A Comparison of Vector-based Representations for Semantic Composition

William Blacoe and **Mirella Lapata**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
w.b.blacoe@sms.ed.ac.uk, mlap@inf.ed.ac.uk

Abstract

In this paper we address the problem of modeling compositional meaning for phrases and sentences using distributional methods. We experiment with several possible combinations of representation and composition, exhibiting varying degrees of sophistication. Some are shallow while others operate over syntactic structure, rely on parameter learning, or require access to very large corpora. We find that shallow approaches are as good as more computationally intensive alternatives with regards to two particular tests: (1) phrase similarity and (2) paraphrase detection. The sizes of the involved training corpora and the generated vectors are not as important as the fit between the meaning representation and compositional method.

1 Introduction

Distributional models of semantics have seen considerable success at simulating a wide range of behavioral data in tasks involving semantic cognition and also in practical applications. For example, they have been used to model judgments of semantic similarity (McDonald, 2000) and association (Denhire and Lemaire, 2004; Griffiths et al., 2007) and have been shown to achieve human level performance on synonymy tests (Landauer and Dumais, 1997; Griffiths et al., 2007) such as those included in the Test of English as a Foreign Language (TOEFL). This ability has been put to practical use in numerous natural language processing tasks such as automatic thesaurus extraction (Grefenstette, 1994), word sense discrimination (Schütze, 1998), language modeling (Bellegarda, 2000), and the identification of analogical relations (Turney, 2006).

While much research has been directed at the most effective ways of constructing representations for individual words, there has been far less consensus regarding the representation of larger constructions such as phrases and sentences. The problem has received some attention in the connectionist literature, particularly in response to criticisms of the ability of connectionist representations to handle complex structures (Smolensky, 1990; Plate, 1995). More recently, several proposals have been put forward for computing the meaning of word combinations in vector spaces. This renewed interest is partly due to the popularity of distributional methods and their application potential to tasks that require an understanding of larger phrases or complete sentences.

For example, Mitchell and Lapata (2010) introduce a general framework for studying vector *composition*, which they formulate as a function f of two vectors $\mathbf{u}$ and $\mathbf{v}$. Different composition models arise, depending on how f is chosen. Assuming that composition is a linear function of the Cartesian product of $\mathbf{u}$ and $\mathbf{v}$ allows to specify *additive* models which are by far the most common method of vector combination in the literature (Landauer and Dumais, 1997; Foltz et al., 1998; Kintsch, 2001). Alternatively, assuming that composition is a linear function of the tensor product of $\mathbf{u}$ and $\mathbf{v}$, gives rise to models based on *multiplication*.

One of the most sophisticated proposals for semantic composition is that of Clark et al. (2008) and the more recent implementation of Grefenstette and

Sadrzadeh (2011a). Using techniques from logic, category theory, and quantum information they develop a compositional distributional semantics that brings type-logical and distributional vector space models together. In their framework, words belong to different type-based categories and different categories exist in different dimensional spaces. The category of a word is decided by the number and type of adjoints (arguments) it can take and the composition of a sentence results in a vector which exists in sentential space. Verbs, adjectives and adverbs act as relational functions, are represented by matrices, and modify the properties of nouns, that are represented by vectors (see also Baroni and Zamparelli (2010) for a proposal similar in spirit). Clarke (2012) introduces context-theoretic semantics, a general framework for combining vector representations, based on a mathematical theory of meaning as context, and shows that it can be used to describe a variety of models including that of Clark et al. (2008).

Socher et al. (2011a) and Socher et al. (2011b) present a framework based on recursive neural networks that learns vector space representations for multi-word phrases and sentences. The network is given a list of word vectors as input and a binary tree representing their syntactic structure. Then, it computes an n-dimensional representation p of two n-dimensional children and the process is repeated at every parent node until a representation for a full tree is constructed. Parent representations are computed essentially by concatenating the representations of their children. During training, the model tries to minimize the reconstruction errors between the n-dimensional parent vectors and those representing their children. This model can also compute compositional representations when the tree structure is not given, e.g., by greedily inferring a binary tree.

Although the type of function used for vector composition has attracted much attention, relatively less emphasis has been placed on the basic distributional representations on which the composition functions operate. In this paper, we examine three types of distributional representation of increasing sophistication and their effect on semantic composition. These include a simple semantic space, where a word's vector represents its co-occurrence with neighboring words (Mitchell and Lapata, 2010), a syntax-aware space based on weighted distributional tuples that encode typed co-occurrence relations among words (Baroni and Lenci, 2010), and word embeddings computed with a neural language model (Bengio, 2001; Collobert and Weston, 2008). Word embeddings are distributed representations, low-dimensional and real-valued. Each dimension of the embedding represents a latent feature of the word, hopefully capturing useful syntactic and semantic properties.

Using these representations, we construct several compositional models, based on addition, multiplication, and recursive neural networks. We assess the effectiveness of these models using two evaluation protocols. The first one involves modeling similarity judgments for short phrases gathered in human experiments (Mitchell and Lapata, 2010). The second one is paraphrase detection, i.e., the task of examining two sentences and determining whether they have the same meaning (Socher et al., 2011a). We find that shallow approaches are as good as more computationally intensive alternatives. They achieve considerable semantic expressivity without any learning, sophisticated linguistic processing, or access to very large corpora.

Our contributions in this work are three-fold: an empirical comparison of a broad range of compositional models, some of which are introduced here for the first time; the use of an evaluation methodology that takes into account the full spectrum of compositionality from phrases to sentences; and the empirical finding that relatively simple compositional models can be used to perform competitively on the paraphrase detection and phrase similarity tasks.

2 Modeling

The elementary objects that we operate on are vectors associated with words. We instantiate these word representations following three distinct semantic space models which we describe in Section 2.1 below. Analogously, in Section 2.2 we consider three methods of vector composition, i.e., how a phrase or a sentence can be represented as a vector using the vectors of its constituent words. Combining different vector representations and composition methods gives rise to several compositional models whose performance we evaluate in Sections 3 and 4.

2.1 Word Representations

For all of our experiments we employ column vectors from a Cartesian, finitely-dimensional space. The dimensionality will depend on the source of the vectors involved. Similarly, the component values inside each source's vectors are not to be interpreted in the same manner. Nonetheless, they have in common that they originate from distributive corpus statistics.

Co-occurence-based Semantic Space Word meaning is commonly represented in a high-dimensional space, where each component corresponds to some contextual element in which the word is found. The contextual elements can be words themselves, or larger linguistic units such as sentences or documents, or even more complex linguistic representations such as the argument slots of predicates. A semantic space that is often employed in studying compositionality across a variety of tasks (Mitchell and Lapata, 2010; Grefenstette and Sadrzadeh, 2011a) uses a context window of five words on either side of the target word, and 2,000 vector dimensions. These are the common context words in the British National Corpus (BNC), a corpus of about 100 million tokens. Their values are set to the ratio of the probability of the context word given the target word to the probability of the context word overall.

More formally, let us consider the BNC as a set of sentences:

$$BNC = \{Sen_1^{(BNC)}, ..., Sen_{n_{BNC}}^{(BNC)}\} \tag{1}$$

where the i-th sentence is a sequence of words $Sen_i = (w_1^{(i)}, ..., w_{n_i}^{(i)})$ from the BNC's vocabulary Voc_{BNC}. Then $freq_w$ is the amount of times that each word $w \in Voc_{BNC}$ appears in the BNC. Mitchell and Lapata (2010) collect the M most frequent non-stoplist words in the set $ctxt_{top} = \{w_1^{(top)}, ..., w_M^{(top)}\}$ and let them consitute the word vectors' dimensions. Each dimension's value is obtained from a co-occurrence count:

$$coCount_w[j] = \sum_{i=1}^{n_{BNC}} \sum_{t=1}^{n_i} \tag{2}$$

$$|\{k \in [t-5; t+5] \,|\, w_t^{(i)} = w,\ w_k^{(i)} = w_j^{(top)}\}|$$

for $w \in Voc_{BNC}$ and $j = 1, ..., M$. Using these counts, they define word vectors component-wise.

$$wdVec_w^{(rp)}[j] = \frac{p(w_j^{(top)}|w)}{p(w_j^{(top)})} = \tag{3}$$

$$\frac{coCount_w[j]}{freq_w} \times \frac{totalCount}{freq_{w_j^{(top)}}}$$

for $j = 1, ..., M$, where $totalCount$ is the total number of words in the BNC.

This space is relatively simple, it has few parameters, requires no preprocessing other than tokenization and involves no syntactic information or parameter learning. Despite its simplicity, it is a good starting point for studying representations for compositional models as a baseline against which to evaluate more elaborate models.

Neural Language Model Another perhaps less well-known approach to meaning representation is to represent words as continuous vectors of parameters. Such word vectors can be obtained with an unsupervised *neural language model* (NLM, Bengio (2001); Collobert and Weston (2008)) which jointly learns an embedding of words into a vector space and uses these vectors to predict how likely a word is, given its context.

We induced word embeddings with Collobert and Weston (2008)'s neural language model. The model is discriminative and non-probabilistic. Each word $i \in \mathcal{D}$ (the vocabulary) is embedded into a d-dimensional space using a lookup table $LT_W(\cdot)$:

$$LT_W(i) = W_i \tag{4}$$

where $W \in \mathbb{R}^{d \times |\mathcal{D}|}$ is a matrix of parameters to be learned. $W_i \in \mathbb{R}^d$ is the i-th column of W and d is the word vector size to be chosen by the user. The parameters W are automatically trained during the learning process using backpropagation.

Specifically, at each training update, the model reads an n-gram $x = (w_1, \ldots, w_n)$ from the corpus. The n-gram is paired with a *corrupted* n-gram $\tilde{x} = (w_1, \ldots, \tilde{w_n})$ where $\tilde{w_n} \neq w_n$ is chosen uniformly from the vocabulary. The model concatenates the learned embeddings of the n words and predicts a score for the n-gram sequence using the learned embeddings as features. The training criterion is that *n*-grams that are present in the training corpus must have a score at least some margin higher than the corrupted *n*-grams. The model learns via gradient descent over the neural network parameters and the embedding lookup table. Word vectors are stored in a word embedding matrix which captures syntactic and semantic information from co-occurrence statistics. As these representations are learned, albeit in an unsupervised manner, one would hope that they capture word meanings more succinctly, compared to the simpler distributional representations that are merely based on co-occurrence.

We trained the neural language model on the BNC. We optimized the model's parameters on a word similarity task using 4% of the BNC as development data. Specifically, we used WordSim353, a benchmark dataset (Finkelstein et al., 2001), consisting of relatedness judgments (on a scale of 0 to 10) for 353 word pairs. We experimented with vectors of varying dimensionality (ranging from 50 to 200, with a step size of 50). The size of the target word's context window was 2, 3 and 4 in turn. The rate at which embeddings were learned ranged from 3.4×10^{-10} to 6.7×10^{-10} to 10^{-9}. We ran each training process for 1.1×10^8 to 2.7×10^8 iterations (ca. 2 days). We obtained the best results with 50 dimensions, a context window of size 4, and a embedding learning rate of 10^{-9}. The NLM with these parameters was then trained for 1.51×10^9 iterations (ca. 2 weeks).

Figure 1 illustrates a two-dimensional projection of the embeddings for the 500 most common words in the BNC. We only show two out of the actual 50 dimensions involved, but one can already begin to see clusterings of a syntactic and semantic nature. In one corner, for example, we encounter a grouping of possessive pronouns together with the possessive clitic *'s*. The singular ones *my*, *her* and *his* are closely positioned, as are the plural ones *our*, *your* and *their*. Also, there is a clustering of socio-political terms, such as *international*, *country*, *national*, *government*, and *council*.

Distributional Memory Tensor Baroni and Lenci (2010) present Distributional Memory, a generalized framework for distributional semantics from which several special-purpose models can be derived. In their framework distributional information is extracted from the corpus once, in the form of a set of weighted word-link-word tuples arranged into a third-order tensor. Different matrices are then generated from the tensor, and their rows and columns give rise to different semantic spaces appropriate for capturing different semantic problems. In this way, the same distributional information can be shared across tasks such as word similarity or analogical learning.

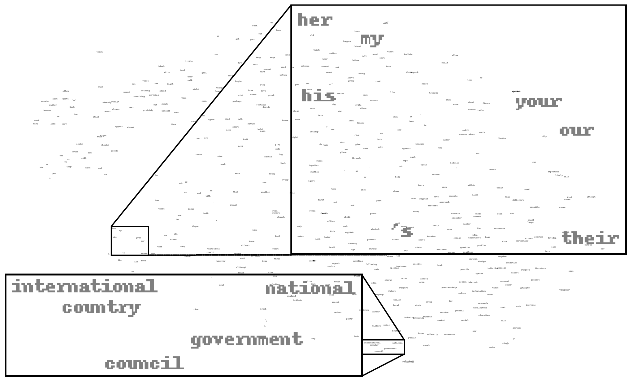

Figure 1: A two-dimensional projection of the word embeddings we trained on the BNC using Turian et al.'s (2010) implementation of the NLM. Two small sections have been blown up to a legible scale. They show examples of syntactic and semantic clustering, respectively.

word *w*	link *l*	co-word *v*	value *c*
1950s-n	of	essence-n	2.4880
1950s-n	during	bring-v	16.4636
Anyone-n	nmod	reaction-n	1.2161
American-n	coord-1	athlete-n	5.6485
American-j	nmod	wasp-n	3.4945
American-n	such_as-1	country-n	14.4269
American-n	sbj_tr	build-v	23.1014

Table 1: Example entries in Baroni and Lenci (2010)'s tensor

More formally, Baroni and Lenci (2010) construct a 3-dimensional tensor T assigning a value c to instances of word pairs w, v and a connecting link-word l. This representation operates over a dependency-parsed corpus and the scores c are obtained via counting the occurrences of tuples, and weighting the raw counts by mutual information. Table 1 presents examples of tensor entries. These were taken from a distributional memory tensor[1]

[1]Available at `http://clic.cimec.unitn.it/dm/`.

frequency	link l	co-word v
17059	obj	include-v
16713	obj	use-v
16573	obj	call-v
16475	obj	see-v
15962	obj	make-v
15707	nmod-1	other-j
15554	nmod-1	new-j
15224	obj	find-v
15221	nmod-1	more-j
14715	nmod-1	first-j
14348	obj	give-v

Table 2: The 11 most frequent contexts in Baroni and Lenci (2010)'s tensor (v and j represent verbs and adjectives, respectively).

that Baroni and Lenci obtained via preprocessing several corpora: the web-derived ukWac corpus of about 1.915 billion words, a mid-2009 dump of the English Wikipedia containing about 820 million words, and the BNC.

Extracting a 3-dimensional tensor from the BNC alone would create very sparse representations. We therefore extract so-called word-fibres, essentially projections onto a lower-dimensional subspace, from the same tensor Baroni and Lenci (2010) collectively derived from the 3 billion word corpus just described (henceforth 3-BWC). We view the 3-dimensional tensor

$$T = \{(w_1^{(T)}, l_1^{(T)}, v_1^{(T)}, c_1^{(T)}), ...\} \tag{5}$$

as a mapping which assigns each target word w a non-zero value c, given the context (l, v). All word-context combinations not listed in T are implicitly assigned a zero value.

Now we consider two possible approaches for obtaining vectors, depending on their application. First, we let the D most frequent contexts

$$ctxt_D = \{(l_1, v_1), ..., (l_D, v_D)\} \tag{6}$$

constitute the D dimensions that each word vector will have. Table 2 shows the 11 contexts (l, v) that appear most frequently in T. Thus, each target word's vector is defined component-wise as:

$$wdVec_w[j] = \begin{cases} c, & \text{if } (w, l_j, v_j, c) \in T \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

for $j = 1, ..., D$. This approach is used when a fixed vector dimensionality is necessary.

A more dynamic approach is possible when very few words $w_1, ..., w_n$ are involved in a test. Their representations can then have a denser format, that is, with no zero-valued components. For this we identify the set of contexts common to the words involved,

$$\begin{aligned} ctxt_{dyn} &= \{(l_1^{(dyn)}, v_1^{(dyn)}), (l_2^{(dyn)}, v_2^{(dyn)}), ...\} \qquad (8) \\ &= \{(l, v) \mid (w_i, l, v, c) \in T, c \in \mathbb{R}, i = 1, ..., n\} \end{aligned}$$

Each context (l, v) again constitutes a vector dimension. The dimensionality varies strongly depending on the selection of words, but if n does not exceed 4, the dimensionality $|ctxt_{dyn}|$ will typically be substantial enough. In this approach, each word's vector consists of the values c found along with that word and its context in the tensor.

$$wdVec_{w_i}[j] = c, \tag{9}$$

where $(w_i, l_j^{(dyn)}, v_j^{(dyn)}, c) \in T$, for $j = 1, ..., |ctxt_{dyn}|$.

2.2 Composition Methods

In our experiments we compose word vectors to create representations for phrase vectors and sentence vectors. The phrases we are interested in consist of two words each: an adjective and a noun like *black hair*, a compound noun made up of two nouns such as *oil industry*, or a verbal phrase with a transitive verb and an object noun, e.g., *pour tea*.

Conceiving of a phrase $phr = (w_1, w_2)$ as a binary tuple of words, we obtain its vector from its words' vectors either by addition:

$$phrVec_{(w_1,w_2)} = wdVec_{w_1} + wdVec_{w_2} \tag{10}$$

or by point-wise multiplication:

$$phrVec_{(w_1,w_2)} = wdVec_{w_1} \odot wdVec_{w_2} \tag{11}$$

In the same way we acquire a vector $senVec_i$ representing a sentence $Sen_i = (w_1^{(i)}, ..., w_{n_i}^{(i)})$ from the vectors for $w_1, ..., w_{n_i}$. We simply sum the existing word vectors, that is, vectors obtained via the respective corpus for words that are not on our stoplist:

$$senVec_i^{(+)}[j] = \sum_{\substack{k=1,...,n_i \\ wdVec_{w_k} \text{exists}}} wdVec_{w_k}[j] \tag{12}$$

And do the same with point-wise multiplication:

$$senVec_i^{(\odot)}[j] = \prod_{\substack{k=1,\ldots,n_i \\ wdVec_{w_k}\text{exists}}} wdVec_{w_k}[j] \qquad (13)$$

The multiplication model in (13) can be seen as an instantiation of the categorical compositional framework put forward by Clark et al. (2008). In fact, a variety of multiplication-based models can be derived from this framework; and comparisons against component-wise multiplication on phrase similarity tasks yield comparable results (Grefenstette and Sadrzadeh, 2011a; Grefenstette and Sadrzadeh, 2011b). We thus opt for the model (13) as an example of compositional models based on multiplication due to its good performance across a variety of tasks, including language modeling and prediction of reading difficulty (Mitchell, 2011).

Our third method, for creating phrase and sentence vectors alike, is the application of Socher et al. (2011a)'s model. They use the Stanford parser (Klein and Manning, 2003) to create a binary parse tree for each input phrase or sentence. This tree is then used as the basis for a deep recursive autoencoder (RAE). The aim is to construct a vector representation for the tree's root bottom-up where the leaves contain word vectors. The latter can in theory be provided by any type of semantic space, however Socher et al. use word embeddings provided by the neural language model (Collobert and Weston, 2008).

Given the binary tree input structure, the model computes parent representations p from their children (c_1, c_2) using a standard neural network layer:

$$p = f(W^{(1)}[c_1; c_2] + b^{(1)}), \qquad (14)$$

where $[c_1; c_2]$ is the concatenation of the two children, f is an element-wise activation function such as tanh, b is a bias term, and $W \in \mathbb{R}^{n \times 2n}$ is an encoding matrix that we want to learn during training. One way of assessing how well p represents its direct children is to decode their vectors in a reconstruction layer:

$$[c_1'; c_2'] = f(W^{(2)}p + b^{(2)}) \qquad (15)$$

During training, the goal is to minimize the reconstruction errors of all input pairs at nonterminal nodes p in a given parse tree by computing the square of the Euclidean distance between the original input and its reconstruction:

$$E_{rec}([c_1; c_2]) = \frac{1}{2}||[c_1; c_2] - [c_1'; c_2']||^2 \qquad (16)$$

Socher et al. (2011a) extend the standard recursive autoencoder sketched above in two ways. Firstly, they present an unfolding autoencoder that tries to reconstruct all leaf nodes underneath each node rather than only its direct children. And secondly, instead of transforming the two children directly into a parent p, they introduce another hidden layer inbetween.

We obtained three compositional models per representation resulting in nine compositional models overall. Plugging different representations into the additive and multiplicative models is relatively straightforward. The RAE can also be used with arbitrary word vectors. Socher et al. (2011a) obtain best results with 100-dimensional vectors which we also used in our experiments. NLM vectors were trained with this dimensionality on the BNC for 7.9×10^8 iterations (with window size 4 and an embedding learning rate of 10^{-9}). We constructed a simple distributional space with $M = 100$ dimensions, i.e., those connected to the 100 most frequent co-occurrence words. In the case of vectors obtained from Baroni and Lenci (2010)'s DM tensor, we differentiated between phrases and sentences, due to the disparate amount of words contained in them (see Section 2.1). To represent phrases, we used vectors of dynamic dimensionality, since these form a richer and denser representation. The sentences considered in Section 4 are too large for this approach and all word vectors must be members of the same vector space. Hence, these sentence vectors have fixed dimensionality $D = 100$, consisting of the "most significant" 100 dimensions, i.e., those reflecting the 100 most frequent contexts.

3 Experiment 1: Phrase Similarity

Our first experiment focused on modeling similarity judgments for short phrases gathered in human experiments. Distributional representations of individual words are commonly evaluated on tasks based on their ability to model semantic similarity relations, e.g., synonymy or priming. Thus, it seems appropriate to evaluate phrase representations in a

	dim.	c.m.	Adj-N	N-N	V-Obj
SDS (BNC)	2000	+	0.37	0.38	0.28
	2000	$\odot$	**0.48**	**0.50**	**0.35**
	100	RAE	0.31	0.30	0.28
DM (3-BWC)	vary	+	0.37	0.30	0.29
	vary	$\odot$	0.21	0.37	0.33
	100	RAE	0.25	0.26	0.09
NLM (BNC)	50	+	0.28	0.26	0.24
	50	$\odot$	0.26	0.22	0.18
	100	RAE	0.19	0.24	0.28

Table 3: Correlation coefficients of model predictions with subject similarity ratings (Spearman's ρ); columns show dimensionality: fixed or varying (see Section 2.1), composition method: $+$ is additive vector composition, $\odot$ is component-wise multiplicative vector composition, RAE is Socher et al. (2011a)'s recursive auto-encoder.

similar manner. Specifically, we used the dataset from Mitchell and Lapata (2010) which contains similarity judgments for adjective-noun, noun-noun and verb-object phrases, respectively.[2] Each item is a phrase pair phr_1, phr_2 which has a human rating from 1 (very low similarity) to 7 (very high similarity).

Using the composition models described above, we compute the cosine similarity of phr_1 and phr_2:

$$phrSim_{phr_1,phr_2} = \frac{phrVec_{phr_1} \cdot phrVec_{phr_2}}{|phrVec_{phr_1}| \times |phrVec_{phr_2}|} \quad (17)$$

Model similarities were evaluated against the human similarity ratings using Spearman's ρ correlation coefficient.

Table 3 summarizes the performance of the various models on the phrase similarity dataset. Rows in the table correspond to different vector representations: the simple distributional semantic space (SDS) from Mitchell and Lapata (2010), Baroni and Lenci's (2010) distributional memory tensor (DM) and the neural language model (NLM), for each phrase combination: adjective noun (Adj-N), noun-noun (N-N) and verb object (V-Obj). For each phrase type we report results for each compositional model, namely additive ($+$), multiplicative ($\odot$) and recursive autoencoder (RAE). The table also shows the dimensionality of the input vectors next to the vector representation.

[2]The dataset is publicly available from `http://homepages.inf.ed.ac.uk/s0453356/share`

As can be seen, for SDS the best performing model is multiplication, as it is mostly for DM. With regard to NLM, vector addition yields overall better results. In general, neither DM or NLM in any compositional configuration are able to outperform SDS with multiplication. All models in Table 3 are significantly correlated with the human similarity judgments ($p < 0.01$). Spearman's ρ differences of 0.3 or more are significant at the 0.01 level, using a *t*-test (Cohen and Cohen, 1983).

4 Experiment 2: Paraphrase Detection

Although the phrase similarity task gives a fairly direct insight into semantic similarity and compositional representations, it is somewhat limited in scope as it only considers two-word constructions rather than naturally occurring sentences. Ideally, we would like to augment our evaluation with a task which is based on large quantities of natural data and for which vector composition has practical consequences. For these reasons, we used the Microsoft Research Paraphrase Corpus (MSRPC) introduced by Dolan et al. (2004). The corpus consists of sentence pairs Sen_{i_1}, Sen_{i_2} and labels indicating whether they are in a paraphrase relationship or not. The vector representations obtained from our various compositional models were used as features for the paraphrase classification task.

The MSRPC dataset contains 5,801 sentence pairs, we used the standard split of 4,076 training pairs (67.5% of which are paraphrases) and 1,725 test pairs (66.5% of which are paraphrases). In order to judge whether two sentences have the same meaning we employ Fan et al. (2008)'s liblinear classifier. For each of our three vector sources and three different compositional methods, we create the following features: (a) a vector representing the pair of input sentences either via concatenation ("con") or subtraction ("sub"); (b) a vector encoding which words appear therein ("enc"); and (c) a vector made up of the following four other pieces of information: the cosine similarity of the sentence vectors, the length of Sen_{i_1}, the length of Sen_{i_2}, and the unigram overlap among the two sentences.

In order to encode which words appear in

	NLM (BNC)	DM (3-BWC)	SDS (BNC)
+	69.04 (con, other)	**73.51** (other)	72.93 (other)
$\odot$	67.83 (sub, other)	67.54 (other)	**73.04** (other)
RAE	**70.26** (con, other)	68.29 (sub, other)	69.10 (con, other)

Table 4: Paraphrase classification accuracy in %. Included features are in parentheses: "con" is sentence vector concatenation, "sub" is sentence vector subtraction, "other" stands for 4 other features (see Section 4)

	NLM (BNC)	DM (3-BWC)	SDS (BNC)
+	81.00 (con, other)	**82.16** (other)	80.76 (other)
$\odot$	80.41 (sub, other)	80.18 (other)	**82.33** (other)
RAE	**81.28** (con, other)	80.43 (sub, other)	80.68 (con, other)

Table 5: Paraphrase classification F1-score in %. The involved features are exactly the same as in Table 4.

each sentence and how often, we define a vector *wdCount*$_i$ for sentence *Sen*$_i$ and enumerate all words occuring in the MSRPC:

$$Voc_{MSRPC} = \{w_1^{(MSRPC)}, ..., w_{n_{MSRPC}}^{(MSRPC)}\} \qquad (18)$$

giving the word count vectors n_{MSRPC} dimensions. Thus the k-th component of *wdCount*$_i$ is the frequency with which the word $w_k^{(MSRPC)}$ appears in $Sen_i = (w_1^{(i)}, ..., w_{n_i}^{(i)})$:

$$wdCount_i[k] = |\{j \in [1; n_i] \,|\, w_k^{(MSRPC)} = w_j^{(i)}\}| \quad (19)$$

for $k = 1, ..., n_{MSRPC}$. Even though n_{MSRPC} may be large, the computer files storing our feature vectors do not explode in size because *wdCount* contains many zeros and the classifier allows a sparse notation of (non-zero) feature values.

Regarding the last four features, we measured the similarity between sentences the same way as we did with phrases in section 3.

$$senSim_{i_1,i_2} = \frac{senVec_{i_1} \cdot senVec_{i_2}}{|senVec_{i_1}| \times |senVec_{i_2}|} \qquad (20)$$

Note that this is the cosine of the angle between $senVec_{i_1}$ and $senVec_{i_2}$. This enables us to observe the similarity or dissimilarity of two sentences independent of their sentence length. Even though each contained word increases or decreases the norm of the resulting sentence vector, this does not distort the overall similarity value, due to normalization.

The lengths of Sen_{i_1} and Sen_{i_2} are simply the number of words they contain. The unigram overlap feature value may be viewed as the cardinality of the intersection of each sentence's multiset-bag-of-words. The latter is encoded in the already-introduced *wdCount* vectors. Therefore,

$$uniOverlap_{i_1,i_2} = \sum_{k=1}^{n_{MSRPC}} \min_{s=1,2} \{wdCount_{i_s}[k]\} \qquad (21)$$

In order to establish which features work best for each representation and composition method, we exhaustively explored all combinations on a development set (20% of the original MSRPC training set). Tables 4 (accuracy) and 5 (F1) show our results on the test set with the best feature combinations for each model (shown in parentheses). Each row corresponds to a different type of composition and each column to a different word representation model.

As can be seen, the distributional memory (DM) is the best performing representation for the additive composition model. The neural language model (NLM) gives best results for the recursive autoencoder (RAE), although the other two representations come close. And finally the simple distributional semantic space (SDS) works best with multiplication. Also note that the best performing models, namely DM with addition and SDS with multiplication, use a basic feature space consisting only of the cosine similarity of the composed sentence vectors, the length of the two sentences involved, and their unigram word overlap.

Although our intention was to use the paraphrase detection task as a test-bed for evaluating compositional models rather than achieving state-of-the-art results, Table 6 compares our approach against previous work on the same task and dataset. Initial research concentrated on individual words rather than sentential representations. Several approaches used

Model	Acc.	F1
Baseline	66.5	79.9
Mihalcea et al. (2006)	70.3	81.3
Rus et al. (2008)	70.6	80.5
Qiu et al. (2006)	72.0	81.6
Islam and Inkpen (2007)	72.6	81.3
Mitchell and Lapata (2010) ($\odot$)	**73.0**	**82.3**
Baroni and Lenci (2010) ($+$)	**73.5**	**82.2**
Fernando and Stevenson (2008)	74.1	82.4
Wan et al. (2006)	75.6	83.0
Das and Smith (2009)	76.1	82.7
Socher et al. (2011a)	76.8	83.6

Table 6: Overview of results on the MSRCP (test corpus). Accuracy differences of 3.3 or more are significant at the 0.01 level (using the χ^2 statistic).

WordNet in conjunction with distributional similarity in an attempt to detect meaning conveyed by synonymous words (Islam and Inkpen, 2007; Mihalcea et al., 2006; Fernando and Stevenson, 2008). More recently, the addition of syntactic features based on dependency parse trees (Wan et al., 2006; Das and Smith, 2009) has been shown to substantially boost performance. The model of Das and Smith (2009), for example, uses quasi-synchronous dependency grammar to model the structure of the sentences involved in the comparison and their correspondences. Socher et al. (2011a) obtain an accuracy that is higher than previously published results. This model is more sophisticated than the one we used in our experiments (see Table 4 and 5). Rather than using the output of the RAE as features for the classifier, it applies dynamic pooling, a procedure that takes a similarity matrix as input (e.g., created by sentences with differing lengths) and maps it to a matrix of fixed size that represents more faithfully the global similarity structure.[3]

Overall, we observe that our own models do as well as some of the models that employ WordNet and more sophisticated syntactic features. With regard to F1, we are comparable with Das and Smith (2009) and Socher et al. (2011a) without using elaborate features, or any additional manipulations over and above the output of the composition functions which if added could increase performance.

[3]Without dynamic pooling, their model yields an accuracy of 74.2.

5 Discussion

In this paper we systematically compared three types of distributional representation and their effect on semantic composition. Our comparisons involved a simple distributional semantic space (Mitchell and Lapata, 2010), word embeddings computed with a neural language model (Collobert and Weston, 2008) and a representation based on weighted word-link-word tuples arranged into a third-order tensor (Baroni and Lenci, 2010). These representations vary in many respects: the amount of preprocessing and linguistic information involved (the third-order tensor computes semantic representations over parsed corpora), whether the semantic space is the by-product of a learning process (in the neural language model the parameters of the lookup table must be learned), and data requirements (the third-order tensor involves processing billions of words). These representations served as input to three composition methods involving addition, multiplication and a deep recursive autoencoder. Again these methods differ in terms of how they implement compositionality: addition and multiplication are commutative and associative operations and thus ignore word order and, more generally, syntactic structure. In contrast, the recursive autoencoder is syntax-aware as it operates over a parse tree. However, the composed representations must be learned with a neural network.

We evaluated nine models on the complementary tasks of phrase similarity and paraphrase detection. The former task simplifies the challenge of finding an adequate method of composition and places more emphasis on the representation, whereas the latter poses, in a sense, the ultimate challenge for composition models. It involves entire sentences exhibiting varied syntactic constructions and in the limit involves genuine natural language undertanding. Across both tasks our results deliver a consistent message: simple is best. Despite being in theory more expressive, the representations obtained by the neural language model and the third-order tensor cannot match the simple semantic space on the phrase similarity task. In this task syntax-oblivious composition models are superior to the more sophisticated recursive autoencoder. The latter performs better on the paraphrase detection task when its output is fed to a classifier. The simple semantic space may not take word order or sentence structure into account, but nevertheless achieves considerable semantic expressivity: it is on par with the third-order tensor without having access to as much data (3 billion words) or a syntactically parsed corpus.

What do these findings tell us about the future of compositional models for distributional semantics? The problem of finding the right methods of vector composition cannot be pursued independent of the choice of lexical representation. Having tested many model combinations, we argue that in a good model of distributive semantics representation and composition must go hand in hand, i.e., they must be mutually learned.

Acknowledgments We are grateful to Jeff Mitchell for his help with the re-implementation of his models. Thanks to Frank Keller and Micha Elsner for their input on earlier versions of this work and to Richard Socher for technical assistance. We acknowledge the support of EPSRC through project grant EP/I032916/1.

References

Marco Baroni and Alessandro Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4):673–721.

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1193, Cambridge, MA.

Jerome R. Bellegarda. 2000. Exploiting latent semantic information in statistical language modeling. *Proceedings of the Institute of of Electrical and Electronics Engineers*, 88(8):1279–1296.

Yoshua Bengio. 2001. Neural net language models. *Scholarpedia*, 3(1):3881.

Stephen Clark, Bob Coecke, and Mehrnoosh Sadrzadeh. 2008. A compositional distributional model of meaning. In *Proceedings of the 2nd Quantum Interaction Symposium*, pages 133–140, Oxford, UK.

Daoud Clarke. 2012. A context-theoretic framework for compositionality in distributional semantics. *Computational Linguistics*, 38(1):41–71.

J Cohen and P Cohen. 1983. *Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences*. Hillsdale, NJ: Erlbaum.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, pages 160–167, New York, NY. ACM.

Dipanjan Das and Noah A. Smith. 2009. Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 468–476, Suntec, Singapore.

G. Denhire and B. Lemaire. 2004. A computational model of children's semantic memory. In *Proceedings of the 26th Annual Meeting of the Cognitive Science Society*, pages 297–302, Mahwah, NJ. Lawrence Erlbaum Associates.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 350–356, Geneva, Switzerland. COLING.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Samuel Fernando and Mark Stevenson. 2008. A semantic similarity approach to paraphrase detection. *Technology*.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: the concept revisited. In *WWW*, pages 406–414.

Peter Foltz, Walter Kintsch, and Thomas Landauer. 1998. The measurement of textual coherence with latent semantic analysis. *Discourse Process*, 15:285–307.

Edward Grefenstette and Mehrnoosh Sadrzadeh. 2011a. Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1394–1404, Edinburgh, Scotland.

Edward Grefenstette and Mehrnoosh Sadrzadeh. 2011b. Experimenting with transitive verbs in a DisCoCat. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, pages 62–66, Edinburgh, UK.

Gregory Grefenstette. 1994. *Explorations in Automatic Thesaurus Discovery*. Kluwer Academic Publishers, Norwell, MA.

Thomas L. Griffiths, Mark Steyvers, and Joshua B. Tenenbaum. 2007. Topics in semantic representation. *Psychological Review*, 114(2):211–244.

Aminul Islam and Diana Inkpen. 2007. Semantic similarity of short texts. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing*, Borovets, Bulgaria.

Walter Kintsch. 2001. Predication. *Cognitive Science*, 25(2):173–202.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430, Sapporo, Japan.

T. K. Landauer and S. T. Dumais. 1997. A solution to Plato's problem: the latent semantic analysis theory of acquisition, induction and representation of knowledge. *Psychological Review*, 104(2):211–240.

Scott McDonald. 2000. *Environmental Determinants of Lexical Processing Effort*. Ph.D. thesis, University of Edinburgh.

Rada Mihalcea, Courtney Corley, and Carlo Strapparava, 2006. *Corpus-based and knowledge-based measures of text semantic similarity*, pages 775–780. AAAI Press.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive Science*, 38(8):1388–1429.

Jeff Mitchell. 2011. *Composition in distributional models of semantics*. Ph.D. thesis, University of Edinburgh.

Tony A. Plate. 1995. Holographic reduced representations. *IEEE Transactions on Neural Networks*, 6(3):623–641.

Long Qiu, Min-Yen Kan, and Tat-Seng Chua. 2006. Paraphrase recognition via dissimilarity significance classification. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 18–26, Sydney, Australia.

Vasile Rus, Philip M. McCarthy, Mihai C. Lintean, Danielle S. McNamara, and Arthur C. Graesser. 2008. Paraphrase identification with lexico-syntactic graph subsumption. In David Wilson and H. Chad Lane, editors, *Florida Artificial Intelligence Research Society Conference*, pages 201–206. AAAI Press.

Hinrich Schütze. 1998. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–124.

Paul Smolensky. 1990. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, 46:159–216.

Richard Socher, Eric H. Huang, Jeffrey Pennin, 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*, pages 801–809. Granada, Spain.

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 the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 151–161, Edinburgh, Scotland.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394, Uppsala, Sweden.

Peter D. Turney. 2006. Similarity of semantic relations. *Computational Linguistics*, 32(3):379–416.

Stephen Wan, Mark Dras, Robert Dale, and Cecile Paris. 2006. Using dependency-based features to take the "para-farce" out of paraphrase. In *Proceedings of the Australasian Language Technology Workshop 2006*, pages 131–138, Sydney, Australia.