The Role of Syntax in Vector Space Models of Compositional Semantics

Karl Moritz Hermann and **Phil Blunsom**
Department of Computer Science
University of Oxford
Oxford, OX1 3QD, UK
{karl.moritz.hermann,phil.blunsom}@cs.ox.ac.uk

Abstract

Modelling the compositional process by which the meaning of an utterance arises from the meaning of its parts is a fundamental task of Natural Language Processing. In this paper we draw upon recent advances in the learning of vector space representations of sentential semantics and the transparent interface between syntax and semantics provided by Combinatory Categorial Grammar to introduce Combinatory Categorial Autoencoders. This model leverages the CCG combinatory operators to guide a non-linear transformation of meaning within a sentence. We use this model to learn high dimensional embeddings for sentences and evaluate them in a range of tasks, demonstrating that the incorporation of syntax allows a concise model to learn representations that are both effective and general.

1 Introduction

Since Frege stated his 'Principle of Semantic Compositionality' in 1892 researchers have pondered both how the meaning of a complex expression is determined by the meanings of its parts, and how those parts are combined. (Frege, 1892; Pelletier, 1994). Over a hundred years on the choice of representational unit for this process of compositional semantics, and how these units combine, remain open questions.

Frege's principle may be debatable from a linguistic and philosophical standpoint, but it has provided a basis for a range of formal approaches to semantics which attempt to capture meaning in logical models. The Montague grammar (Montague, 1970) is a prime example for this, building a model of composition based on lambda-calculus and formal logic. More recent work in this field includes the Combinatory Categorial Grammar (CCG), which also places increased emphasis on syntactic coverage (Szabolcsi, 1989).

Recently those searching for the right representation for compositional semantics have drawn inspiration from the success of distributional models of lexical semantics. This approach represents single words as distributional vectors, implying that a word's meaning is a function of the environment it appears in, be that its syntactic role or co-occurrences with other words (Pereira et al., 1993; Schütze, 1998). While distributional semantics is easily applied to single words, sparsity implies that attempts to directly extract distributional representations for larger expressions are doomed to fail. Only in the past few years has it been attempted to extend these representations to semantic composition. Most approaches here use the idea of vector-matrix composition to learn larger representations from single-word encodings (Baroni and Zamparelli, 2010; Grefenstette and Sadrzadeh, 2011; Socher et al., 2012b). While these models have proved very promising for compositional semantics, they make minimal use of linguistic information beyond the word level.

In this paper we bridge the gap between recent advances in machine learning and more traditional approaches within computational linguistics. We achieve this goal by employing the CCG formalism to consider compositional structures at any point in a parse tree. CCG is attractive both for its transparent interface between syntax and semantics, and a small but powerful set of combinatory operators with which we can parametrise our non-linear transformations of compositional meaning.

We present a novel class of recursive models, the Combinatory Categorial Autoencoders (CCAE), which marry a semantic process provided by a recursive autoencoder with the syntactic representations of the CCG formalism. Through this model we seek to answer two ques-

Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, pages 894–904,
Sofia, Bulgaria, August 4-9 2013.

tions: Can recursive vector space models be reconciled with a more formal notion of compositionality; and is there a role for syntax in guiding semantics in these types of models? CCAEs make use of CCG combinators and types by conditioning each composition function on its equivalent step in a CCG proof. In terms of learning complexity and space requirements, our models strike a balance between simpler greedy approaches (Socher et al., 2011b) and the larger recursive vector-matrix models (Socher et al., 2012b).

We show that this combination of state of the art machine learning and an advanced linguistic formalism translates into concise models with competitive performance on a variety of tasks. In both sentiment and compound similarity experiments we show that our CCAE models match or better comparable recursive autoencoder models.[1]

2 Background

There exist a number of formal approaches to language that provide mechanisms for compositionality. Generative Grammars (Jackendoff, 1972) treat semantics, and thus compositionality, essentially as an extension of syntax, with the generative (syntactic) process yielding a structure that can be interpreted semantically. By contrast Montague grammar achieves greater separation between the semantic and the syntactic by using lambda calculus to express meaning. However, this greater separation between surface form and meaning comes at a price in the form of reduced computability. While this is beyond the scope of this paper, see e.g. Kracht (2008) for a detailed analysis of compositionality in these formalisms.

2.1 Combinatory Categorial Grammar

In this paper we focus on CCG, a linguistically expressive yet computationally efficient grammar formalism. It uses a constituency-based structure with complex syntactic types (categories) from which sentences can be deduced using a small number of combinators. CCG relies on combinatory logic (as opposed to lambda calculus) to build its expressions. For a detailed introduction and analysis vis-à-vis other grammar formalisms see e.g. Steedman and Baldridge (2011).

CCG has been described as having a transparent surface between the syntactic and the semantic. It is this property which makes it attractive for our purposes of providing a conditioning structure for semantic operators. A second benefit of the formalism is that it is designed with computational efficiency in mind. While one could debate the relative merits of various linguistic formalisms the existence of mature tools and resources, such as the CCGBank (Hockenmaier and Steedman, 2007), the Groningen Meaning Bank (Basile et al., 2012) and the C&C Tools (Curran et al., 2007) is another big advantage for CCG.

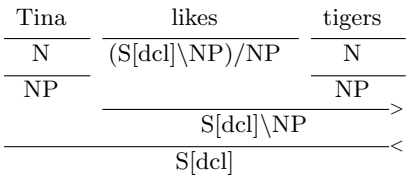

Figure 1: CCG derivation for *Tina likes tigers* with forward (>) and backward application (<).

CCG's transparent surface stems from its categorial property: Each point in a derivation corresponds directly to an interpretable category. These categories (or types) associated with each term in a CCG govern how this term can be combined with other terms in a larger structure, implicitly making them semantically expressive.

For instance in Figure 1, the word *likes* has type *(S[dcl]\NP)/NP*, which means that it first looks for a type *NP* to its right hand side. Subsequently the expression *likes tigers* (as type *S[dcl]\NP*) requires a second *NP* on its left. The final type of the phrase *S[dcl]* indicates a sentence and hence a complete CCG proof. Thus at each point in a CCG parse we can deduce the possible next steps in the derivation by considering the available types and combinatory rules.

2.2 Vector Space Models of Semantics

Vector-based approaches for semantic tasks have become increasingly popular in recent years.

Distributional representations encode an expression by its environment, assuming the context-dependent nature of meaning according to which one "shall know a word by the company it keeps" (Firth, 1957). Effectively this is usually achieved by considering the co-occurrence with other words in large corpora or the syntactic roles a word performs.

Distributional representations are frequently used to encode single words as vectors. Such rep-

[1] A C++ implementation of our models is available at `http://www.karlmoritz.com/`

resentations have then successfully been applied to a number of tasks including word sense disambiguation (Schütze, 1998) and selectional preference (Pereira et al., 1993; Lin, 1999).

While it is theoretically possible to apply the same mechanism to larger expressions, sparsity prevents learning meaningful distributional representations for expressions much larger than single words.[2]

Vector space models of compositional semantics aim to fill this gap by providing a methodology for deriving the representation of an expression from those of its parts. While distributional representations frequently serve to encode single words in such approaches this is not a strict requirement.

There are a number of ideas on how to define composition in such vector spaces. A general framework for semantic vector composition was proposed in Mitchell and Lapata (2008), with Mitchell and Lapata (2010) and more recently Blacoe and Lapata (2012) providing good overviews of this topic. Notable approaches to this issue include Baroni and Zamparelli (2010), who compose nouns and adjectives by representing them as vectors and matrices, respectively, with the compositional representation achieved by multiplication. Grefenstette and Sadrzadeh (2011) use a similar approach with matrices for relational words and vectors for arguments. These two approaches are combined in Grefenstette et al. (2013), producing a tensor-based semantic framework with tensor contraction as composition operation.

Another set of models that have very successfully been applied in this area are recursive autoencoders (Socher et al., 2011a; Socher et al., 2011b), which are discussed in the next section.

2.3 Recursive Autoencoders

Autoencoders are a useful tool to compress information. One can think of an autoencoder as a funnel through which information has to pass (see Figure 2). By forcing the autoencoder to reconstruct an input given only the reduced amount of information available inside the funnel it serves as a compression tool, representing high-dimensional objects in a lower-dimensional space.

Typically a given autoencoder, that is the functions for encoding and reconstructing data, are used on multiple inputs. By optimizing the two functions to minimize the difference between all inputs and their respective reconstructions, this autoencoder will effectively discover some hidden structures within the data that can be exploited to represent it more efficiently.

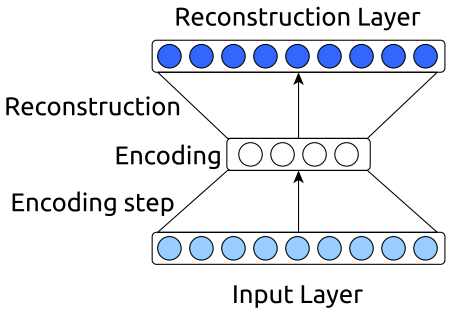

Figure 2: A simple three-layer autoencoder. The input represented by the vector at the bottom is being encoded in a smaller vector (middle), from which it is then reconstructed (top) into the same dimensionality as the original input vector.

As a simple example, assume input vectors $x_i \in \mathbb{R}^n, i \in (0..N)$, weight matrices $W^{enc} \in \mathbb{R}^{(m \times n)}, W^{rec} \in \mathbb{R}^{(n \times m)}$ and biases $b^{enc} \in \mathbb{R}^m$, $b^{rec} \in \mathbb{R}^n$. The encoding matrix and bias are used to create an encoding e_i from x_i:

$$e_i = f^{enc}(x_i) = W^{enc}x_i + b^{enc} \quad (1)$$

Subsequently $e \in \mathbb{R}^m$ is used to reconstruct x as x' using the reconstruction matrix and bias:

$$x'_i = f^{rec}(e_i) = W^{rec}e_i + b^{rec} \quad (2)$$

$\theta = (W^{enc}, W^{rec}, b^{enc}, b^{rec})$ can then be learned by minimizing the error function describing the difference between x' and x:

$$E = \frac{1}{2}\sum_{i}^{N} \left\| x'_i - x_i \right\|^2 \quad (3)$$

Now, if $m < n$, this will intuitively lead to e_i encoding a latent structure contained in x_i and shared across all $x_j, j \in (0..N)$, with θ encoding and decoding to and from that hidden structure.

It is possible to apply multiple autoencoders on top of each other, creating a deep autoencoder (Bengio et al., 2007; Hinton and Salakhutdinov, 2006). For such a multi-layered model to learn anything beyond what a single layer could learn, a non-linear transformation g needs to be applied at each layer. Usually, a variant of the sigmoid (σ)

[2] The experimental setup in (Baroni and Zamparelli, 2010) is one of the few examples where distributional representations are used for word pairs.

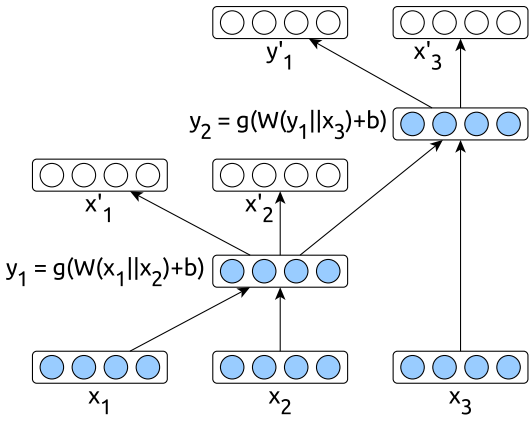

Figure 3: RAE with three inputs. Vectors with filled (blue) circles represent input and hidden units; blanks (white) denote reconstruction layers.

or hyperbolic tangent ($tanh$) function is used for g (LeCun et al., 1998).

$$f^{enc}(x_i) = g\left(W^{enc}x_i + b^{enc}\right) \quad (4)$$
$$f^{rec}(e_i) = g\left(W^{rec}e_i + b^{rec}\right)$$

Furthermore, autoencoders can easily be used as a composition function by concatenating two input vectors, such that:

$$e = f(x_1, x_2) = g\left(W(x_1\|x_2) + b\right) \quad (5)$$
$$(x_1'\|x_2') = g\left(W'e + b'\right)$$

Extending this idea, recursive autoencoders (RAE) allow the modelling of data of variable size. By setting the $n = 2m$, it is possible to recursively combine a structure into an autoencoder tree. See Figure 3 for an example, where x_1, x_2, x_3 are recursively encoded into y_2.

The recursive application of autoencoders was first introduced in Pollack (1990), whose recursive auto-associative memories learn vector representations over pre-specified recursive data structures. More recently this idea was extended and applied to dynamic structures (Socher et al., 2011b).

These types of models have become increasingly prominent since developments within the field of Deep Learning have made the training of such hierarchical structures more effective and tractable (LeCun et al., 1998; Hinton et al., 2006).

Intuitively the top layer of an RAE will encode aspects of the information stored in all of the input vectors. Previously, RAE have successfully been applied to a number of tasks including sentiment analysis, paraphrase detection, relation extraction and 3D object identification (Blacoe and Lapata, 2012; Socher et al., 2011b; Socher et al., 2012a).

Model	CCG Elements
CCAE-A	parse
CCAE-B	parse + rules
CCAE-C	parse + rules + types
CCAE-D	parse + rules + child types

Table 1: Aspects of the CCG formalism used by the different models explored in this paper.

3 Model

The models in this paper combine the power of recursive, vector-based models with the linguistic intuition of the CCG formalism. Their purpose is to learn semantically meaningful vector representations for sentences and phrases of variable size, while the purpose of this paper is to investigate the use of syntax and linguistic formalisms in such vector-based compositional models.

We assume a CCG parse to be given. Let C denote the set of combinatory rules, and T the set of categories used, respectively. We use the parse tree to structure an RAE, so that each combinatory step is represented by an autoencoder function. We refer to these models Categorial Combinatory Autoencoders (CCAE). In total this paper describes four models making increasing use of the CCG formalism (see table 1).

As an internal baseline we use model CCAE-A, which is an RAE structured along a CCG parse tree. CCAE-A uses a single weight matrix each for the encoding and reconstruction step (see Table 2. This model is similar to Socher et al. (2011b), except that we use a fixed structure in place of the greedy tree building approach. As CCAE-A uses only minimal syntactic guidance, this should allow us to better ascertain to what degree the use of syntax helps our semantic models.

Our second model (CCAE-B) uses the composition function in equation (6), with $c \in C$.

$$f^{enc}(x, y, c) = g\left(W^c_{enc}(x\|y) + b^c_{enc}\right) \quad (6)$$
$$f^{rec}(e, c) = g\left(W^c_{rec}e + b^c_{rec}\right)$$

This means that for every combinatory rule we define an equivalent autoencoder composition function by parametrizing both the weight matrix and bias on the combinatory rule (e.g. Figure 4).

In this model, as in the following ones, we assume a reconstruction step symmetric with the

Model	Encoding Function
CCAE-A	$f(x,y) = g\left(W(x\|y) + b\right)$
CCAE-B	$f(x,y,c) = g\left(W^c(x\|y) + b^c\right)$
CCAE-C	$f(x,y,c,t) = g\left(\sum_{p\in\{c,t\}}\left(W^p(x\|y) + b^p\right)\right)$
CCAE-D	$f(x,y,c,tx,ty) = g\left(W^c\left(W^{tx}x + W^{ty}y\right) + b^c\right)$

Table 2: Encoding functions of the four CCAE models discussed in this paper.

$$\frac{\alpha : \mathrm{X/Y} \quad \beta : \mathrm{Y}}{\alpha\beta : X}> \qquad g\left(W^{>}_{enc}(\alpha\|\beta) + b^{>}_{enc}\right)$$

Figure 4: Forward application as CCG combinator and autoencoder rule respectively.

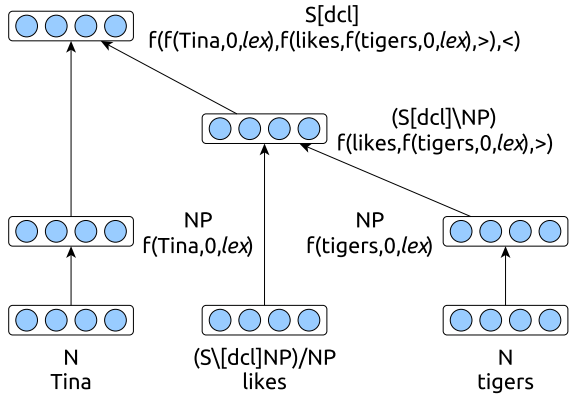

Figure 5: CCAE-B applied to *Tina likes tigers*. Next to each vector are the CCG category (top) and the word or function representing it (bottom). *lex* describes the unary type-changing operation. $>$ and $<$ are forward and backward application.

composition step. For the remainder of this paper we will focus on the composition step and drop the use of *enc* and *rec* in variable names where it isn't explicitly required. Figure 5 shows model CCAE-B applied to our previous example sentence.

While CCAE-B uses only the combinatory rules, we want to make fuller use of the linguistic information available in CCG. For this purpose, we build another model CCAE-C, which parametrizes on both the combinatory rule $c \in C$ and the CCG category $t \in T$ at every step (see Figure 2). This model provides an additional degree of insight, as the categories T are semantically and syntactically more expressive than the CCG combinatory rules by themselves. Summing over weights parametrised on c and t respectively, adds an additional degree of freedom and also allows for some model smoothing.

An alternative approach is encoded in model CCAE-D. Here we consider the categories not of the element represented, but of the elements it is generated from together with the combinatory rule applied to them. The intuition is that in the first step we transform two expressions based on their syntax. Subsequently we combine these two conditioned on their joint combinatory rule.

4 Learning

In this section we briefly discuss unsupervised learning for our models. Subsequently we describe how these models can be extended to allow for semi-supervised training and evaluation.

Let $\theta = (\mathcal{W}, \mathcal{B}, L)$ be our model parameters and λ a vector with regularization parameters for all model parameters. $\mathcal{W}$ represents the set of all weight matrices, $\mathcal{B}$ the set of all biases and L the set of all word vectors. Let N be the set of training data consisting of tree-nodes n with inputs x_n, y_n and reconstruction r_n. The error given n is:

$$E(n|\theta) = \frac{1}{2}\left\| r_n - (x_n\|y_n) \right\|^2 \tag{7}$$

The gradient of the regularised objective function then becomes:

$$\frac{\partial J}{\partial \theta} = \frac{1}{N}\sum_{n}^{N}\frac{\partial E(n|\theta)}{\partial \theta} + \lambda\theta \tag{8}$$

We learn the gradient using backpropagation through structure (Goller and Küchler, 1996), and minimize the objective function using L-BFGS.

For more details about the partial derivatives used for backpropagation, see the documentation accompanying our model implementation.[3]

[3] http://www.karlmoritz.com/

4.1 Supervised Learning

The unsupervised method described so far learns a vector representation for each sentence. Such a representation can be useful for some tasks such as paraphrase detection, but is not sufficient for other tasks such as sentiment classification, which we are considering in this paper.

In order to extract sentiment from our models, we extend them by adding a supervised classifier on top, using the learned representations v as input for a binary classification model:

$$pred(l{=}1|v,\theta) = \text{sigmoid}(W_{label}\, v + b_{label}) \quad (9)$$

Given our corpus of CCG parses with label pairs (N, l), the new objective function becomes:

$$J = \frac{1}{N} \sum_{(N,l)} E(N,l,\theta) + \frac{\lambda}{2}||\theta||^2 \quad (10)$$

Assuming each node $n \in N$ contains children x_n, y_n, encoding e_n and reconstruction r_n, so that $n = \{x, y, e, r\}$ this breaks down into:

$$E(N,l,\theta) = \quad (11)$$
$$\sum_{n \in N} \alpha E_{rec}\,(n,\theta) + (1{-}\alpha) E_{lbl}(e_n, l, \theta)$$

$$E_{rec}(n,\theta) = \frac{1}{2}\left\| [x_n \| y_n] - r_n \right\|^2 \quad (12)$$

$$E_{lbl}(e,l,\theta) = \frac{1}{2}\left\| l - e \right\|^2 \quad (13)$$

This method of introducing a supervised aspect to the autoencoder largely follows the model described in Socher et al. (2011b).

5 Experiments

We describe a number of standard evaluations to determine the comparative performance of our model. The first task of sentiment analysis allows us to compare our CCG-conditioned RAE with similar, existing models. In a second experiment, we apply our model to a compound similarity evaluation, which allows us to evaluate our models against a larger class of vector-based models (Blacoe and Lapata, 2012). We conclude with some qualitative analysis to get a better idea of whether the combination of CCG and RAE can learn semantically expressive embeddings.

In our experiments we use the hyperbolic tangent as nonlinearity g. Unless stated otherwise we use word-vectors of size 50, initialized using the embeddings provided by Turian et al. (2010) based on the model of Collobert and Weston (2008).[4]

We use the C&C parser (Clark and Curran, 2007) to generate CCG parse trees for the data used in our experiments. For models CCAE-C and CCAE-D we use the 25 most frequent CCG categories (as extracted from the British National Corpus) with an additional general weight matrix in order to catch all remaining types.

5.1 Sentiment Analysis

We evaluate our model on the MPQA opinion corpus (Wiebe et al., 2005), which annotates expressions for sentiment.[5] The corpus consists of 10,624 instances with approximately 70 percent describing a negative sentiment. We apply the same pre-processing as (Nakagawa et al., 2010) and (Socher et al., 2011b) by using an additional sentiment lexicon (Wilson et al., 2005) during the model training for this experiment.

As a second corpus we make use of the sentence polarity (SP) dataset v1.0 (Pang and Lee, 2005).[6] This dataset consists of 10662 sentences extracted from movie reviews which are manually labelled with positive or negative sentiment and equally distributed across sentiment.

Experiment 1: Semi-Supervised Training In the first experiment, we use the semi-supervised training strategy described previously and initialize our models with the embeddings provided by Turian et al. (2010). The results of this evaluation are in Table 3. While we achieve the best performance on the MPQA corpus, the results on the SP corpus are less convincing. Perhaps surprisingly, the simplest model CCAE-A outperforms the other models on this dataset.

When considering the two datasets, sparsity seems a likely explanation for this difference in results: In the MPQA experiment most instances are very short with an average length of 3 words, while the average sentence length in the SP corpus is 21 words. The MPQA task is further simplified through the use or an additional sentiment lexicon. Considering dictionary size, the SP corpus has a dictionary of 22k words, more than three times the size of the MPQA dictionary.

[4] http://www.metaoptimize.com/projects/wordreprs/

[5] http://mpqa.cs.pitt.edu/

[6] http://www.cs.cornell.edu/people/pabo/movie-review-data/

Method	MPQA	SP
Voting with two lexica	81.7	63.1
MV-RNN (Socher et al., 2012b)	-	79.0
RAE (rand) (Socher et al., 2011b)	85.7	76.8
TCRF (Nakagawa et al., 2010)	86.1	77.3
RAE (init) (Socher et al., 2011b)	86.4	77.7
NB (Wang and Manning, 2012)	86.7	79.4
CCAE-A	86.3	77.8
CCAE-B	87.1	77.1
CCAE-C	87.1	77.3
CCAE-D	**87.2**	76.7

Table 3: Accuracy of sentiment classification on the sentiment polarity (SP) and MPQA datasets. For NB we only display the best result among a larger group of models analysed in that paper.

This issue of sparsity is exacerbated in the more complex CCAE models, where the training points are spread across different CCG types and rules. While the initialization of the word vectors with previously learned embeddings (as was previously shown by Socher et al. (2011b)) helps the models, all other model variables such as composition weights and biases are still initialised randomly and thus highly dependent on the amount of training data available.

Experiment 2: Pretraining Due to our analysis of the results of the initial experiment, we ran a second series of experiments on the SP corpus. We follow (Scheible and Schütze, 2013) for this second series of experiments, which are carried out on a random 90/10 training-testing split, with some data reserved for development.

Instead of initialising the model with external word embeddings, we first train it on a large amount of data with the aim of overcoming the sparsity issues encountered in the previous experiment. Learning is thus divided into two steps:

The first, unsupervised training phase, uses the British National Corpus together with the SP corpus. In this phase only the reconstruction signal is used to learn word embeddings and transformation matrices. Subsequently, in the second phase, only the SP corpus is used, this time with both the reconstruction and the label error.

By learning word embeddings and composition matrices on more data, the model is likely to generalise better. Particularly for the more complex models, where the composition functions are conditioned on various CCG parameters, this should help to overcome issues of sparsity.

	Training	
Model	Regular	Pretraining
CCAE-A	77.8	79.5
CCAE-B	76.9	79.8
CCAE-C	77.1	81.0
CCAE-D	76.9	79.7

Table 4: Effect of pretraining on model performance on the SP dataset. Results are reported on a random subsection of the SP corpus; thus numbers for the regular training method differ slightly from those in Table 3.

If we consider the results of the pre-trained experiments in Table 4, this seems to be the case. In fact, the trend of the previous results has been reversed, with the more complex models now performing best, whereas in the previous experiments the simpler models performed better. Using the Turian embeddings instead of random initialisation did not improve results in this setup.

5.2 Compound Similarity

In a second experiment we use the dataset from Mitchell and Lapata (2010) which contains similarity judgements for adjective-noun, noun-noun and verb-object pairs.[7] All compound pairs have been ranked for semantic similarity by a number of human annotators. The task is thus to rank these pairs of word pairs by their semantic similarity.

For instance, the two compounds *vast amount* and *large quantity* are given a high similarity score by the human judges, while *northern region* and *early age* are assigned no similarity at all.

We train our models as fully unsupervised autoencoders on the British National Corpus for this task. We assume fixed parse trees for all of the compounds (Figure 6), and use these to compute compound level vectors for all word pairs. We subsequently use the cosine distance between each compound pair as our similarity measure. We use Spearman's rank correlation coefficient (ρ) for evaluation; hence there is no need to rescale our scores (-1.0 – 1.0) to the original scale (1.0 – 7.0).

Blacoe and Lapata (2012) have an extensive comparison of the performance of various vector-based models on this data set to which we compare our model in Table 5. The CCAE models outper-

[7] http://homepages.inf.ed.ac.uk/mlap/resources/index.html

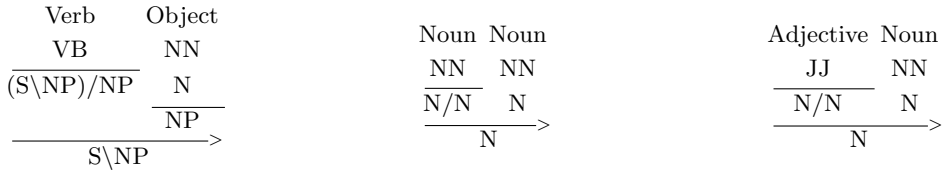

Figure 6: Assumed CCG parse structure for the compound similarity evaluation.

Method	Adj-N	N-N	V-Obj
Human	0.52	0.49	0.55
(Blacoe and Lapata, 2012)			
$\odot$/+	0.21 - 0.48	0.22 - 0.50	0.18 - 0.35
RAE	0.19 - 0.31	0.24 - 0.30	0.09 - 0.28
CCAE-B	0.38	0.44	0.34
CCAE-C	0.38	0.41	0.23
CCAE-D	0.41	0.44	0.29

Table 5: Correlation coefficients of model predictions for the compound similarity task. Numbers show Spearman's rank correlation coefficient (ρ). Higher numbers indicate better correlation.

form the RAE models provided by Blacoe and Lapata (2012), and score towards the upper end of the range of other models considered in that paper.

5.3 Qualitative Analysis

To get better insight into our models we also perform a small qualitative analysis. Using one of the models trained on the MPQA corpus, we generate word-level representations of all phrases in this corpus and subsequently identify the most related expressions by using the cosine distance measure. We perform this experiment on all expressions of length 5, considering all expressions with a word length between 3 and 7 as potential matches.

As can be seen in Table 6, this works with varying success. Linking expressions such as *conveying the message of peace* and *safeguard(ing) peace and security* suggests that the model does learn some form of semantics.

On the other hand, the connection between *expressed their satisfaction and support* and *expressed their admiration and surprise* suggests that the pure word level content still has an impact on the model analysis. Likewise, the expressions *is a story of success* and *is a staunch supporter* have some lexical but little semantic overlap. Further reducing this link between the lexical and the semantic representation is an issue that should be addressed in future work in this area.

6 Discussion

Overall, our models compare favourably with the state of the art. On the MPQA corpus model CCAE-D achieves the best published results we are aware of, whereas on the SP corpus we achieve competitive results. With an additional, unsupervised training step we achieved results beyond the current state of the art on this task, too.

Semantics The qualitative analysis and the experiment on compounds demonstrate that the CCAE models are capable of learning semantics. An advantage of our approach—and of autoencoders generally—is their ability to learn in an unsupervised setting. The pre-training step for the sentiment task was essentially the same training step as used in the compound similarity task. While other models such as the MV-RNN (Socher et al., 2012b) achieve good results on a particular task, they do not allow unsupervised training. This prevents the possiblity of pretraining, which we showed to have a big impact on results, and further prevents the training of general models: The CCAE models can be used for multiple tasks without the need to re-train the main model.

Complexity Previously in this paper we argued that our models combined the strengths of other approaches. By using a grammar formalism we increase the expressive power of the model while the complexity remains low. For the complexity analysis see Table 7. We strike a balance between the greedy approaches (e.g. Socher et al. (2011b)), where learning is quadratic in the length of each sentence and existing syntax-driven approaches such as the MV-RNN of Socher et al. (2012b), where the size of the model, that is the number of variables that needs to be learned, is quadratic in the size of the word-embeddings.

Sparsity Parametrizing on CCG types and rules increases the size of the model compared to a greedy RAE (Socher et al., 2011b). The effect of this was highlighted by the sentiment analysis task, with the more complex models performing

Expression	Most Similar
convey the message of peace	safeguard peace and security
keep alight the flame of	keep up the hope
has a reason to repent	has no right
a significant and successful strike	a much better position
it is reassuring to believe	it is a positive development
expressed their satisfaction and support	expressed their admiration and surprise
is a story of success	is a staunch supporter
are lining up to condemn	are going to voice their concerns
more sanctions should be imposed	charges being leveled
could fray the bilateral goodwill	could cause serious damage

Table 6: Phrases from the MPQA corpus and their semantically closest match according to CCAE-D.

	Complexity	
Model	Size	Learning
MV-RNN	$\mathcal{O}(nw^2)$	$O(l)$
RAE	$\mathcal{O}(nw)$	$O(l^2)$
CCAE-*	$\mathcal{O}(nw)$	$O(l)$

Table 7: Comparison of models. n is dictionary size, w embedding width, l is sentence length. We can assume $l \gg n \gg w$. Additional factors such as CCG rules and types are treated as small constants for the purposes of this analysis.

worse in comparison with the simpler ones. We were able to overcome this issue by using additional training data. Beyond this, it would also be interesting to investigate the relationships between different types and to derive functions to incorporate this into the learning procedure. For instance model learning could be adjusted to enforce some mirroring effects between the weight matrices of forward and backward application, or to support similarities between those of forward application and composition.

CCG-Vector Interface Exactly how the information contained in a CCG derivation is best applied to a vector space model of compositionality is another issue for future research. Our investigation of this matter by exploring different model setups has proved somewhat inconclusive. While CCAE-D incorporated the deepest conditioning on the CCG structure, it did not decisively outperform the simpler CCAE-B which just conditioned on the combinatory operators. Issues of sparsity, as shown in our experiments on pretraining, have a significant influence, which requires further study.

7 Conclusion

In this paper we have brought a more formal notion of semantic compositionality to vector space models based on recursive autoencoders. This was achieved through the use of the CCG formalism to provide a conditioning structure for the matrix vector products that define the RAE.

We have explored a number of models, each of which conditions the compositional operations on different aspects of the CCG derivation. Our experimental findings indicate a clear advantage for a deeper integration of syntax over models that use only the bracketing structure of the parse tree.

The most effective way to condition the compositional operators on the syntax remains unclear. Once the issue of sparsity had been addressed, the complex models outperformed the simpler ones. Among the complex models, however, we could not establish significant or consistent differences to convincingly argue for a particular approach.

While the connections between formal linguistics and vector space approaches to NLP may not be immediately obvious, we believe that there is a case for the continued investigation of ways to best combine these two schools of thought. This paper represents one step towards the reconciliation of traditional formal approaches to compositional semantics with modern machine learning.

Acknowledgements

We thank the anonymous reviewers for their feedback and Richard Socher for providing additional insight into his models. Karl Moritz would further like to thank Sebastian Riedel for hosting him at UCL while this paper was written. This work has been supported by the EPSRC.

References

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of EMNLP*, pages 1183–1193.

Valerio Basile, Johan Bos, Kilian Evang, and Noortje Venhuizen. 2012. Developing a large semantically annotated corpus. In *Proceedings of LREC*, pages 3196–3200, Istanbul, Turkey.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. 2007. Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems 19*, pages 153–160.

William Blacoe and Mirella Lapata. 2012. A comparison of vector-based representations for semantic composition. In *Proceedings of EMNLP-CoNLL*, pages 546–556.

Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with ccg and log-linear models. *CL*, 33(4):493–552, December.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of ICML*.

James Curran, Stephen Clark, and Johan Bos. 2007. Linguistically motivated large-scale nlp with c&c and boxer. In *Proceedings of ACL Demo and Poster Sessions*, pages 33–36.

J. R. Firth. 1957. A synopsis of linguistic theory 1930-55. 1952-59:1–32.

Gottfried Frege. 1892. Über Sinn und Bedeutung. In Mark Textor, editor, *Funktion - Begriff - Bedeutung*, volume 4 of *Sammlung Philosophie*. Vandenhoeck & Ruprecht, Göttingen.

Christoph Goller and Andreas Küchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *Proceedings of the ICNN-96*, pages 347–352. IEEE.

Edward Grefenstette and Mehrnoosh Sadrzadeh. 2011. Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of EMNLP*, pages 1394–1404.

Edward Grefenstette, Georgiana Dinu, Yao-Zhong Zhang, Mehrnoosh Sadrzadeh, and Marco Baroni. 2013. Multi-step regression learning for compositional distributional semantics.

G. E. Hinton and R. R. Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

Geoffrey E. Hinton, Simon Osindero, Max Welling, and Yee Whye Teh. 2006. Unsupervised discovery of nonlinear structure using contrastive backpropagation. *Cognitive Science*, 30(4):725–731.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A Corpus of CCG Derivations and Dependency Structures Extracted from the Penn Treebank. *CL*, 33(3):355–396, September.

Ray Jackendoff. 1972. *Semantic Interpretation in Generative Grammar*. MIT Press, Cambridge, MA.

Marcus Kracht. 2008. Compositionality in Montague Grammar. In Edouard Machery und Markus Werning Wolfram Hinzen, editor, *Handbook of Compositionality*, pages 47 – 63. Oxford University Press.

Yann LeCun, Leon Bottou, Genevieve Orr, and Klaus-Robert Muller. 1998. Efficient backprop. In G. Orr and Muller K., editors, *Neural Networks: Tricks of the trade*. Springer.

Dekang Lin. 1999. Automatic identification of non-compositional phrases. In *Proceedings of ACL*, pages 317–324.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *In Proceedings of ACL*, pages 236–244.

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

R. Montague. 1970. Universal grammar. *Theoria*, 36(3):373–398.

Tetsuji Nakagawa, Kentaro Inui, and Sadao Kurohashi. 2010. Dependency tree-based sentiment classification using crfs with hidden variables. In *NAACL-HLT*, pages 786–794.

Bo Pang and Lillian Lee. 2005. Seeing stars: exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of ACL*, pages 115–124.

Francis Jeffry Pelletier. 1994. The principle of semantic compositionality. *Topoi*, 13:11–24.

Fernando Pereira, Naftali Tishby, and Lillian Lee. 1993. Distributional clustering of english words. In *Proceedings of ACL*, ACL '93, pages 183–190.

Jordan B. Pollack. 1990. Recursive distributed representations. *Artificial Intelligence*, 46:77–105.

Christian Scheible and Hinrich Schütze. 2013. Cutting recursive autoencoder trees. In *Proceedings of the International Conference on Learning Representations*.

Hinrich Schütze. 1998. Automatic word sense discrimination. *CL*, 24(1):97–123, March.

Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. 2011a. Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection. In *Advances in Neural Information Processing Systems 24*.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011b. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of EMNLP*, pages 151–161.

Richard Socher, Brody Huval, Bharath Bhat, Christopher D. Manning, and Andrew Y. Ng. 2012a. Convolutional-Recursive Deep Learning for 3D Object Classification. In *Advances in Neural Information Processing Systems 25*.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012b. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of EMNLP-CoNLL*, pages 1201–1211.

Mark Steedman and Jason Baldridge, 2011. *Combinatory Categorial Grammar*, pages 181–224. Wiley-Blackwell.

Anna Szabolcsi. 1989. Bound Variables in Syntax: Are There Any? In Renate Bartsch, Johan van Benthem, and Peter van Emde Boas, editors, *Semantics and Contextual Expression*, pages 295–318. Foris, Dordrecht.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of ACL*, pages 384–394.

Sida Wang and Christopher D. Manning. 2012. Baselines and bigrams: simple, good sentiment and topic classification. In *Proceedings of ACL*, pages 90–94.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation*, 39(2-3):165–210.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of EMNLP-HLT*, HLT '05, pages 347–354.