics, 2006.

[33] Marco Baroni, Raffaella Bernardi, and Roberto Zamparelli. Frege in space: A program for compositional distributional semantics. In A. Zaenen, B. Webber, and M. Palmer, editors, *Linguistic Issues in Language Technologies*. CSLI Publications, Stanford, CA, 2013.

A Inside-Outside Semantics Framework with λ-Expressions

We will demonstrate how the IOS framework works with λ-expressions as in formal semantics. First of all, we rely on the fact that the content and context of a constituent must be strongly correlated in the sense that we can predict the content thanks to the context. For instance, given the world model $M = \{$Tom loves cheese, mice love cheese, mice hate cats$\}$, and the context "mice love X", we can infer that $X \in \{cheese\}$. Therefore, we hypothesise that, given a world model M, $M \models \mathbf{o}_p\mathbf{i}_p$ must hold for any constituent p, for instance $\mathbf{i}_{cheese} = cheese$ and $\mathbf{o}_{cheese} = \lambda y.love(mice, y)$.

Combinatory Categorial Grammar (CCG) [19] is a widely used grammar formalism for semantic applications thanks to its transparent surface between syntax and semantics [7]. In our terminology, it provides an elegant method to compute *inner representations*. However, it does not specify the operations needed to compute outer representations. To do so, we will introduce extensions for its combinatory rules.

Combinatorial Categorial Grammar In a CCG, a word or constituent is assigned to a category, which could be basic (e.g., NP, S) or complex (e.g., NP/NP, $S\backslash NP$). A complex category defines selectional and directional combination. For instance, NP/NP implies that any constituent or word in this category can combine with a noun phrase to it right-hand side in order to form a new noun phrase. Standard CCG specifies five basic rules for combination: forward application (fa), backward application (ba), composition (comp), coordination (conj), and type raising (tr) (see the first and second columns of Table 5). Figure 3 shows a CCG derivation for the sentence "mice love cheese".

Table 5: The five basic CCG combinatory rules (the first and second columns) and their extensions for computing outer representations (the third column). p, l, r, c respectively denote parent, left-child, right-child, and child.

Rule	Inner (parent)	Outer (children)
(fa) $X/Y \quad Y \Rightarrow X$	$\mathbf{i}_p = \mathbf{i}_l\mathbf{i}_r$	$\mathbf{o}_l = \lambda P.\mathbf{o}_p\,(P\mathbf{i}_r)$ $\mathbf{o}_r = \lambda P.\mathbf{o}_p\,(\mathbf{i}_l P)$
(ba) $Y \quad X\backslash Y \Rightarrow X$	$\mathbf{i}_p = \mathbf{i}_r\mathbf{i}_l$	$\mathbf{o}_l = \lambda P.\mathbf{o}_p\,(\mathbf{i}_r P)$ $\mathbf{o}_r = \lambda P.\mathbf{o}_p\,(P\mathbf{i}_l)$
(comp) $X/Y \quad Y/Z \Rightarrow X/Z$	$\mathbf{i}_p = \lambda x.(\mathbf{i}_l(\mathbf{i}_r x))$	$\mathbf{o}_l = \lambda P.\mathbf{o}_p\,(\lambda x.(P(\mathbf{i}_r x)))$ $\mathbf{o}_r = \lambda P.\mathbf{o}_p\,(\lambda x.(\mathbf{i}_l(Px)))$
(conj) $X \quad conj \quad X' \Rightarrow X''$	$\mathbf{i}_p = \lambda x.(\mathbf{i}_l x \wedge \mathbf{i}_r x)$	$\mathbf{o}_l = \lambda P.\mathbf{o}_p\,(\lambda x.(Px \wedge \mathbf{i}_r x)))$ $\mathbf{o}_r = \lambda P.\mathbf{o}_p\,(\lambda x.(\mathbf{i}_l x \wedge Px)))$
(tr) $A \Rightarrow X/(X\backslash A)$	$\mathbf{i}_p = \lambda R.(R\mathbf{i}_c)$	$\mathbf{o}_c = \lambda P.\mathbf{o}_p\,(\lambda R.(RP))$

Extended rules From the hypothesis that $M \models \mathbf{o}_p\mathbf{i}_p$ must hold for any constituent p, we postulate a more strict constraint: $\mathbf{o}_p\mathbf{i}_p = \mathbf{o}_q\mathbf{i}_q$ for all constituents p, q, which is equivalent to the constraint

$$\mathbf{o}_c\mathbf{i}_c = \mathbf{o}_p\mathbf{i}_p \;\; \forall c \in \mathcal{C}(p), \text{where } \mathcal{C}(p) \text{ is the set of children of } p \tag{6}$$

Therefore, we add formulas in the third column of Table 5 to the CCG rules. To prove that these formulas satisfy the constraint in Equation 6, for an arbitrary child node, simply applying the outer representation to the inner representation and then using the beta reduction, we will get $\mathbf{o}_p\mathbf{i}_p$.

In order to see how those formulas work, let's consider a simple example of the sentence "Mice love cheese" (Figure 3). First of all, because this sentence stands alone, $\lambda P.P$ is assigned to the outer representation of the whole sentence, which means that the truth of this sentence solely depends on its own meaning. Then, using the formulas given in Table 5 we compute the outer and inner representations for each constituent.

The first thing we can see is that, given the world model M above, all (inner,outer) pairs satisfy the requirement $M \models \mathbf{oi}$. Secondly, it turns out that those outer representations are intuitive. For instance, the outer representation of the word "cheese" $\mathbf{o}_c = \lambda x.love(mice, x)$ perfectly intuitively matches the context "mice love –": the *missing* word/constituent must represent an object x that makes the predicate $love(cheese, x)$ true.

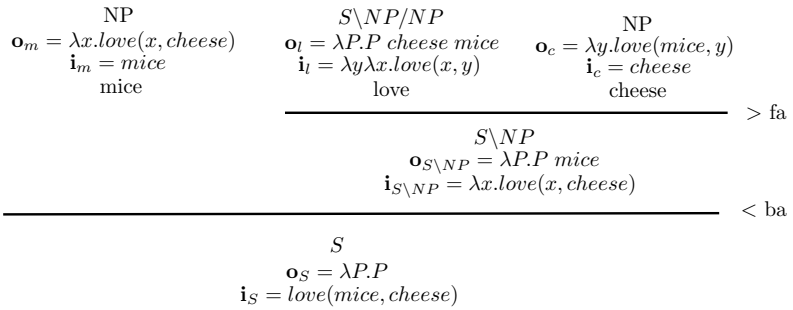

Figure 3: CCG parse tree and inner, outer representations for the sentence *"Mice love cheese"*.

One application for this might be to represent questions. Since a question can be seen as the context of an unknown object/information, we can use the IOS framework to represent a question by converting it to a non-question sentence containing a blank corresponding to the unknown constituent. The outer representation of the unknown constituent therefore can be used as the representation for the question. Another application is to be a base for formal-semantics-liked compositional distributional semantics approaches, e.g. [33], to compute context representations.