Concrete Models and Empirical Evaluations for the Categorical Compositional Distributional Model of Meaning

Edward Grefenstette*†
Google DeepMind

Mehrnoosh Sadrzadeh**†
Queen Mary University of London

Modeling compositional meaning for sentences using empirical distributional methods has been a challenge for computational linguists. The categorical model of Clark, Coecke, and Sadrzadeh (2008) and Coecke, Sadrzadeh, and Clark (2010) provides a solution by unifying a categorial grammar and a distributional model of meaning. It takes into account syntactic relations during semantic vector composition operations. But the setting is abstract: It has not been evaluated on empirical data and applied to any language tasks. We generate concrete models for this setting by developing algorithms to construct tensors and linear maps and instantiate the abstract parameters using empirical data. We then evaluate our concrete models against several experiments, both existing and new, based on measuring how well models align with human judgments in a paraphrase detection task. Our results show the implementation of this general abstract framework to perform on par with or outperform other leading models in these experiments.[1]

1. Introduction

The distributional approach to the semantic modeling of natural language, inspired by the notion—presented by Firth (1957) and Harris (1968)—that the meaning of a word is tightly related to its context of use, has grown in popularity as a method of semantic representation. It draws from the frequent use of vector-based document models in information retrieval, modeling the meaning of words as vectors based on the distribution of co-occurring terms within the context of a word.

Using various vector similarity metrics as a measure of semantic similarity, these distributional semantic models (DSMs) are used for a variety of NLP tasks, from

* DeepMind Technologies Ltd, 5 New Street Square, London EC4A 3 TW.
E-mail: etg@google.com.

** School of Electronic Engineering and Computer Science, Queen Mary University of London, Mile End Road, London E1 4NS, United Kingdom. E-mail: mehrnoosh.sadrzadeh@qmul.ac.uk.

† The work described in this article was performed while the authors were at the University of Oxford.

[1] Support from EPSRC grant EP/J002607/1 is acknowledged.

Submission received: 26 September 2012; revised submission received: 31 October 2013;
accepted for publication: 5 April 2014.

doi:10.1162/COLI_a_00209

automated thesaurus building (Grefenstette 1994; Curran 2004) to automated essay marking (Landauer and Dumais 1997). The broader connection to information retrieval and its applications is also discussed by Manning, Raghavan, and Schütze (2011). The success of DSMs in essentially word-based tasks such as thesaurus extraction and construction (Grefenstette 1994; Curran 2004) invites an investigation into how DSMs can be applied to NLP and information retrieval (IR) tasks revolving around larger units of text, using semantic representations for phrases, sentences, or documents, constructed from lemma vectors. However, the problem of compositionality in DSMs—of how to go from word to sentence and beyond—has proved to be non-trivial.

A new framework, which we refer to as *DisCoCat*, initially presented in Clark, Coecke, and Sadrzadeh (2008) and Coecke, Sadrzadeh, and Clark (2010) reconciles distributional approaches to natural language semantics with the structured, logical nature of formal semantic models. This framework is abstract; its theoretical predictions have not been evaluated on real data, and its applications to empirical natural language processing tasks have not been studied.

This article is the journal version of Grefenstette and Sadrzadeh (2011a, 2011b), which fill this gap in the DisCoCat literature; in it, we develop a concrete model and an unsupervised learning algorithm to instantiate the abstract vectors, linear maps, and vector spaces of the theoretical framework; we develop a series of empirical natural language processing experiments and data sets and implement our algorithm on large scale real data; we analyze the outputs of the algorithm in terms of linear algebraic equations; and we evaluate the model on these experiments and compare the results with other competing unsupervised models. Furthermore, we provide a linear algebraic analysis of the algorithm of Grefenstette and Sadrzadeh (2011a) and present an in-depth study of the better performance of the method of Grefenstette and Sadrzadeh (2011b).

We begin in Section 2 by presenting the background to the task of developing compositional distributional models. We briefly introduce two approaches to semantic modeling: formal semantic models and distributional semantic models. We discuss their differences, and relative advantages and disadvantages. We then present and critique various approaches to bridging the gap between these models, and their limitations. In Section 3, we summarize the categorical compositional distributional framework of Clark, Coecke, and Sadrzadeh (2008) and Coecke, Sadrzadeh, and Clark (2010) and provide the theoretical background necessary to understand it; we also sketch the road map of the literature leading to the development of this setting and outline the contributions of this paper to the field. In Section 4, we present the details of an implementation of this framework, and introduce learning algorithms used to build semantic representations in this implementation. In Section 5, we present a series of experiments designed to evaluate this implementation against other unsupervised distributional compositional models. Finally, in Section 6 we discuss these results, and posit future directions for this research area.

2. Background

Compositional formal semantic models represent words as parts of logical expressions, and compose them according to grammatical structure. They stem from classical ideas in logic and philosophy of language, mainly Frege's principle that the meaning of a sentence is a function of the meaning of its parts (Frege 1892). These models relate to well-known and robust logical formalisms, hence offering a scalable theory of meaning that can be used to reason about language using logical tools of proof and inference. Distributional models are a more recent approach to semantic modeling, representing the meaning of words as vectors learned empirically from corpora. They have found their way into real-world applications such as thesaurus extraction (Grefenstette 1994; Curran 2004) or automated essay marking (Landauer and Dumais 1997), and have connections to semantically motivated information retrieval (Manning, Raghavan, and Schütze 2011). This two-sortedness of defining properties of meaning: "logical form" versus "contextual use," has left the quest for "what is the foundational structure of meaning?"—a question initially the concern of solely linguists and philosophers of language—even more of a challenge.

In this section, we present a short overview of the background to the work developed in this article by briefly describing formal and distributional approaches to natural language semantics, and providing a non-exhaustive list of some approaches to compositional distributional semantics. For a more complete review of the topic, we encourage the reader to consult Turney (2012) or Clark (2013).

2.1 Montague Semantics

Formal semantic models provide methods for translating sentences of natural language into logical formulae, which can then be fed to computer-aided automation tools to reason about them (Alshawi 1992).

To compute the meaning of a sentence consisting of n words, meanings of these words must *interact* with one another. In formal semantics, this further interaction is represented as a function derived from the grammatical structure of the sentence. Such models consist of a pairing of syntactic analysis rules (in the form of a grammar) with semantic interpretation rules, as exemplified by the simple model presented on the left of Figure 1.

The semantic representations of words are lambda expressions over parts of logical formulae, which can be combined with one another to form well-formed logical expressions. The function $|-|: \mathcal{L} \to \mathcal{M}$ maps elements of the lexicon $\mathcal{L}$ to their interpretation (e.g., predicates, relations, domain objects) in the logical model $\mathcal{M}$ used. Nouns are typically just logical atoms, whereas adjectives and verbs and other relational words are interpreted as predicates and relations. The parse of a sentence such as "cats like milk," represented here as a binarized parse tree, is used to produce its semantic interpretation by substituting semantic representations for their grammatical constituents and applying β-reduction where needed. Such a derivation is shown on the right of Figure 1.

What makes this class of models attractive is that it reduces language meaning to logical expressions, a subject well studied by philosophers of language, logicians, and linguists. Its properties are well known, and it becomes simple to evaluate the meaning of a sentence if given a logical model and domain, as well as verify whether or not one sentence entails another according to the rules of logical consequence and deduction.

Syntactic Analysis	Semantic Interpretation
S → NP VP	$\|VP\|(\|NP\|)$
NP → cats, milk, etc.	$\|\text{cats}\|, \|\text{milk}\|, \ldots$
VP → Vt NP	$\|Vt\|(\|NP\|)$
Vt → like, hug, etc.	$\lambda yx.\|\text{like}\|(x, y), \ldots$

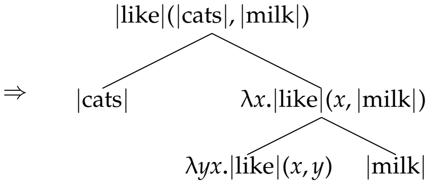

Figure 1
A simple model of formal semantics.

However, such logical analysis says nothing about the closeness in meaning or topic of expressions beyond their truth-conditions and which models satisfy these truth conditions. Hence, formal semantic approaches to modeling language meaning do not perform well on language tasks where the notion of similarity is not strictly based on truth conditions, such as document retrieval, topic classification, and so forth. Furthermore, an underlying domain of objects and a valuation function must be provided, as with any logic, leaving open the question of how we might *learn* the meaning of language using such a model, rather than just use it.

2.2 Distributional Semantics

A popular way of representing the meaning of words in lexical semantics is as distributions in a high-dimensional vector space. This approach is based on the *distributional hypothesis* of Harris (1968), who postulated that the meaning of a word was dictated by the context of its use. The more famous dictum stating this hypothesis is the statement of Firth (1957) that "You shall know a word by the company it keeps." This view of semantics has furthermore been associated (Grefenstette 2009; Turney and Pantel 2010) with earlier work in philosophy of language by Wittgenstein (presented in Wittgenstein 1953), who stated that language meaning was equivalent to its real world use.

Practically speaking, the meaning of a word can be learned from a corpus by looking at what other words occur with it within a certain *context*, and the resulting distribution can be represented as a vector in a semantic vector space. This vectorial representation is convenient because vectors are a familiar structure with a rich set of ways of computing vector distance, allowing us to experiment with different word similarity metrics. The geometric nature of this representation entails that we can not only compare individual words' meanings with various levels of granularity (e.g., we might, for example, be able to show that cats are closer to kittens than to dogs, but that all three are mutually closer than cats and steam engines), but also apply methods frequently called upon in IR tasks, such as those described by Manning, Raghavan, and Schütze (2011), to group concepts by topic, sentiment, and so on.

The distribution underlying word meaning is a vector in a vector space, the basis vectors of which are dictated by the context. In simple models, the basis vectors will be annotated with words from the lexicon. Traditionally, the vector spaces used in such models are Hilbert spaces (i.e., vector spaces with orthogonal bases, such that the inner product of any one basis vector with another [other than itself] is zero). The semantic vector for any word can be represented as the weighted sum of the basis vectors:

$$\overrightarrow{\text{some word}} = \sum_i c_i \overrightarrow{n_i}$$

where $\{\overrightarrow{n_i}\}_i$ is the basis of the vector space the meaning of the word lives in, and $c_i \in \mathbb{R}$ is the weight associated with basis vector $\overrightarrow{n_i}$.

The construction of the vector for a word is done by counting, for each lexicon word n_i associated with basis vector $\overrightarrow{n_i}$, how many times it occurs in the context of each occurrence of the word for which we are constructing the vector. This count is then typically adjusted according to a weighting scheme (e.g., TF-IDF). The "context" of a word can be something as simple as the other words occurring in the same sentence as the word or within k words of it, or something more complex, such as using dependency relations (Padó and Lapata 2007) or other syntactic features.

Commonly, the similarity of two semantic vectors is computed by taking their cosine measure, which is the sum of the product of the basis weights of the vectors:

$$cosine(\overrightarrow{a}, \overrightarrow{b}) = \frac{\sum_i c_i^a c_i^b}{\sqrt{\sum_i (c_i^a)^2 \sum_i (c_i^b)^2}}$$

where c_i^a and c_i^b are the basis weights for $\overrightarrow{a}$ and $\overrightarrow{b}$, respectively. However, other options may be a better fit for certain implementations, typically dependent on the weighting scheme.

Readers interested in learning more about these aspects of distributional lexical semantics are invited to consult Curran (2004), which contains an extensive overview of implementation options for distributional models of word meaning.

2.3 Compositionality and Vector Space Models

In the previous overview of distributional semantic models of lexical semantics, we have seen that DSMs are a rich and tractable way of learning word meaning from a corpus, and obtaining a measure of semantic similarity of words or groups of words. However, it should be fairly obvious that the same method cannot be applied to sentences, whereby the meaning of a sentence would be given by the distribution of other sentences with which it occurs.

First and foremost, a sentence typically occurs only once in a corpus, and hence substantial and informative distributions cannot be created in this manner. More importantly, human ability to understand new sentences is a compositional mechanism: We understand sentences we have never seen before because we can generate sentence meaning from the words used, and how they are *put into relation*. To go from word vectors to sentence vectors, we must provide a *composition operation* allowing us to construct a sentence vector from a collection of word vectors. In this section, we will discuss several approaches to solving this problem, their advantages, and their limitations.

2.3.1 Additive Models. The simplest composition operation that comes to mind is straightforward vector addition, such that:

$$\overrightarrow{ab} = \overrightarrow{a} + \overrightarrow{b}$$

Conceptually speaking, if we view word vectors as semantic information distributed across a set of properties associated with basis vectors, using vector addition as a semantic composition operation states that the information of a set of lemmas in a sentence is simply the sum of the information of the individual lemmas. Although crude, this approach is computationally cheap, and appears sufficient for certain NLP tasks: Landauer and Dumais (1997) show it to be sufficient for automated essay marking tasks, and Grefenstette (2009) shows it to perform better than a collection of other simple similarity metrics for summarization, sentence paraphrase, and document paraphrase detection tasks.

However, there are two principal objections to additive models of composition: first, vector addition is commutative, therefore, $\overrightarrow{\text{John drank wine}} = \overrightarrow{\text{John}} + \overrightarrow{\text{drank}} + \overrightarrow{\text{wine}} =$

$\overrightarrow{\text{Wine drank John}}$, and thus vector addition ignores syntactic structure completely; and second, vector addition sums the information contained in the vectors, effectively jumbling the meaning of words together as sentence length grows.

The first objection is problematic, as the syntactic insensitivity of additive models leads them to equate the representation of sentences with patently different meanings. Mitchell and Lapata (2008) propose to add some degree of syntactic sensitivity—namely, accounting for word order—by weighting word vectors according to their order of appearance in a sentence as follows:

$$\overrightarrow{ab} = \alpha\overrightarrow{a} + \beta\overrightarrow{b}$$

where $\alpha, \beta \in \mathbb{R}$. Consequently, $\overrightarrow{\text{John drank wine}} = \alpha \cdot \overrightarrow{\text{John}} + \beta \cdot \overrightarrow{\text{drank}} + \gamma \cdot \overrightarrow{\text{wine}}$ would not have the same representation as $\overrightarrow{\text{Wine drank John}} = \alpha \cdot \overrightarrow{\text{wine}} + \beta \cdot \overrightarrow{\text{drank}} + \gamma \cdot \overrightarrow{\text{John}}$.

The question of how to obtain weights and whether they are only used to reflect word order or can be extended to cover more subtle syntactic information is open, but it is not immediately clear how such weights may be obtained empirically and whether this mode of composition scales well with sentence length and increase in syntactic complexity. Guevara (2010) suggests using machine-learning methods such as partial least squares regression to determine the weights empirically, but states that this approach enjoys little success beyond minor composition such as adjective-noun or noun-verb composition, and that there is a dearth of metrics by which to evaluate such machine learning–based systems, stunting their growth and development.

The second objection states that vector addition leads to increase in ambiguity as we construct sentences, rather than decrease in ambiguity as we would expect from giving words a context; and for this reason Mitchell and Lapata (2008) suggest replacing additive models with multiplicative models as discussed in Section 2.3.2, or combining them with multiplicative models to form mixture models as discussed in Section 2.3.3.

2.3.2 Multiplicative Models. The multiplicative model of Mitchell and Lapata (2008) is an attempt to solve the ambiguity problem discussed in Section 2.3.1 and provide implicit disambiguation during composition. The composition operation proposed is the component-wise multiplication ($\odot$) of two vectors: Vectors are expressed as the weighted sum of their basis vectors, and the weight of the basis vectors of the composed vector is the product of the weights of the original vectors; for $\overrightarrow{a} = \sum_i c_i \overrightarrow{n_i}$, and $\overrightarrow{b} = \sum_i c'_i \overrightarrow{n_i}$, we have

$$\overrightarrow{ab} = \overrightarrow{a} \odot \overrightarrow{b} = \sum_i c_i c'_i \overrightarrow{n_i}$$

Such multiplicative models are shown by Mitchell and Lapata (2008) to perform better at verb disambiguation tasks than additive models for noun-verb composition, against a baseline set by the original verb vectors. The experiment they use to show this will also serve to evaluate our own models, and form the basis for further experiments, as discussed in Section 5.

This approach to compositionality still suffers from two conceptual problems: First, component-wise multiplication is still commutative and hence word order is not accounted for; second, rather than "diluting" information during large compositions and creating ambiguity, it may remove too much through the "filtering" effect of component-wise multiplication.

The first problem is more difficult to deal with for multiplicative models than for additive models, because both scalar multiplication and component-wise multiplication are commutative and hence $\alpha\overrightarrow{a} \odot \beta\overrightarrow{b} = \alpha\overrightarrow{b} \odot \beta\overrightarrow{a}$ and thus word order cannot be taken into account using scalar weights.

To illustrate how the second problem entails that multiplicative models do not scale well with sentence length, we look at the structure of component-wise multiplication again: $\overrightarrow{a} \odot \overrightarrow{b} = \sum_i c_i c'_i \overrightarrow{n_i}$. For any i, if $c_i = 0$ or $c'_i = 0$, then $c_i c'_i = 0$, and therefore for any composition, the number of non-zero basis weights of the produced vector is less than or equal to the number of non-zero basis weights of the original vectors: At each composition step information is filtered out (or preserved, but never increased). Hence, as the number of vectors to be composed grows, the number of non-zero basis weights of the product vector stays the same or—more realistically—decreases. Therefore, for any composition of the form $\overrightarrow{a_1 \ldots a_i \ldots a_n} = \overrightarrow{a_1} \odot \ldots \odot \overrightarrow{a_i} \odot \ldots \odot \overrightarrow{a_n}$, if for any i, $\overrightarrow{a_i}$ is orthogonal to $\overrightarrow{a_1 \ldots a_{i-1}}$ then $\overrightarrow{a_1 \ldots a_i \ldots a_n} = \overrightarrow{0}$. It follows that purely multiplicative models alone are not apt as a single mode of composition beyond noun-verb (or adjective-noun) composition operations.

One solution to this second problem not discussed by Mitchell and Lapata (2008) would be to introduce some smoothing factor $s \in \mathbb{R}^+$ for point-wise multiplication such that $\overrightarrow{a} \odot \overrightarrow{b} = \sum_i (c_i + s)(c'_i + s)\overrightarrow{n_i}$, ensuring that information is never completely filtered out. Seeing how the problem of syntactic insensitivity still stands in the way of full-blown compositionality for multiplicative models, we leave it to those interested in salvaging purely multiplicative models to determine whether some suitable value of s can be determined.

2.3.3 Mixture Models. The problems faced by multiplicative models presented in Section 2.3.2 are acknowledged in passing by Mitchell and Lapata (2008), who propose mixing additive and multiplicative models in the hope of leveraging the advantage of each while doing away with their pitfalls. This is simply expressed as the weighted sum of additive and multiplicative models:

$$\overrightarrow{ab} = \alpha\overrightarrow{a} + \beta\overrightarrow{b} + \gamma(\overrightarrow{a} \odot \overrightarrow{b})$$

where α, β, and γ are predetermined scalar weights.

The problems for these models are threefold. First, the question of how scalar weights are to be obtained still needs to be determined. Mitchell and Lapata (2008) concede that one advantage of purely multiplicative models over weighted additive or mixture models is that the lack of scalar weights removes the need to optimize the scalar weights for particular tasks (at the cost of not accounting for syntactic structure), and avoids the methodological concerns accompanying this requirement.

Second, the question of how well this process scales from noun-verb composition to more syntactically rich expressions must be addressed. Using scalar weights to account for word order seems ad hoc and superficial, as there is more to syntactic structure than the mere ordering of words. Therefore, an account of how to build sentence vectors for sentences such as *The dog bit the man* and *The man was bitten by the dog* in order to give both sentences the same (or a similar) representation would need to give a richer role to scalar weights than mere token order. Perhaps specific weights could be given to particular syntactic classes (such as nouns) to introduce a more complex syntactic element into vector composition, but it is clear that this alone is not a solution, as the weight for nouns *dog* and *man* would be the same, allowing for the same commutative degeneracy observed in non-weighted additive models, in which $\overrightarrow{\text{the dog bit the man}} = \overrightarrow{\text{the man bit the dog}}$. Introducing a mixture of weighting systems accounting for both word order and syntactic roles may be a solution, but it is not only ad hoc but also arguably only partially reflects the syntactic structure of the sentence.

The third problem is that Mitchell and Lapata (2008) show that in practice, although mixture models perform better at verb disambiguation tasks than additive models and weighted additive models, they perform equivalently to purely multiplicative models with the added burden of requiring parametric optimization of the scalar weights.

Therefore, whereas mixture models aim to take the best of additive and multiplicative models while avoiding their problems, they are only partly successful in achieving the latter goal, and demonstrably do little better in achieving the former.

2.3.4 Tensor-Based Models. From Sections 2.3.1–2.3.3 we observe that the need for incorporating syntactic information into DSMs to achieve true compositionality is pressing, if only to develop a non-commutative composition operation that can take into account word order without the need for adhoc weighting schemes, and hopefully richer syntactic information as well.

An early proposal by Smolensky and colleagues (Smolensky 1990; Smolensky and Legendre 2006) to use linear algebraic tensors as a composition operation solves the problem of finding non-commutative vector composition operators. The composition of two vectors is their tensor product, sometimes called the kronecker product when applied to vectors rather than vector spaces. For $\overrightarrow{a} \in V = \sum_i c_i \overrightarrow{n_i}$, and $\overrightarrow{b} \in W = \sum_j c'_j \overrightarrow{n'_j}$, we have:

$$\overrightarrow{ab} = \overrightarrow{a} \otimes \overrightarrow{b} = \sum_{ij} c_i c'_j \, \overrightarrow{n_i} \otimes \overrightarrow{n'_j}$$

The composition operation takes the original vectors and maps them to a vector in a larger vector space $V \otimes W$, which is the tensor space of the original vectors' spaces. Here the second instance of $\otimes$ is not a recursive application of the kronecker product, but rather the pairing of basis elements of V and W to form a basis element of $V \otimes W$. The shared notation and occasional conflation of kronecker and tensor products may seem confusing, but is fairly standard in multilinear algebra.

The advantage of this approach is twofold: First, vectors for different words need not live in the same spaces but can be composed nonetheless. This allows us to represent vectors for different word classes (topics, syntactic roles, etc.) in different spaces with different bases, which was not possible under additive or multiplicative models. Second, because the product vector lives in a larger space, we obtain the intuitive notion that the information of the whole is richer and more complex than the information of the parts.

Dimensionality Problems. However, as observed and commented upon by Smolensky himself, this increase in dimensionality brings two rather large problems for tensor based models. The first is computational: The size of the product vector space is the product of the size of the original vector spaces. If we assume that all words live in the same space N of dimensionality $dim(N)$, then the dimensionality of an n-word sentence vector is $dim(N)^n$. If we have as many basis vectors for our word semantic space as there are lexemes in our vocabulary (e.g., approximately 170k in English[2]), then the size of our sentence vectors quickly reaches magnitudes for which vector comparison (or even storage) are computationally intractable.[3] Even if, as most DSM implementations do, we restrict the basis vectors of word semantic spaces to the k (e.g., $k = 2{,}000$) most frequent words in a corpus, the sentence vector size still grows exponentially with sentence length, and the implementation problems remain.

The second problem is mathematical: Sentences of different length live in different spaces, and if we assign different vector spaces to different word types (e.g., syntactic classes), then sentences of different syntactic structure live in different vector spaces, and hence cannot be compared directly using inner product or cosine measures, leaving us with no obvious mode of semantic comparison for sentence vectors. If any model wishes to use tensor products in composition operations, it must find some way of reducing the dimensionality of product vectors to some common vector space so that they may be directly compared.

One notable method by which these dimensionality problems can be solved in general are the holographic reduced representations proposed by Plate (1991). The product vector of two vectors is projected into a space of smaller dimensionality by circular convolution to produce a trace vector. The circular correlation of the trace vector and one of the original vectors produces a noisy version of the other original vector. The noisy vector can be used to recover the clean original vector by comparing it with a predefined set of candidates (e.g., the set of word vectors if our original vectors are word meanings). Traces can be summed to form new traces effectively containing several vector pairs from which original vectors can be recovered. Using this encoding/decoding mechanism, the tensor product of sets of vectors can be encoded in a space of smaller dimensionality, and then recovered for computation without ever having to fully represent or store the full tensor product, as discussed by Widdows (2008).

There are problems with this approach that make it unsuitable for our purposes, some of which are discussed in Mitchell and Lapata (2010). First, there is a limit to the information that can be stored in traces, which is independent of the size of the vectors stored, but is a logarithmic function of their number. As we wish to be able to store information for sentences of variable word length without having to directly represent the tensored sentence vector, setting an upper bound to the number of vectors that can be composed in this manner limits the length of the sentences we can represent compositionally using this method.

Second, and perhaps more importantly, there are restrictions on the nature of the vectors that can be encoded in such a way: The vectors must be independently distributed such that the mean Euclidean length of each vector is 1. Such conditions are unlikely to be met in word semantic vectors obtained from a corpus; and as the failure to do so affects the system's ability to recover clean vectors, holographic reduced representations are not *prima facie* usable for compositional DSMs, although it is important to note that Widdows (2008) considers possible application areas where they may be of use, although once again these mostly involve noun-verb and adjective-noun compositionality rather than full blown sentence vector construction. We retain

[2] Source: `http://www.oxforddictionaries.com/page/howmanywords`.

[3] At four bytes per integer, and one integer per basis vector weight, the vector for *John loves Mary* would require roughly $(170{,}000 \cdot 4)^3 \approx 280$ petabytes of storage, which is over ten times the data Google processes on a daily basis according to Dean and Ghemawat (2008).

from Plate (1991) the importance of finding methods by which to project the tensored sentence vectors into a common space for direct comparison, as will be discussed further in Section 3.

Syntactic Expressivity. An additional problem of a more conceptual nature is that using the tensor product as a composition operation simply preserves word order. As we discussed in Section 2.3.3, this is not enough on its own to model sentence meaning. We need to have some means by which to incorporate syntactic analysis into composition operations.

Early work on including syntactic sensitivity into DSMs by Grefenstette (1992) suggests using crude syntactic relations to determine the frame in which the distributions for word vectors are collected from the corpus, thereby embedding syntactic information into the word vectors. This idea was already present in the work of Smolensky, who used sums of pairs of vector representations and their roles, obtained by taking their tensor products, to obtain a vector representation for a compound. The application of these ideas to DSMs was studied by Clark and Pulman (2007), who suggest instantiating the roles to dependency relations and using the distributional representations of words as the vectors. For example, in the sentence *Simon loves red wine*, *Simon* is the subject of *loves*, *wine* is its object, and *red* is an adjective describing *wine*. Hence, from the dependency tree with *loves* as root node, its subject and object as children, and their adjectival descriptors (if any) as their children, we read the following structure: $\overrightarrow{\text{loves}} \otimes \overrightarrow{subj} \otimes \overrightarrow{\text{Simon}} \otimes \overrightarrow{obj} \otimes \overrightarrow{\text{wine}} \otimes \overrightarrow{adj} \otimes \overrightarrow{\text{red}}$. Using the equality relation for inner products of tensor products:

$$\langle \overrightarrow{a} \otimes \overrightarrow{b} \mid \overrightarrow{c} \otimes \overrightarrow{d} \rangle = \langle \overrightarrow{a} \mid \overrightarrow{c} \rangle \times \langle \overrightarrow{b} \mid \overrightarrow{d} \rangle$$

We can therefore express inner-products of sentence vectors efficiently without ever having to actually represent the tensored sentence vector:

$$\begin{aligned}
&\langle \overrightarrow{\text{Simon loves red wine}} \mid \overrightarrow{\text{Mary likes delicious rosé}} \rangle \\
&= \langle \overrightarrow{\text{loves}} \otimes \overrightarrow{subj} \otimes \overrightarrow{\text{Simon}} \otimes \overrightarrow{obj} \otimes \overrightarrow{\text{wine}} \otimes \overrightarrow{adj} \otimes \overrightarrow{\text{red}} \mid \overrightarrow{\text{likes}} \otimes \overrightarrow{subj} \otimes \overrightarrow{\text{Mary}} \otimes \overrightarrow{obj} \otimes \overrightarrow{\text{rosé}} \otimes \overrightarrow{adj} \otimes \overrightarrow{\text{delicious}} \rangle \\
&= \langle \overrightarrow{\text{loves}} \mid \overrightarrow{\text{likes}} \rangle \times \langle \overrightarrow{subj} \mid \overrightarrow{subj} \rangle \times \langle \overrightarrow{\text{Simon}} \mid \overrightarrow{\text{Mary}} \rangle \times \langle \overrightarrow{obj} \mid \overrightarrow{obj} \rangle \times \langle \overrightarrow{\text{wine}} \mid \overrightarrow{\text{rosé}} \rangle \times \langle \overrightarrow{adj} \mid \overrightarrow{adj} \rangle \times \langle \overrightarrow{\text{red}} \mid \overrightarrow{\text{delicious}} \rangle \\
&= \langle \overrightarrow{\text{loves}} \mid \overrightarrow{\text{likes}} \rangle \times \langle \overrightarrow{\text{Simon}} \mid \overrightarrow{\text{Mary}} \rangle \times \langle \overrightarrow{\text{wine}} \mid \overrightarrow{\text{rosé}} \rangle \times \langle \overrightarrow{\text{red}} \mid \overrightarrow{\text{delicious}} \rangle
\end{aligned}$$

This example shows that this formalism allows for sentence comparison of sentences with identical dependency trees to be broken down to term-to-term comparison without the need for the tensor products to ever be computed or stored, reducing computation to inner product calculations.

However, although matching terms with identical syntactic roles in the sentence works well in the given example, this model suffers from the same problems as the original tensor-based compositionality of Smolensky (1990) in that, by the authors' own admission, sentences of different syntactic structure live in spaces of different dimensionality and thus cannot be directly compared. Hence we cannot use this to measure the similarity between even small variations in sentence structure, such as the pair "Simon likes red wine" and "Simon likes wine."

2.3.5 SVS Models. The idea of including syntactic relations to other lemmas in word representations discussed in Section 2.3.4 is applied differently in the structured vector space model presented by Erk and Padó (2008). They propose to represent word meanings not as simple vectors, but as triplets:

$$w = (v, R, R^{-1})$$

where v is the word vector, constructed as in any other DSM, R and R^{-1} are selectional preferences, and take the form of $\mathcal{R} \to \mathcal{D}$ maps where $\mathcal{R}$ is the set of dependency relations, and $\mathcal{D}$ is the set of word vectors. Selectional preferences are used to encode the lemmas that w is typically the parent of in the dependency trees of the corpus in the case of R, and typically the child of in the case of R^{-1}.

Composition takes the form of vector updates according to the following protocol. Let $a = (v_a, R_a, R_a^{-1})$ and $b = (v_b, R_b, R_b^{-1})$ be two words being composed, and let r be the dependency relation linking a to b. The vector update procedure is as follows:

$$a' = (v_a \odot R_b^{-1}(r), R_a - \{r\}, R_a^{-1})$$
$$b' = (v_b \odot R_a(r), R_b, R_b^{-1} - \{r\})$$

where a', b' are the updated word meanings, and $\odot$ is whichever vector composition (addition, component-wise multiplication) we wish to use. The word vectors in the triplets are effectively filtered by combination with the lemma which the word they are being composed with expects to bear relation r to, and this relation between the composed words a and b is considered to be used and hence removed from the domain of the selectional preference functions used in composition.

This mechanism is therefore a more sophisticated version of the compositional disambiguation mechanism discussed by Mitchell and Lapata (2008) in that the combination of words filters the meaning of the original vectors that may be ambiguous (e.g., if we have one vector for *bank*); but contrary to Mitchell and Lapata (2008), the information of the original vectors is modified but essentially preserved, allowing for further combination with other terms, rather than directly producing a joint vector for the composed words. The added fact that R and R^{-1} are partial functions associated with specific lemmas forces grammaticality during composition, since if a holds a dependency relation r to b which it never expects to hold (for example a verb having as its subject another verb, rather than the reverse), then R_a and R_b^{-1} are undefined for r and the update fails. However, there are some problems with this approach if our goal is true compositionality.

First, this model does not allow some of the "repeated compositionality" we need because of the update of R and R^{-1}. For example, we expect that an adjective composed with a noun produces something *like* a noun in order to be further composed with a verb or even another adjective. However, here, because the relation *adj* would be removed from R_b^{-1} for some noun b composed with an adjective a, this new representation b' would not have the properties of a noun in that it would no longer expect composition with an adjective, rendering representations of simple expressions like "the new red car" impossible. Of course, we could remove the update of the selectional preference functions from the compositional mechanism, but then we would lose this attractive feature of grammaticality enforcement through the partial functionality of R and R^{-1}.

Second, this model does little more than represent the implicit disambiguation that is expected during composition, rather than actually provide a full blown compositional model. The inability of this system to provide a novel mechanism by which to obtain a joint vector for two composed lemmas—thereby building towards sentence vectors—entails that this system provides no means by which to obtain semantic representations of larger syntactic structures that can be compared by inner product or cosine measure as is done with any other DSM. Of course, this model could be combined with the compositional models presented in Sections 2.3.1–2.3.3 to produce sentence vectors, but whereas some syntactic sensitivity would have been obtained, the word ordering and other problems of the aforementioned models would still remain, and little progress would have been made towards true compositionality.

We retain from this attempt to introduce compositionality in DSMs that including information obtained from syntactic dependency relations is important for proper disambiguation, and that having some mechanism by which the grammaticality of the expression being composed is a precondition for its composition is a desirable feature for any compositional mechanism.

2.4 Matrix-Based Compositionality

The final class of approaches to vector composition we wish to discuss are three matrix-based models.

Generic Additive Model. The first is the Generic Additive Model of Zanzotto et al. (2010). This is a generalization of the weighted additive model presented in Section 2.3.1. In this model, rather than multiplying lexical vectors by fixed parameters α and β before adding them to form the representation of their combination, they are instead the arguments of matrix multiplication by square matrices A and B:

$$\overrightarrow{ab} = A\overrightarrow{a} + B\overrightarrow{b}$$

Here, A and B represent the added information provided by putting two words into relation.

The numerical content of A and B is learned by linear regression over triplets $(\overrightarrow{a}, \overrightarrow{b}, \overrightarrow{c})$ where $\overrightarrow{a}$ and $\overrightarrow{b}$ are lexical semantic vectors, and $\overrightarrow{c}$ is the *expected* output of the combination of $\overrightarrow{a}$ and $\overrightarrow{b}$. This learning system thereby requires the provision of labeled data for linear regression to be performed. Zanzotto et al. (2010) suggest several sources for this labeled data, such as dictionary definitions and word etymologies.

This approach is richer than the weighted additive models because the matrices act as linear maps on the vectors they take as "arguments," and thus can encode more subtle syntactic or semantic relations. However, this model treats all word combinations as the same operation—for example, treating the combination of an adjective with its argument and a verb with its subject as the same sort of composition. Because of the diverse ways there are of training such supervised models, we leave it to those who wish to further develop this specific line of research to perform such evaluations.

Adjective Matrices. The second approach is the matrix-composition model of Baroni and Zamparelli (2010), which they develop only for the case of adjective-noun composition, although their approach can seamlessly be used for any other predicate-argument composition. Contrary to most of the earlier approaches proposed, which aim to combine two lexical vectors to form a lexical vector for their combination, Baroni and Zamparelli suggest giving different semantic representations to different types, or more specifically to adjectives and nouns.

In this model, nouns are lexical vectors, as with other models. However, embracing a view of adjectives that is more in line with formal semantics than with distributional semantics, they model adjectives as linear maps taking lexical vectors as input and producing lexical vectors as output. Such linear maps can be encoded as square matrices, and applied to their arguments by matrix multiplication. Concretely, let $M^{adjective}$ be the matrix encoding the adjective's linear map, and $\overrightarrow{noun}$ be the lexical semantic vector for a noun; their combination is simply

$$\overrightarrow{adjective\ noun} = M^{adjective} \times \overrightarrow{noun}$$

Similarly to the Generic Additive Model, the matrix for each adjective is learned by linear regression over a set of pairs $(\overrightarrow{noun}, \overrightarrow{c})$ where the vectors $\overrightarrow{noun}$ are the lexical semantic vectors for the arguments of the adjective in a corpus, and $\overrightarrow{c}$ is the semantic vector corresponding to the expected output of the composition of the adjective with that noun.

This may, at first blush, also appear to be a supervised training method for learning adjective matrices from "labeled data," seeing how the expected output vectors are needed. However, Baroni and Zamparelli (2010) work around this constraint by automatically producing the labeled data from the corpus by treating the adjective-noun compound as a single token, and learning its vector using the same distributional learning methods they used to learn the vectors for nouns. This same approach can be extended to other unary relations without change and, using the general framework of the current article, an extension of it to binary predicates has been presented in Grefenstette et al. (2013), using multistep regression. For a direct comparison of the results of this approach with some of the results of the current article, we refer the reader to Grefenstette et al. (2013).

Recursive Matrix-Vector Model. The third approach is the recently developed Recursive Matrix-Vector Model (MV-RNN) of Socher et al. (2012), which claims the two matrix-based models described here as special cases. In MV-RNN, words are represented as a pairing of a lexical semantic vector $\overrightarrow{a}$ with an operation matrix A. Within this model, given the parse of a sentence in the form of a binarized tree, the semantic representation $(\overrightarrow{c}, C)$ of each non-terminal node in the tree is produced by performing the following two operations on its children $(\overrightarrow{a}, A)$ and $(\overrightarrow{b}, B)$.

First, the vector component $\overrightarrow{c}$ is produced by applying the operation matrix of one child to the vector of the other, and vice versa, and then projecting both of the products back into the same vector space as the child vectors using a projection matrix W, which must also be learned:

$$\overrightarrow{c} = W \times \begin{bmatrix} B \times \overrightarrow{a} \\ A \times \overrightarrow{b} \end{bmatrix}$$

Second, the matrix C is calculated by projecting the pairing of matrices A and B back into the same space, using a projection matrix W_M, which must also be learned:

$$C = W_M \times \begin{bmatrix} A \\ B \end{bmatrix}$$

The pairing $(\overrightarrow{c}, C)$ obtained through these operations forms the semantic representation of the phrase falling under the scope of the segment of the parse tree below that node.

This approach to compositionality yields good results in the experiments described in Socher et al. (2012). It furthermore has appealing characteristics, such as treating relational words differently through their operation matrices, and allowing for recursive composition, as the output of each composition operation is of the same type of object as its inputs. This approach is highly general and has excellent coverage of different syntactic types, while leaving much room for parametric optimization.

The principal mathematical difference with the compositional framework presented subsequently is that composition in MV-RNN is always a binary operation; for example, to compose a transitive verb with its subject and object one would first need to compose it with its object, and then compose the output of that operation with the subject. The framework we discuss in this article allows for the construction of larger representations for relations of larger arities, permitting the simultaneous composition of a verb with its subject and object. Whether or not this theoretical difference leads to significant differences in composition quality requires joint evaluation. Additionally, the description of MV-RNN models in Socher et al. (2012) specifies the need for a source of learning error during training, which is easy to measure in the case of label prediction experiments such as sentiment prediction, but non-trivial in the case of paraphrase detection where no objective label exists. A direct comparison to MV-RNN methods within the context of experiments similar to those presented in this article has been produced by Blacoe and Lapata (2012), showing that simple operations perform on par with the earlier complex deep learning architectures produced by Socher and colleagues; we leave direct comparisons to future work. Early work has shown that the addition of a hidden layer with non-linearities to these simple models will improve the results.

2.5 Some Other Approaches to Distributional Semantics

Domains and Functions. In recent work, Turney (2012) suggests modeling word representations not as a single semantic vector, but as a pair of vectors: one containing the information of the word relative to its *domain* (the other words that are ontologically similar), and another containing information relating to its *function*. The former vector is learned by looking at what nouns a word co-occurs with, and the latter is learned by looking at what verb-based patterns the word occurs in. Similarity between sets of words is not determined by a single similarity function, but rather through a combination of comparisons of the domain components of words' representations with the function components of the words' representations. Such combinations are designed on a task-specific basis. Although Turney's work does not directly deal with vector composition of the sort we explore in this article, Turney shows that similarity measures can be designed for tasks similar to those presented here. The particular limitation of his approach, which Turney discusses, is that similarity measures must be specified for each task, whereas most of the compositional models described herein produce representations that can be compared in a task-independent manner (e.g., through cosine similarity). Nonetheless, this approach is innovative, and will merit further attention in future work in this area.

Language as Algebra. A theoretical model of *meaning as context* has been proposed in Clarke (2009, 2012). In that model, the meaning of any string of words is a vector built from the occurrence of the string in a corpus. This is the most natural extension of distributional models from words to strings of words: in that model, one builds vectors for strings of words in exactly the same way as one does for words. The main problem, however, is that of data sparsity for the occurrences of strings of words. Words do appear repeatedly in a document, but strings of words, especially for longer strings, rarely do so; for instance, it hardly happens that an exact same sentence appears more than once in a document. To overcome this problem, the model is based on the hypothetical concept of an infinite corpus, an assumption that prevents it from being applied to real-world corpora and experimented within natural language processing tasks. On the positive side, the model provides a theoretical study of the abstract properties of a general bilinear associative composition operator; in particular, it is shown that this operator encompasses other composition operators, such as addition, multiplication, and even tensor product.

3. DisCoCat

In Section 2, we discussed lexical DSMs and the problems faced by attempts to provide a vector composition operation that would allow us to form distributional sentence representations as a function of word meaning. In this section, we will present an existing formalism aimed at solving this compositionality problem, as well as the mathematical background required to understand it and further extensions, building on the features and failures of previously discussed attempts at syntactically sensitive compositionality.

Clark, Coecke, and Sadrzadeh (2008) and Coecke, Sadrzadeh, and Clark (2010) propose adapting a category theoretic model, inspired by the categorical compositional vector space model of quantum protocols (Abramsky and Coecke 2004), to the task of compositionality of semantic vectors. Syntactic analysis in the form of pregroup grammars—a categorial grammar—is given categorical semantics in order to be represented as a compact closed category P (a concept explained subsequently), the objects of which are syntactic types and the morphisms of which are the reductions forming the basis of syntactic analysis. This syntactic category is then mapped onto the semantic compact closed category **FVect** of finite dimensional vector spaces and linear maps. The mapping is done in the product category **FVect** $\times P$ via the following procedure. Each syntactic type is interpreted as a vector space in which semantic vectors for words with that particular syntactic type live; the reductions between the syntactic types are interpreted as linear maps between the interpreted vector spaces of the syntactic types.

The key feature of category theory exploited here is its ability to relate in a canonical way different mathematical formalisms that have similar structures, even if the original formalisms belong in different branches of mathematics. In this context, it has enabled us to relate syntactic types and reductions to vector spaces and linear maps and obtain a mechanism by which *syntactic analysis guides semantic composition operations*.

This pairing of syntactic analysis and semantic composition ensures both that grammaticality restrictions are in place as in the model of Erk and Padó (2008) and syntactically driven semantic composition in the form of inner-products provide the implicit disambiguation features of the compositional models of Erk and Padó (2008) and Mitchell and Lapata (2008). The composition mechanism also involves the projection of tensored vectors into a common semantic space without the need for full representation of the tensored vectors in a manner similar to Plate (1991), without restriction to the nature of the vector spaces it can be applied to. This avoids the problems faced by other tensor-based composition mechanisms such as Smolensky (1990) and Clark and Pulman (2007).

The word vectors can be specified model-theoretically and the sentence space can be defined over Boolean values to obtain grammatically driven truth-theoretic semantics in the style of Montague (1974), as proposed by Clark, Coecke, and Sadrzadeh (2008). Some logical operators can be emulated in this setting, such as using swap matrices for negation as shown by Coecke, Sadrzadeh, and Clark (2010). Alternatively, corpus-based variations on this formalism have been proposed by Grefenstette et al. (2011) to obtain a non-truth theoretic semantic model of sentence meaning for which logical operations have yet to be defined.

Before explaining how this formalism works, in Section 3.3, we will introduce the notions of pregroup grammars in Section 3.1, and the required basics of category theory in Section 3.2.

3.1 Pregroup Grammars

Presented by Lambek (1999, 2008) as a successor to his syntactic calculus (Lambek 1958), pregroup grammars are a class of categorial type grammars with pregroup algebras as semantics. Pregroups are particularly interesting within the context of this work because of their well-studied algebraic structure, which can trivially be mapped onto the structure of the category of vector spaces, as will be discussed subsequently. Logically speaking, a pregroup is a non-commutative form of Linear Logic (Girard 1987) in which the tensor and its dual par coincide; this logic is sometimes referred to as *Bi-Compact Linear Logic* (Lambek 1999). The formalism works alongside the general guidelines of other categorial grammars, for instance, those of the combinatory categorial grammar (CCG) designed by Steedman (2001) and Steedman and Baldridge (2011). They consist of atomic grammatical types that combine to form compound types. A series of CCG-like application rules allow for type-reductions, forming the basis of syntactic analysis. As our first step, we show how this syntactic analysis formalism works by presenting an introduction to pregroup algebras.

Pregroups. A pregroup is an algebraic structure of the form $(P, \leq, \cdot, 1, (-)^l, (-)^r)$. Its elements are defined as follows:

- P is a set $\{a, b, c, \ldots\}$.
- The relation $\leq$ is a partial ordering on P.
- The binary operation $\cdot$ is an associative, non-commutative monoid multiplication with the type $- \cdot - : P \times P \to P$, such that if $a, b \in P$ then $a \cdot b \in P$. In other words, P is closed under this operation.
- $1 \in P$ is the unit, satisfying $a \cdot 1 = a = 1 \cdot a$ for all $a \in P$.
- $(-)^l$ and $(-)^r$ are maps with types $(-)^l : P \to P$ and $(-)^r : P \to P$ such that for any $a \in P$, we have that $a^l, a^r \in P$. The images of these maps are referred to as the **left** and the **right adjoints**. These are unique and satisfy the following conditions:
  - Reversal: if $a \leq b$ then $b^l \leq a^l$ (and similarly for a^r, b^r).
  - Ordering: $a \cdot a^r \leq 1 \leq a^r \cdot a$ and $a^l \cdot a \leq 1 \leq a \cdot a^l$.
  - Cancellation: $a^{lr} = a = a^{rl}$.
  - Self-adjointness of identity: $1^r = 1 = 1^l$.
  - Self-adjointness of multiplication: $(a \cdot b)^r = b^r \cdot a^r$.

As a notational simplification we write ab for $a \cdot b$, and if $abcd \leq cd$ we write $abcd \rightarrow cd$ and call this a **reduction**, omitting the identity wherever it might appear. Monoid multiplication is associative, so parentheses may be added or removed for notational clarity without changing the meaning of the expression as long as they are not directly under the scope of an adjoint operator.

An example reduction in pregroup might be:

$$aa^rbc^lc \rightarrow bc^lc \rightarrow b$$

We note here that the reduction order is not always unique, as we could have reduced as follows: $aa^rbc^lc \rightarrow aa^rb \rightarrow b$. As a further notational simplification, if there exists a chain of reductions $a \rightarrow \ldots \rightarrow b$ we may simply write $a \rightarrow b$ (in virtue of the transitivity of partial ordering relations). Hence in our given example, we can express both reduction paths as $aa^rbc^lc \rightarrow b$.

Pregroups and Syntactic Analysis. Pregroups are used for grammatical analysis by freely generating the set P of a pregroup from the basic syntactic types $n, s, \ldots,$ where here n stands for the type of both a noun and a noun phrase and s for that of a sentence. The conflation of nouns and noun phrases suggested here is done to keep the work discussed in this article as simple as possible, but we could of course model them as different types in a more sophisticated version of this pregroup grammar. As in any categorial grammar, words of the lexicon are assigned one or more possible types (corresponding to different syntactic roles) in a predefined *type dictionary*, and the grammaticality of a sentence is verified by demonstrating the existence of a reduction from the combination of the types of words within the sentence to the sentence type s.

For example, having assigned to noun phrases the type n and to sentences the type s, the transitive verbs will have the compound type n^rsn^l. We can read from the type of a transitive verb that it is the type of a word which "expects" a noun phrase on its left and a noun phrase on its right, in order to produce a sentence. A sample reduction of *John loves cake* with *John* and *cake* being noun phrases of type n and *loves* being a verb of type n^rsn^l is as follows:

$$n(n^rsn^l)n \rightarrow s$$

We see that the transitive verb has combined with the subject and object to reduce to a sentence. Because the combination of the types of the words in the string *John loves cake* reduces to s, we say that this string of words is a grammatical sentence. As for more examples, we recall that intransitive verbs can be given the type n^rs such that *John sleeps* would be analyzed in terms of the reduction $n(n^rs) \rightarrow s$. Adjectives can be given the type nn^l such that *red round rubber ball* would be analyzed by $(nn^l)(nn^l)(nn^l)n \rightarrow n$. And so on and so forth for other syntactic classes.

Lambek (2008) presents the details of a slightly more complex pregroup grammar with a richer set of types than presented here. This grammar is hand-constructed and iteratively extended by expanding the type assignments as more sophisticated grammatical constructions are discussed. No general mechanism is proposed to cover all such types of assignments for larger fragments (e.g., as seen in empirical data). Pregroup grammars have been proven to be learnable by Béchet, Foret, and Tellier (2007), who also discuss the difficulty of this task and the nontractability of the procedure. Because of these constraints and lack of a workable pregroup parser, the pregroup grammars we will use in our categorical formalism are derived from CCG types, as we explain in the following.

Pregroup Grammars and Other Categorial Grammars. Pregroup grammars, in contrast with other categorial grammars such as CCG, do not yet have a large set of tools for parsing available. If quick implementation of the formalism described later in this paper is required, it would be useful to be able to leverage the mature state of parsing tools available for other categorial grammars, such as the Clark and Curran (2007) statistical CCG parser, as well as Hockenmaier's CCG lexicon and treebank (Hockenmaier 2003; Hockenmaier and Steedman 2007). In other words, is there any way we can translate at least some subset of CCG types into pregroup types?

There are some theoretical obstacles to consider first: Pregroup grammars and CCG are not equivalent. Buszkowski (2001) shows pregroup grammars to be equivalent to context-free grammars, whereas Joshi, Vijay-Shanker, and Weir (1989) show CCG to be weakly equivalent to more expressive mildly context-sensitive grammars. However, if our goal is to exploit the CCG used in Clark and Curran's parser, or Hockenmaier's lexicon and treebank, we may be in luck: Fowler and Penn (2010) prove that some CCGs, such as those used in the aforementioned tools, are strongly context-free and thus expressively equivalent to pregroup grammars. In order to be able to apply the parsing tools for CCGs to our setting, we use a translation mechanism from CCG types to pregroup types based on the Lambek-calculus-to-pregroup-grammar translation originally presented in Lambek (1999). In this mechanism, each atomic CCG type X is assigned a unique pregroup type x; for any X/Y in CCG we have xy^l in the pregroup grammar; and for any $X\backslash Y$ in CCG we have $y^r x$ in pregroup grammar. Therefore, by assigning NP to n and S to s we could, for example, translate the CCG transitive verb type $(S\backslash NP)/NP$ into the pregroup type $n^r s n^l$, which corresponds to the pregroup type we used for transitive verbs in Section 3.1. Wherever type replacement (e.g., $N \to NP$) is allowed in CCG we set an ordering relation in the pregroup grammar (e.g., $\bar{n} \leq n$, where $\bar{n}$ is the pregroup type associated with N). Because forward and backward slash "operators" in CCG are not associative whereas monoid multiplication in pregroups is, it is evident that some information is lost during the translation process. But because the translation we need is one-way, we may ignore this problem and use CCG parsing tools to obtain pregroup parses. Another concern lies with CCG's crossed composition and substitution rules. The translations of these rules do not in general hold in a pregroup; this is not a surprise as pregroups are a simplification of the Lambek Calculus and these rules did not hold in the Lambek Calculus either, as shown in Moortgat (1997), for example. However, for the phenomena modeled in this paper, the CCG rules without the backward cross rules will suffice. In general for the case of English, one can avoid the use of these rules by overloading the lexicon and using additional categories. To deal with languages that have cross dependancies, such as Dutch, various solutions have been suggested (e.g., see Genkin, Francez, and Kaminski 2010; Preller 2010).

3.2 Categories

Category theory is a branch of pure mathematics that allows for a general and uniform formulation of various different mathematical formalisms in terms of their main structural properties using a few abstract concepts such as objects, arrows, and combinations and compositions of these. This uniform conceptual language allows for derivation of new properties of existing formalisms and for relating these to properties of other formalisms, if they bear similar categorical representation. In this function, it has been at the center of recent work in unifying two orthogonal models of meaning, a qualitative categorial grammar model and a quantitative distributional model (Clark, Coecke, and Sadrzadeh 2008; Coecke, Sadrzadeh, and Clark 2010). Moreover, the unifying categorical structures at work here were inspired by the ones used in the foundations of physics and the modeling of quantum information flow, as presented in Abramsky and Coecke (2004), where they relate the logical structure of quantum protocols to their state-based vector spaces data. The connection between the mathematics used for this branch of physics and those potentially useful for linguistic modeling has also been noted by several sources, such as Widdows (2005), Lambek (2010), and Van Rijsbergen (2004).

In this section, we will briefly examine the basics of category theory, monoidal categories, and compact closed categories. The focus will be on defining enough basic concepts to proceed rather than provide a full-blown tutorial on category theory and the modeling of information flow, as several excellent sources already cover both aspects (e.g., Mac Lane 1998; Walters 1991; Coecke and Paquette 2011). A categories-in-a-nutshell crash course is also provided in Clark, Coecke, and Sadrzadeh (2008) and Coecke, Sadrzadeh, and Clark (2010).

The Basics of Category Theory. A basic category **C** is defined in terms of the following elements:

- A collection of objects $ob(\mathbf{C})$.
- A collection of morphisms $hom(\mathbf{C})$.
- A morphism composition operation $\circ$.

Each morphism f has a domain $dom(f) \in ob(\mathbf{C})$ and a codomain $codom(f) \in ob(\mathbf{C})$. For $dom(f) = A$ and $codom(f) = B$ we abbreviate these definitions as $f : A \to B$. Despite the notational similarity to function definitions (and sets and functions being an example of a category), it is important to state that nothing else is presupposed about morphisms, and we should not treat them a priori as functions.

The following axioms hold in every category **C** :

- For any $f : A \to B$ and $g : B \to C$ there exists $h : A \to C$ and $h = g \circ f$.
- For any $f : A \to B, g : B \to C$ and $h : C \to D$, $\circ$ satisfies $(h \circ g) \circ f = h \circ (g \circ f)$.
- For every $A \in ob(\mathbf{C})$ there is an identity morphism $id_A : A \to A$ such that for any $f : A \to B, f \circ id_A = f = id_B \circ f$.

We can model various mathematical formalisms using these basic concepts, and verify that these axioms hold for them. For example, category *Set* with sets as objects and functions as morphisms, or category *Rel* with sets as objects and relations as morphisms, category *Pos* with posets as objects and order-preserving maps as morphisms, and category *Group* with groups as objects and group homomorphisms as morphisms, to name a few.

The product $\mathbf{C} \times \mathbf{D}$ of two categories **C** and **D** is a category with pairs (A, B) as objects, where $A \in ob(\mathbf{C})$ and $B \in ob(\mathbf{D})$. There exists a morphism $(f, g) : (A, B) \to (C, D)$ in $\mathbf{C} \times \mathbf{D}$ if and only if there exists $f : A \to C \in hom(\mathbf{C})$ and $g : B \to D \in hom(\mathbf{D})$. Product categories allow us to relate objects and morphisms of one mathematical formalism to those in another, in this example those of **C** to **D**.

Compact Closed Categories. A monoidal category **C** is a basic category to which we add a monoidal tensor $\otimes$ such that:

- For all $A, B \in ob(\mathbf{C})$ there is an object $A \otimes B \in ob(\mathbf{C})$.
- For all $A, B, C \in ob(\mathbf{C})$, we have $(A \otimes B) \otimes C = A \otimes (B \otimes C)$.
- There exists some $I \in ob(\mathbf{C})$ such that for any $A \in ob(C)$, we have $I \otimes A = A = A \otimes I$.
- For $f : A \to C$ and $g : B \to D$ in $hom(\mathbf{C})$ there is $f \otimes g : A \otimes B \to C \otimes D$ in $hom(\mathbf{C})$.
- For $f_1 : A \to C, f_2 : B \to D, g_1 : C \to E$ and $g_2 : D \to F$ the following equality holds:

$$(g_1 \otimes g_2) \circ (f_1 \otimes f_2) = (g_1 \circ f_1) \otimes (g_2 \circ f_2)$$

The product category of two monoidal categories has a monodial tensor, defined pointwisely by $(a, A) \otimes (b, B) := (a \otimes b, A \otimes B)$.

A compact closed category **C** is a monoidal category with the following additional axioms:

- Each object $A \in ob(\mathbf{C})$ has left and right "adjoint" objects A^l and A^r in $ob(\mathbf{C})$.
- There exist four structural morphisms for each object $A \in ob(\mathbf{C})$:
  - $\eta^l_A : I \to A \otimes A^l$.
  - $\eta^r_A : I \to A^r \otimes A$.
  - $\epsilon^l_A : A^l \otimes A \to I$.
  - $\epsilon^r_A : A \otimes A^r \to I$.
- The previous structural morphisms satisfy the following equalities:
  - $(1_A \otimes \epsilon^l_A) \circ (\eta^l_A \otimes 1_A) = 1_A$.
  - $(\epsilon^r_A \otimes 1_A) \circ (1_A \otimes \eta^r_A) = 1_A$.
  - $(1_{A^r} \otimes \epsilon^r_A) \circ (\eta^r \otimes 1_{A^r}) = 1_{A^r}$.
  - $(\epsilon^l_A \otimes 1_{A^l}) \circ (1_{A^l} \otimes \eta^l_A) = 1_{A^l}$.

Compact closed categories come equipped with complete graphical calculi, surveyed in Selinger (2010). These calculi visualize and simplify the axiomatic reasoning within the category to a great extent. In particular, Clark, Coecke, and Sadrzadeh (2008) and Coecke, Sadrzadeh, and Clark (2010) show how they depict the pregroup grammatical reductions and visualize the flow of information in composing meanings of single words and forming meanings for sentences. Although useful at an abstract level, these calculi do not play the same simplifying role when it comes to the concrete and empirical computations; therefore we will not discuss them in this article.

A very relevant example of a compact closed category is a pregroup algebra P. Elements of a pregroup are objects of the category, the partial ordering relation provides the morphisms, 1 is I, and monoidal multiplication is the monoidal tensor.

Another very relevant example is the category **FVect** of finite dimensional Hilbert spaces and linear maps over $\mathbb{R}$—that is, vector spaces over $\mathbb{R}$ with orthogonal bases of finite dimension, and an inner product operation $\langle - \mid - \rangle : A \times A \to \mathbb{R}$ for every vector space A. The objects of **FVect** are vector spaces, and the morphisms are linear maps between vector spaces. The unit object is $\mathbb{R}$ and the monoidal tensor is the linear algebraic tensor product. **FVect** is degenerate in its adjoints, in that for any vector space A, we have $A^l = A^r = A^*$, where A^* is the dual space of A. Moreover, by fixing a basis we obtain that $A^* \cong A$. As such, we can effectively do away with adjoints in this category, and "collapse" ϵ^l, ϵ^r, η^l, and η^r maps into "adjoint-free" ϵ and η maps. In this category, the ϵ maps are inner product operations, $\epsilon_A : A \otimes A \to \mathbb{R}$, and the η maps $\eta : \mathbb{R} \to A \otimes A$ generate maximally entangled states, also referred to as **Bell-states**.

3.3 A Categorical Passage from Grammar to Semantics

In Section 3.2 we showed how a pregroup algebra and vector spaces can be modeled as compact closed categories and how product categories allow us to relate the objects and morphisms of one category to those of another. In this section, we will present how Clark, Coecke, and Sadrzadeh (2008) and Coecke, Sadrzadeh, and Clark (2010) suggest building on this by using categories to relate semantic composition to syntactic analysis in order to achieve syntax-sensitive composition in DSMs.

3.3.1 Syntax Guides Semantics. The product category **FVect** $\times P$ has as object pairs (A, a), where A is a vector space and a is a pregroup grammatical type, and as morphism pairs $(f, \leq)$ where f is a linear map and $\leq$ a pregroup ordering relation. By the definition of product categories, for any two vector space-type pairs (A, a) and (B, b), there exists a morphism $(A, a) \to (B, b)$ only if there is both a linear map from A into B and a partial ordering $a \to b$. If we view these pairings as the association of syntactic types with vector spaces containing semantic vectors for words of that type, this restriction effectively states that a linear map from A to B is only "permitted" in the product category if a reduces to b.

Both P and **FVect** being compact closed, it is simple to show that **FVect** $\times P$ is as well, by considering the pairs of unit objects and structural morphisms from the separate categories: I is now $(\mathbb{R}, 1)$, and the structural morphisms are $(\epsilon_A, \epsilon_a^l)$, $(\epsilon_A, \epsilon_a^r)$, (η_A, η_a^l), and (η_A, η_a^r). We are particularly interested in the ϵ maps, which are defined as follows (from the definition of product categories):

$$(\epsilon_A, \epsilon_A^l) : (A \otimes A, a^l a) \to (\mathbb{R}, 1) \qquad (\epsilon_A, \epsilon_A^r) : (A \otimes A, a a^r) \to (\mathbb{R}, 1)$$

This states that whenever there is a reduction step in the grammatical analysis of a sentence, there is a composition operation in the form of an inner product on the semantic front. Hence, if nouns of type n live in some noun space N and transitive verbs of type $n^l s n^r$ live in some space $N \otimes S \otimes N$, then there must be some structural morphism of the form:

$$(\epsilon_N \otimes 1_S \otimes \epsilon_N, \epsilon_n^r 1_s \epsilon_n^l) : (N \otimes (N \otimes S \otimes N) \otimes N, n(n^r s n^l)n) \to (S, s)$$

We can read from this morphism the functions required to compose a sentence with a noun, a transitive verb, and an object to obtain a vector living in some sentence space S, namely, $(\epsilon_N \otimes 1_S \otimes \epsilon_N)$.

The form of a syntactic type is therefore what dictates the structure of the semantic space associated with it. The structural morphisms of the product category guarantee that for every syntactic reduction there is a semantic composition morphism provided by the product category: *syntactic analysis guides semantic composition.*

3.3.2 Example. To give an example, we give syntactic type n to nouns, and $n^r s$ to intransitive verbs. The grammatical reduction for *kittens sleep*, namely, $nn^r s \to s$, corresponds to the morphism $\epsilon_n^r \otimes 1_s$ in P. The syntactic types dictate that the noun $\overrightarrow{\text{kittens}}$ lives in some vector space N, and the intransitive verb $\overrightarrow{\text{sleep}}$ in $N \otimes S$. The reduction morphism gives us the composition morphism $(\epsilon_N \otimes 1_S)$, which we can apply to $\overrightarrow{\text{kittens}} \otimes \overrightarrow{\text{sleep}}$.

Because we can express any vector as the weighted sum of its basis vectors, let us expand $\overrightarrow{\text{kittens}} = \sum_i c_i^{\text{kittens}} \overrightarrow{n_i}$ and $\overrightarrow{\text{sleep}} = \sum_{ij} c_{ij}^{\text{sleep}} \overrightarrow{n_i} \otimes \overrightarrow{s_j}$; then we can express the composition as follows:

$$\begin{aligned}
\overrightarrow{\text{kittens sleep}} &= (\epsilon_N \otimes 1_S)(\overrightarrow{\text{kittens}} \otimes \overrightarrow{\text{sleep}}) \\
&= (\epsilon_N \otimes 1_S)\left(\sum_i c_i^{\text{kittens}} \overrightarrow{n_i} \otimes \sum_{jk} c_{jk}^{\text{sleep}} \overrightarrow{n_j} \otimes \overrightarrow{s_k}\right) \\
&= (\epsilon_N \otimes 1_S)\left(\sum_{ijk} c_i^{\text{kittens}} c_{jk}^{\text{sleep}} \overrightarrow{n_i} \otimes \overrightarrow{n_j} \otimes \overrightarrow{s_k}\right) \\
&= \sum_{ijk} c_i^{\text{kittens}} c_{jk}^{\text{sleep}} \langle \overrightarrow{n_i} \mid \overrightarrow{n_j} \rangle \overrightarrow{s_k} \\
&= \sum_{ik} c_i^{\text{kittens}} c_{ik}^{\text{sleep}} \overrightarrow{s_k}
\end{aligned}$$

In these equations, we have expressed the vectors in their explicit form, we have consolidated the sums by virtue of distributivity of linear algebraic tensor product over addition, we have applied the tensored linear maps to the vector components (as the weights are scalars), and finally, we have simplified the indices since $\langle \overrightarrow{n_i} \mid \overrightarrow{n_j} \rangle = 1$ if $\overrightarrow{n_i} = \overrightarrow{n_j}$ and 0 otherwise. As a result of these, we have obtained a vector that lives in sentence space S.

Transitive sentences can be dealt with in a similar fashion:

$$\begin{aligned}
&\overrightarrow{\text{kittens chase mice}} \\
&\quad = (\epsilon_N \otimes 1_S \otimes \epsilon_N)(\overrightarrow{\text{kittens}} \otimes \overrightarrow{\text{chase}} \otimes \overrightarrow{\text{mice}}) \\
&\quad = (\epsilon_N \otimes 1_S \otimes \epsilon_N)\left(\sum_i c_i^{\text{kittens}} \overrightarrow{n_i} \otimes \left(\sum_{jkl} c_{jkl}^{\text{chase}} \overrightarrow{n_j} \otimes \overrightarrow{s_k} \otimes \overrightarrow{n_l}\right) \otimes \sum_m c_m^{\text{mice}} \overrightarrow{n_m}\right) \\
&\quad = (\epsilon_N \otimes 1_S \otimes \epsilon_N)\left(\sum_{ijklm} c_i^{\text{kittens}} c_{jkl}^{\text{chase}} c_m^{\text{mice}} \overrightarrow{n_i} \otimes \overrightarrow{n_j} \otimes \overrightarrow{s_k} \otimes \overrightarrow{n_l} \otimes \overrightarrow{n_m}\right) \\
&\quad = \sum_{ijklm} c_i^{\text{kittens}} c_{jkl}^{\text{chase}} c_m^{\text{mice}} \langle \overrightarrow{n_i} \mid \overrightarrow{n_j} \rangle \overrightarrow{s_k} \langle \overrightarrow{n_l} \mid \overrightarrow{n_m} \rangle \\
&\quad = \sum_{ikm} c_i^{\text{kittens}} c_{ikm}^{\text{chase}} c_m^{\text{mice}} \overrightarrow{s_k}
\end{aligned}$$

In both cases, it is important to note that the tensor product passed as argument to the composition morphism, namely, $\overrightarrow{\text{kittens}} \otimes \overrightarrow{\text{sleep}}$ in the intransitive case and $\overrightarrow{\text{kittens}} \otimes \overrightarrow{\text{chase}} \otimes \overrightarrow{\text{mice}}$ in the transitive case, never needs to be computed. We can treat the tensor products here as commas separating function arguments, thereby avoiding the dimensionality problems presented by earlier tensor-based approaches to compositionality.

3.4 This Article and the DisCoCat Literature

As elaborated on in Section 2.3.4, the first general setting for pairing meaning vectors with syntactic types was proposed in Clark and Pulman (2007). The setting of a DisCoCat generalized this by making the meaning derivation process rely on a syntactic type system, hence overcoming its central problem whereby the vector representations of strings of words with different grammatical structure lived in different spaces. A preliminary version of a DisCoCat was developed in Clark, Coecke, and Sadrzadeh (2008), a full version was elaborated on in Coecke, Sadrzadeh, and Clark (2010), where, based on the developments of Preller and Sadrzadeh (2010), it was also exemplified how the vector space model may be instantiated in a truth theoretic setting where meanings of words were sets of their denotations and meanings of sentences were their truth values. A nontechnical description of this theoretical setting was presented in Clark (2013), where a plausibility truth-theoretic model for sentence spaces was worked out and exemplified. The work of Grefenstette et al. (2011) focused on a tangential branch and developed a toy example where neither words nor sentence spaces were Boolean. The applicability of the theoretical setting to a real empirical natural language processing task and data from a large scale corpus was demonstrated in Grefenstette and Sadrzadeh (2011a, 2011b). There, we presented a general algorithm to build vector representations for words with simple and complex types and the sentences containing them; then applied the algorithm to a disambiguation task performed on the British National Corpus (BNC). We also investigated the vector representation of transitive verbs and showed how a number of single operations may optimize the performance. We discuss these developments in detail in the following sections.

4. Concrete Semantic Spaces

In Section 3.3 we presented a categorical formalism that relates syntactic analysis steps to semantic composition operations. The structure of our syntactic representation dictates the structure of our semantic spaces, but in exchange, we are provided with composition functions by the syntactic analysis, rather than having to stipulate them ad hoc. Whereas the approaches to compositional DSMs presented in Section 2 either failed to take syntax into account during composition, or did so at the cost of not being able to compare sentences of different structure in a common space, this categorical approach projects all sentences into a common sentence space where they can be directly compared. However, this alone does not give us a compositional DSM.

As we have seen in the previous examples, the structure of semantic spaces varies with syntactic types. We therefore cannot construct vectors for different syntactic types in the same way, as they live in spaces of different structure and dimensionality. Furthermore, nothing has yet been said about the structure of the sentence space S into which expressions reducing to type s are projected. If we wish to have a compositional DSM that leverages all the benefits of lexical DSMs and ports them to sentence-level distributional representations, we must specify a new sort of vector construction procedure.

In the original formulation of this formalism by Clark, Coecke, and Sadrzadeh (2008) and Coecke, Sadrzadeh, and Clark (2010), examples of how such a compositional DSM could be used for logical evaluation are presented, where S is defined as a Boolean space with True and False as basis vectors. However, the word vectors used are hand-written and specified model-theoretically, as the authors leave it for future research to determine how such vectors might be obtained from a corpus. In this section, we will discuss a new way of constructing vectors for compositional DSMs, and of defining sentence space S, in order to reconcile this powerful categorical formalism with the applicability and flexibility of standard distributional models.

4.1 Defining Sentence Space

Assume the following sentences are all true:

1. The dogs chased the cats.
2. The dogs annoyed the cats.
3. The puppies followed the kittens.
4. The train left the station.
5. The president followed his agenda.

If asked which sentences have similar meaning, we would most likely point to the pair (1) and (3), and perhaps to a lesser degree (1) and (2), and (2) and (3). Sentences (4) and (5) obviously speak of a different state of the world as the other sentences.

If we compare these by truth value, we obviously have no means of making such distinctions. If we compare these by lexical similarity, (1) and (2) seem to be a closer match than (1) and (3). If we are classifying these sentences by some higher order relation such as “topic,” (5) might end up closer to (3) than (1). What, then, might cause us to pair (1) and (3)?

Intuitively, this similarity seems to be because the subjects and objects brought into relation by similar verbs are themselves similar. Abstracting away from tokens to some notion of property, we might say that both (1) and (3) express the fact that something furry and feline and furtive is being pursued by something aggressive (or playful) and canine. Playing along with the idea that lexical distributional semantics present concepts (word meanings) as “messy bundles of properties,” it seems only natural to have the way these properties are acted upon, qualified, and related as the basis for sentence-level distributional representations. In this respect, we here suggest that the sentence space S, instead of qualifying the truth value of a sentence, should express how the properties of the semantic objects within are qualified or brought into relation by verbs, adjectives, and other predicates and relations.

More specifically, we examine two suggestions for defining the sentence space, namely, $S_I \cong N$ for sentences with intransitive verbs and $S_T \cong N \otimes N$ for sentences with transitive verbs. These definitions mean that the sentence space's dimensions are commensurate with either those of N, or those of $N \otimes N$. These are by no means the only options, but as we will discuss here, they offer practical benefits.

In the case of S_I, the basis elements are labeled with unique basis elements of N, hence, $\overrightarrow{s_1} = \overrightarrow{n_1}$, $\overrightarrow{s_2} = \overrightarrow{n_2}$, and so on. In the case of S_T, the basis elements are labeled with unique ordered pairs of elements from N, for example, $\overrightarrow{s_1} = \overrightarrow{(n_1, n_1)}$, $\overrightarrow{s_2} = \overrightarrow{(n_2, n_1)}$, $\overrightarrow{s_3} = \overrightarrow{(n_1, n_2)}$, and so on, following the same matrix flattening structure used in the proof of the equal cardinality of $\mathbb{N}$ and $\mathbb{Q}$. Because of the isomorphism between S_T and $N \otimes N$, we will use the notations $\overrightarrow{(n_i, n_j)}$ and $\overrightarrow{n_i} \otimes \overrightarrow{n_j}$ interchangeably, as both constitute appropriate ways of representing the basis elements of such a space. To propagate this distinction on the syntactic level, we define types s_I and s_T for intransitive and transitive sentences, respectively.

In creating this distinction, we lost one of the most appealing features of the framework of Coecke, Sadrzadeh, and Clark (2010), namely, the result that all sentence vectors live in the same sentence space. A mathematical solution to this two-space problem was suggested in Grefenstette et al. (2011), and a variant of the models presented in this article permitting the non-problematic projection into a single sentence space ($S \cong N$) has been presented by Grefenstette et al. (2013). Keeping this separation allows us to deal with both the transitive and intransitive cases in a simpler manner, and because the experiments in this article only compare intransitive sentences with intransitive sentences, and transitive sentences with transitive sentences, we will not address the issue of unification here.

4.2 Noun-Oriented Types

While Lambek's pregroup types presented in Lambek (2008) include a rich array of basic types and hand-designed compound types in order to capture specific grammatic properties, for the sake of simplicity we will use a simpler set of grammatical types for experimental purposes, similar to some common types found in the CCG-bank (Steedman 2001).

We assign a basic pregroup type n for all nouns, with an associated vector space N for their semantic representations. Furthermore, we will treat noun-phrases as nouns, assigning to them the same pregroup type and semantic space.

CCG treats intransitive verbs as functions $NP\backslash S$ that consume a noun phrase and return a sentence, and transitive verbs as functions $(NP\backslash S)/NP$ that consume a noun phrase and return an intransitive verb function, which in turn consumes a noun phrase and returns a sentence. Using our distinction between intransitive sentences, we give intransitive verbs the type $n^r s_I$ associated with the semantic space $N \otimes S_I$, and transitive verbs the type $n^r s_T n^l$ associated with the semantic space $N \otimes S_T \otimes N$.

Adjectives, in CCG, are treated as functions NP/NP, consuming a noun phrase and returning a noun phrase; and hence we give them the type nn^l and associated semantic space $N \otimes N$.

With the provision of a learning procedure for vectors in these semantic spaces, we can use these types to construct sentence vector representations for simple intransitive verb–based and transitive verb–based sentences, with and without adjectives applied to subjects and objects.

4.3 Learning Procedures

To begin, we construct the semantic space N for all nouns in our lexicon (typically limited by the words available in the corpus used). Any distributional semantic model can be used for this stage, such as those presented in Curran (2004), or the lexical semantic models used by Mitchell and Lapata (2008). It seems reasonable to assume that higher quality lexical semantic vectors—as measured by metrics such as the WordSim353 test of Finkelstein et al. (2001)—will produce better relational vectors from the procedure designed subsequently. We will not test the hypothesis here, but note that it is an underlying assumption in most of the current literature on the subject (Erk and Padó 2008; Mitchell and Lapata 2008; Baroni and Zamparelli 2010).

Building upon the foundation of the thus constructed noun vectors, we construct semantic representations for relational words. In pregroup grammars (or other combinatorial grammars such as CCG), we can view such words as functions, taking as arguments those types present as adjoints in the compound pregroup type, and returning a syntactic object whose type is that of the corresponding reduction. For example, an adjective nn^l takes a noun or noun phrase n and returns a noun phrase n from the reduction $(nn^l)n \to n$. It can also compose with another adjective to return an adjective $(nn^l)(nn^l) \to nn^l$. We wish for our semantic representations to be viewed in the same way, such that the composition of an adjective with a noun $(1_N \otimes \epsilon_N)((N \otimes N) \otimes N)$ can be viewed as the application of a function $f : N \to N$ to its argument of type N.

To learn the representation of such functions, we assume that their meaning can be characterized by the properties that their arguments hold in the corpus, rather than just by their context as is the case in lexical distributional semantic models. To give an example, rather than learning what the adjective *angry* means by observing that it co-occurs with words such as *fighting*, *aggressive*, or *mean*, we learn its meaning by observing that it typically takes as argument words that have the property of being (i.e., co-occur with words such as) *fighting*, *aggressive*, and *mean*. Whereas in the lexical semantic case, such associations might only rarely occur in the corpus, in this indirect method we learn what properties the adjective relates to even if they do not co-occur with it directly.

In turn, through composition with its argument, we expect the function for such an adjective to *strengthen* the properties that characterize it in the object it takes as argument. If, indeed, *angry* is characterized by arguments that have high basis weights for basis elements corresponding to the concepts (or context words) *fighting*, *aggressive*, and *mean*, and relatively low counts for semantically different concepts such as *passive*, *peaceful*, and *loves*, then when we apply *angry* to *dog* the vector for the compound *angry dog* should contain some of the information found in the vector for *dog*. But this vector should also have *higher* values for the basis weights of *fighting*, *aggressive*, and *mean*, and correspondingly lower values for the basis weights of *passive*, *peaceful*, and *loves*.

To turn this idea into a concrete algorithm for constructing the semantic representation for relations of any arity, as first presented in Grefenstette et al. (2011), we examine how we would deal with this for binary relations such as transitive verbs. If a transitive verb of semantic type $N \otimes S_T \otimes N$ is viewed as a function $f : N \times N \to S_T$ that expresses the extent to which the properties of subject and object are brought into relation by the verb, we learn the meaning of the verb by looking at what properties are brought into relation by its arguments in a corpus. Recall that the vector for a verb v, $\overrightarrow{v} \in N \otimes S_T \otimes N$, can be expressed as the weighted sum of its basis elements:

$$\overrightarrow{v} = \sum_{ijk} c^v_{ijk} \overrightarrow{n_i} \otimes \overrightarrow{s_j} \otimes \overrightarrow{n_k}$$

We take the set of vectors for the subject and object of v in the corpus to be $arg_v = \{(\overrightarrow{SUBJ_l}, \overrightarrow{OBJ_l})\}_l$. We wish to calculate the basis weightings $\{c^v_{ijk}\}_{ijk}$ for v. Exploiting our earlier definition of the basis $\{s_j\}_j$ of S_T, which states that for any value of i and k there is some value of j such that $s_j = (n_i, n_k)$, we define $\Delta_{ijk} = 1$ if indeed $s_j = (n_i, n_k)$ and 0 otherwise. Using all of this, we define the calculation of each basis weight c^v_{ijk} as:

$$c^v_{ijk} = \sum_l \Delta_{ijk} c_i^{SUBJ_l} c_k^{OBJ_l}$$

This allows for a full formulation of $\overrightarrow{v}$ as follows:

$$\overrightarrow{v} = \sum_l \sum_{ijk} \Delta_{ijk} c_i^{SUBJ_l} c_k^{OBJ_l} \overrightarrow{n_i} \otimes \overrightarrow{s_j} \otimes \overrightarrow{n_k}$$

The nested sums here may seem computationally inefficient, seeing how this would involve computing $size(arg_v) \times dim(N)^2 \times dim(S) = size(arg_v) \times dim(N)^4$ products. However, using the decomposition of basis elements of S into pairs of basis elements of N (effectively basis elements of $N \otimes N$), we can remove the Δ_{ijk} term and ignore all values of j where $s_j \neq (n_i, n_k)$, because the basis weight for this combination of indices would be 0. Hence we simplify:

$$\overrightarrow{v} = \sum_l \sum_{ik} c_i^{SUBJ_l} c_k^{OBJ_l} \overrightarrow{n_i} \otimes \overrightarrow{(n_i, n_k)} \otimes \overrightarrow{n_k}$$

This representation is still bloated: We perform fewer calculations, but still obtain a vector in which all the basis weights where $s_j \neq (n_i, n_k)$ are 0, hence where only $dim(N)^2$ of the $dim(N)^4$ values are non-zero. In short, the vector weights for $\overrightarrow{v}$ are, under this learning algorithm, entirely characterized by the values of a $dim(N)$ by $dim(N)$ matrix, the entries of which are products $c_i^{SUBJ_l} c_k^{OBJ_l}$ where i and k have become row and column indices.

Using this and our definition of S_T as a space isomorphic to $N \otimes N$, we can formulate a compact expression of $\overrightarrow{v}$ as follows. Let the kronecker product of two vectors $\overrightarrow{u}, \overrightarrow{w} \in N$, written $\overrightarrow{u} \otimes \overrightarrow{w} \in N \otimes N$, be as follows:

$$\overrightarrow{u} \otimes \overrightarrow{w} = \sum_{ij} c_i^u c_j^w \overrightarrow{n_i} \otimes \overrightarrow{n_j}$$

Equipped with this definition, we can formulate the compact form of $\overrightarrow{v}$:

$$\begin{aligned}\overrightarrow{v} &= \sum_l \sum_{ik} c_i^{SUBJ_l} c_k^{OBJ_l} \overrightarrow{n_i} \otimes \overrightarrow{n_k} \\ &= \sum_l \overrightarrow{SUBJ_l} \otimes \overrightarrow{OBJ_l}\end{aligned}$$

In short, we are only required to iterate through the corpus once, taking for each instance of a transitive verb v the kronecker product of its subject and object, and summing these across all instances of v. It is simple to see that no information was discarded relative to the previous definition of $\overrightarrow{v}$: The dimensionality reduction by a factor of $dim(N)^2$ simply discards all basis elements for which the basis weight was 0 by default.

This raises a small problem though: This compact representation can no longer be used in the compositional mechanism presented in Section 3.3, as the dimensions of $\overrightarrow{v}$ no longer match those that it is required to have according to its syntactic type. However, a solution can be devised if we return to the sample calculation shown in Section 3.3.2 of the composition of a transitive verb with its arguments. The composition reduces as follows:

$$(\epsilon_N \otimes 1_S \otimes \epsilon_N)(\overrightarrow{SUBJ} \otimes \overrightarrow{v} \otimes \overrightarrow{OBJ}) = \sum_{ikm} c_i^{SUBJ} c_{ikm}^{v} c_m^{OBJ} \overrightarrow{s_k}$$

where the verb v is represented in its non-compact form. If we introduce the compactness given to us by our isomorphism $S_T \cong N \otimes N$ we can express this as

$$\overrightarrow{SUBJ\ v\ OBJ} = \sum_{im} c_i^{SUBJ} c_{im}^{v} c_m^{OBJ} \overrightarrow{n_i} \otimes \overrightarrow{n_m}$$

where v is represented in its compact form. Furthermore, by introducing the component wise multiplication operation $\odot$:

$$\overrightarrow{u} \odot \overrightarrow{v} = \sum_i c_i^u c_i^v \overrightarrow{n_i}$$

we can show the general form of transitive verb composition with the reduced verb representation to be as follows:

$$\begin{aligned}\overrightarrow{SUBJ\ v\ OBJ} &= \sum_{im} c_i^{SUBJ} c_{im}^{v} c_m^{OBJ} \overrightarrow{n_i} \otimes \overrightarrow{n_m} \\ &= \left(\sum_{im} c_{im}^{v} \overrightarrow{n_i} \otimes \overrightarrow{n_m}\right) \odot \left(\sum_{im} c_i^{SUBJ} c_m^{OBJ} \overrightarrow{n_i} \otimes \overrightarrow{n_m}\right) \\ &= \overrightarrow{v} \odot \left(\overrightarrow{SUBJ} \otimes \overrightarrow{OBJ}\right)\end{aligned}$$

To summarize what we have done with transitive verbs:

1. We have treated them as functions, taking two nouns and returning a sentence in a space $S_T \cong N \otimes N$.
2. We have built them by counting what properties of subject and object noun vectors are related by the verb in transitive sentences in a corpus.
3. We have assumed that output properties were a function of input properties by the use of the Δ_{ijk} function—for example, the weights associated with n_i from the subject argument and with n_k from the object argument only affect the "output" weight for the sentence basis element $\overrightarrow{s_j} = \overrightarrow{n_i} \otimes \overrightarrow{n_k}$.
4. We have shown that this leads to a compact representation of the verb's semantic form, and an optimized learning procedure.
5. We have shown that the composition operations of our formalism can be adapted to this compact representation.

The compact representation and amended composition operation hinge on the choice of $S_T \cong N \otimes N$ as output type for $N \otimes N$ as an input type (the pair of arguments the verb takes), justifying our choice of a transitive sentence space. In the intransitive case, the same phenomenon can be observed, since such a verb takes as argument a vector in N and produces a vector in $S_I \cong N$. Furthermore, our choice to make all other types dependent on one base type—namely, n (with associated semantic space N)—yields the same property for every relational word we wish to learn: The output type is the same as the concatenated (on the syntactic level) or tensored (on the semantic level) input types. It is this symmetry between input and output types that guarantees that any m-ary relation, expressed in the original formulation as an element of tensor space $\underbrace{N \otimes \ldots \otimes N}_{2m}$, has a compact representation in $\underbrace{N \otimes \ldots \otimes N}_{m}$, where the ith basis weight of the reduced representation stands for the degree to which the ith element of the input vector affects the ith element of the output vector.

This allows the specification of the generalized learning algorithm for reduced representations, first presented in Grefenstette and Sadrzadeh (2011a), which is as follows. Each relational word P with grammatical type π and m adjoint types $\alpha_1, \alpha_2, \cdots, \alpha_m$ is encoded as an $(r \times \ldots \times r)$ multi-dimensional array with m degrees of freedom (i.e., a rank m tensor). Because our vector space N has a fixed basis, each such array is represented in vector form as follows:

$$\overrightarrow{\mathrm{P}} = \sum_{\underbrace{ij\cdots\zeta}_{m}} c_{ij\cdots\zeta} \underbrace{(\overrightarrow{n}_i \otimes \overrightarrow{n}_j \otimes \cdots \otimes \overrightarrow{n}_\zeta)}_{m}$$

This vector lives in the tensor space $\underbrace{N \otimes N \otimes \cdots \otimes N}_{m}$. Each $c_{ij\cdots\zeta}$ is computed according to the procedure described in Figure 2.

Linear algebraically, this procedure corresponds to computing the following

$$\overrightarrow{\mathrm{P}} = \sum_k \left(\overrightarrow{\mathrm{w}}_1 \otimes \overrightarrow{\mathrm{w}}_2 \otimes \cdots \otimes \overrightarrow{\mathrm{w}}_m\right)_k$$

1) Consider a sequence of words containing a relational word P and its arguments w_1, $w_2, \cdots, w_m$, occurring in the same order as described in P's grammatical type π. Refer to these sequences as P-relations. Suppose there are k of them.
2) Retrieve the vector $\overrightarrow{w}_l$ of each argument w_l.
3) Suppose w_1 has weight c_i^1 on basis vector $\overrightarrow{n}_i$, w_2 has weight c_j^2 on basis vector $\overrightarrow{n}_j, \cdots$, and w_m has weight c_ζ^m on basis vector $\overrightarrow{n}_\zeta$. Multiply these weights

$$c_i^1 \times c_j^2 \times \cdots \times c_\zeta^m$$

4) Repeat the above steps for all the k P-relations, and sum the corresponding weights

$$c_{ij\cdots\zeta} = \sum_k \left(c_i^1 \times c_j^2 \times \cdots \times c_\zeta^m\right)_k$$

Figure 2
Procedure for learning weights for matrices of words P with relational types π of m arguments.

The general formulation of composing a relational word P with its arguments $arg_1, \ldots, \text{arg}_m$ is now expressed as

$$\overrightarrow{P} \odot \left(\overrightarrow{arg_1} \otimes \ldots \otimes \overrightarrow{arg_m}\right)$$

For example, the computation of *furry cats nag angry dogs* would correspond to the following operations:

$$\overrightarrow{\text{furry cats nag angry dogs}} = \overrightarrow{\text{nag}} \odot \left(\left(\overrightarrow{\text{furry}} \odot \overrightarrow{\text{cat}}\right) \otimes \left(\overrightarrow{\text{angry}} \odot \overrightarrow{\text{dog}}\right)\right)$$

This generalized learning algorithm effectively extends the coverage of our approach to any sentence for which we have a pregroup parse (e.g., as might be obtained by our translation mechanism from CCG). For example, determiners would have a type nn^l, allowing us to model them as matrices in $N \otimes N$; adverbs would be of type $(n^r s)^r (n^r s)$, and hence be elements of $S \otimes N \otimes N \otimes S$. We could learn them using the procedure defined earlier, although for words like determiners and conjunctions, it is unclear whether we would want to learn such logical words or design them by hand, as was done in Coecke, Sadrzadeh, and Clark (2010) for negation. Nonetheless, the procedure given here allows us to generalize the work presented in this article to sentences of any length or structure based on the pregroup types of the words they contain.

4.4 Example

To conclude this section with an example taken from Grefenstette and Sadrzadeh (2011a), we demonstrate how the meaning of the word *show* might be learned from a corpus and then composed.

Suppose there are two instances of the verb *show* in the corpus:

s_1 = table show result
s_2 = map show location

The vector of *show* is

$$\overrightarrow{\text{show}} = \overrightarrow{\text{table}} \otimes \overrightarrow{\text{result}} + \overrightarrow{\text{map}} \otimes \overrightarrow{\text{location}}$$

Consider a vector space N with four basis vectors *far*, *room*, *scientific*, and *elect*. The TF/IDF-weighted values for vectors of selected nouns (built from the BNC) are as shown in Table 1.

Part of the matrix compact representation of *show* is presented in Table 2.

Table 1
Sample weights for selected noun vectors.

i	$\overrightarrow{\mathbf{n_i}}$	table	map	result	location	master	dog
1	far	6.6	5.6	7	5.9	4.3	5.5
2	room	27	7.4	1.0	7.3	9.1	7.6
3	scientific	0	5.4	13	6.1	4.1	0
4	elect	0	0	4.2	0	6.2	0

Table 2
Sample semantic matrix for *show*.

	far	room	scientific	elect
far	$\boxed{79.24}$	47.41	119.96	27.72
room	$\boxed{232.66}$	80.75	396.14	113.2
scientific	32.94	31.86	32.94	0
elect	0	0	0	0

As a sample computation, the weight c_{11} for vector $(\overrightarrow{n_1}, \overrightarrow{n_1})$, that is, $(\overrightarrow{\text{far}}, \overrightarrow{\text{far}})$, is computed by multiplying weights of *table* and *result* on $\overrightarrow{\text{far}}$ (6.6×7), multiplying weights of *map* and *location* on $\overrightarrow{\text{far}}$ (5.6×5.9), and then adding these ($46.2 + 33.04$) and obtaining the total weight 79.24. Similarly, the weight c_{21} for vector $(\overrightarrow{n_2}, \overrightarrow{n_1})$, that is, $(\overrightarrow{\text{room}}, \overrightarrow{\text{far}})$, is computed by multiplying the weight of $\overrightarrow{\text{room}}$ for *table* by the weight of $\overrightarrow{\text{far}}$ for *result* (27×7), then multiplying the weight of $\overrightarrow{\text{room}}$ for *map* by the weight of $\overrightarrow{\text{far}}$ for *location* (7.4×5.9), and then adding these ($189 + 43.66$) to obtain the total weight of 232.66.

We now wish to compute the vector for the sentence *[the] master shows [his] dog*, omitting the determiner and possessive for simplicity, as we have left open the question as to whether or not we would want to learn them using the generalized procedure from Section 4.3 or specify them by design due to their logical structure. The calculation according to the vectors in Table 1 and the matrix in Table 2 will be:

$$
\begin{aligned}
&\overrightarrow{\text{master show dog}} \\
&\quad = \overrightarrow{\text{show}} \odot \left(\overrightarrow{\text{master}} \otimes \overrightarrow{\text{dog}}\right) \\
&\quad = \begin{bmatrix} 79.24 & 47.41 & 119.96 & 27.72 \\ 232.66 & 80.75 & 396.14 & 113.2 \\ 32.94 & 31.86 & 32.94 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \odot \begin{bmatrix} 23.65 & 32.68 & 0 & 0 \\ 50.05 & 69.16 & 0 & 0 \\ 22.55 & 31.16 & 0 & 0 \\ 34.1 & 47.12 & 0 & 0 \end{bmatrix} \\
&\quad = \begin{bmatrix} 1{,}874.03 & 1{,}549.36 & 0 & 0 \\ 11{,}644.63 & 5{,}584.67 & 0 & 0 \\ 742.80 & 992.76 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}
\end{aligned}
$$

Row-wise, flattening the final matrix representation gives us the result we seek, namely, the sentence vector in S_T for *[the] master showed [his] dog*:

$$\overrightarrow{\text{master show dog}} = [1{,}874, 1{,}549, 0, 0, 11{,}645, 5{,}585, 0, 0, 743, 993, 0, 0, 0, 0, 0, 0]$$

5. Experiments

Evaluating compositional models of semantics is no easy task: First, we are trying to evaluate how well the compositional process works; second, we also are trying to determine how useful the final representation—the *output* of the composition—is, relative to our needs.

The scope of this second problem covers most phrase and sentence-level semantic models, from bag-of-words approaches in information retrieval to logic-based formal semantic models, via language models used for machine translation. It is heavily task dependent, in that a representation that is suitable for machine translation may not be appropriate for textual inference tasks, and one that is appropriate for IR may not be ideal for paraphrase detection. Therefore this aspect of semantic model evaluation ideally should take the form of application-oriented testing. For instance, to test semantic representations designed for machine translation purposes, we should use a machine translation evaluation task.

The DisCoCat framework (Clark, Coecke, and Sadrzadeh 2008; Coecke, Sadrzadeh, and Clark 2010), described in Section 3, allows for the composition of any words of any syntactic type. The general learning algorithm presented in Section 4.3 technically can be applied to learn and model relations of any semantic type. However, many open questions remain, such as how to deal with logical words, determiners, and quantification, and how to reconcile the different semantic types used for sentences with transitive and intransitive sentences. We will leave these questions for future work, briefly discussed in Section 6. In the meantime, we concretely are left with a way of satisfactorily modeling only simple sentences without having to answer these bigger questions.

With this in mind, in this section we present a series of experiments centered around evaluating how well various models of semantic vector composition perform (along with the one described in Section 4) in a phrase similarity comparison task. This task aims to test the quality of a compositional process by determining how well it forms a clear joint meaning for potentially ambiguous words. The intuition here is that tokens, on their own, can have several meanings; and that it is through the compositional process—through giving them *context*—that we understand their specific meaning. For example, *bank* itself could (among other meanings) mean a river bank or a financial bank; yet in the context of a sentence such as *The bank refunded the deposit*, it is likely we are talking about the financial institution.

In this section, we present three data sets designed to evaluate how well word-sense disambiguation occurs as a byproduct of composition. We begin by describing the first data set, based around noun-intransitive verb phrases, in Section 5.1. In Section 5.2, we present a data set based around short transitive-verb phrases (a transitive verb with subject and object). In Section 5.3, we discuss a new data set, based around short transitive-verb phrases where the subject and object are qualified by adjectives. We leave discussion of these results for Section 6.

5.1 First Experiment

This first experiment, originally presented in Mitchell and Lapata (2008), evaluates the degree to which an ambiguous intransitive verb (e.g., *draws*) is disambiguated by combination with its subject.

Data set Description. The data set[4] comprises 120 pairs of intransitive sentences, each of the form NOUN VERB. These sentence pairs are generated according to the following

[4] Available at `http://homepages.inf.ed.ac.uk/s0453356/results`.

procedure, which will be the basis for the construction of the other data sets discussed subsequently:

1. A number of ambiguous intransitive verbs (15, in this case) are selected from frequently occurring verbs in the corpus.
2. For each verb V, two mutually exclusive synonyms V_1 and V_2 of the verb are produced, and each is paired with the original verb separately (for a total of 30 verb pairs). These are generated by taking maximally distant pairs of synonyms of the verb on WordNet, but any method could be used here.
3. For each *pair of verb pairs* (V, V_1) and (V, V_2), two frequently occurring nouns N_1 and N_2 are picked from the corpus, one for each synonym of V. For example, if V is *glow* and the synonyms V_1 and V_2 are *beam* and *burn*, we might choose *face* as N_1, because a face glowing and a face beaming mean roughly the same thing; and *fire* as N_2, because a fire glowing and a fire burning mean roughly the same thing.
4. By combining the nouns with the verb pairs, we form two high similarity triplets (V, V_1, N_1) and (V, V_2, N_2), and two low similarity triplets (V, V_1, N_2) and (V, V_2, N_1).

The last two steps can be repeated to form more than four triplets per pair of verb pairs. In Mitchell and Lapata (2008), eight triplets are generated for each pair of verb pairs, obtaining a total of 120 triplets from the 15 original verbs. Each triplet, along with its HIGH or LOW classification (based on the choice of noun for the verb pair) is an entry in the data set, and can be read as a pair of sentences: (V, V_i, N) translates into the intransitive sentences $N\ V$ and $N\ V_i$.

Finally, the data set is presented, without the HIGH/LOW ratings, to human annotators. These annotators are asked to rate the similarity of meaning of the pairs of sentences in each entry on a scale of 1 (low similarity) to 7 (high similarity). The final form of the data set is a set of lines each containing:

- A (V, V_i, N) triplet.
- A HIGH or LOW label for that triplet.
- An annotator identifier and the annotator's score for that triplet.

Sample sentences from this data set are shown in Table 3.

Table 3
Example entries from the intransitive data set without annotator score, first experiment.

Sentence 1	Sentence 2
butler bow	butler submit
head bow	head stoop
company bow	company submit
government bow	government stoop

Evaluation Methodology. This data set is used to compare various compositional distributional semantic models, according to the following procedure:

1. For each entry in the data set, the representation of the two sentences $N\ V$ and $N\ V_i$ formed from the entry triple (V, V_i, N), which we will name S_1 and S_2, is constructed by the model.
2. The similarity of these sentences according to the model's semantic distance measure constitutes the *model score* for the entry.
3. The rank correlation of entry model scores against entry annotator scores is calculated using Spearman's rank correlation coefficient ρ.

The Spearman ρ scores are values between -1 (perfect negative correlation) and 1 (perfect correlation). The higher the ρ score, the higher the compositional model can be said to produce sentence representations that match human understanding of sentence meaning, when it comes to comparing the meaning of sentences. As such, we will rank the models evaluated using the task by decreasing order of ρ score.

One of the principal appealing features of Spearman's ρ is that the coefficient is rank-based: It does not require models' semantic similarity metrics to be normalized for a comparison to be made. One consequence is that a model providing excellent rank correlation with human scores, but producing model scores on a small scale (e.g., values between 0.5 and 0.6), will obtain a higher ρ score than a model producing model scores on a larger scale (e.g., between 0 and 1) but with less perfect rank correlation. If we wished to then use the former model in a task requiring some greater degree of numerical separation (let us say 0 for non-similar sentences and 1 for completely similar sentences), we could simply renormalize the model scores to fit the scale. By eschewing score normalization as an evaluation factor, we minimize the risk of erroneously ranking one model over another.

Finally, in addition to computing the rank alignment coefficient between model scores and annotator scores, Mitchell and Lapata (2008) calculate the mean model scores for entries labeled HIGH, and for entries labeled LOW. This information is reported in their paper as additional means for model comparison. However, for the same reason we considered Spearman's ρ to be a fair means of model comparison—namely, in that it required no model score normalization procedure and thus was less likely to introduce error by adding such a degree of freedom—we consider the HIGH/LOW means to be inadequate grounds for comparison, precisely because it *requires* normalized model scores for comparison to be meaningful. As such, we will not include these mean scores in the presentation of this, or any further experiments in this article.

Models Compared. In this experiment, we compare the best and worst performing models of Mitchell and Lapata (2008) to our own. We begin by building a vector space W for all words in the corpus, using standard distributional semantic model construction procedures (n-word window) with the parameters of Mitchell and Lapata (2008). These parameters are as follows: The basis elements of this vector space are the 2,000 most frequently occurring words in the BNC, excluding common stop words. For our evaluation, the corpus was lemmatized using Carroll's Morpha (Minnen, Carroll, and Pearce 2001), which was applied as a byproduct of our parsing of the BNC with C&C Tools (Curran, Clark, and Bos 2007).

The context of each occurrence of a word in the corpus was defined to be five words on either side. After the vector for each word w was produced, the basis weights c_i^w associated with context word b_i were normalized by the following ratio of probabilities weighting scheme:

$$c_i^w = \frac{P(b_i|w)}{P(w)}$$

Let $\overrightarrow{verb}$ and $\overrightarrow{noun}$ be the lexical semantic vectors for the verb and the noun, from W. It should be evident that there is no significant difference in using W for nouns in lieu of building a separate vector space N strictly for noun vectors.

As a first baseline, **Verb Baseline**, we ignored the information provided by the noun in constructing the sentence representation, effectively comparing the semantic content of the verbs:

$$\textbf{Verb Baseline}: \ \overrightarrow{noun\ verb} = \overrightarrow{verb}$$

The models from Mitchell and Lapata (2008) we evaluate here are those which were strictly unsupervised (i.e., no free parameters for composition). These are the additive model **Add**, wherein

$$\textbf{Add}: \ \overrightarrow{noun\ verb} = \overrightarrow{noun} + \overrightarrow{verb}$$

and the multiplicative model **Multiply**, wherein

$$\textbf{Multiply}: \ \overrightarrow{noun\ verb} = \overrightarrow{noun} \odot \overrightarrow{verb}$$

Other models with parameters that must be optimized against a held-out section of the data set are presented in Mitchell and Lapata (2008). We omitted them here principally because they do not perform as well as **Multiply**, but also because the need to optimize the free parameters for this and other data sets makes fair comparison with completely unsupervised models more difficult, and less fair.

We also evaluate the **Categorical** model from Section 4 here, wherein

$$\textbf{Categorical}: \ \overrightarrow{noun\ verb} = \overrightarrow{verb_{cat}} \odot \overrightarrow{noun}$$

where $\overrightarrow{verb_{cat}}$ is the compact representation of the relation the verb stands for, computed according to the procedure described in Section 4.3. It should be noted that for the case of intransitive verbs, this composition operation is mathematically equivalent to that of the **Multiply** model (as component-wise multiplication $\odot$ is a commutative operation), the difference being the learning procedure for the verb vector.

For all such vector-based models, the similarity of the two sentences s_1 and s_2 is taken to be the cosine similarity of their vectors, defined in Section 2.2:

$$similarity(s_1, s_2) = cosine(\overrightarrow{s_1}, \overrightarrow{s_2})$$

In addition to these vector-based methods, we define an additional baseline and an upper-bound. The additional baseline, **Bigram Baseline**, is a bigram-based language model trained on the BNC with SRILM (Stolcke 2002), using the standard language model settings for computing log-probabilities of bigrams. To determine the semantic similarity of sentences $s_1 = noun\ verb_1$ and $s_2 = noun\ verb_2$, we assumed sentences have mutually conditionally independent properties, and computed the joint probability:

$$similarity(s_1, s_2) = logP(s_1 \wedge s_2) = log(P(s_1)P(s_2)) = logP(s_1) + logP(s_2)$$

The reasoning behind this baseline is as follows. The sentence formed by combining the first verb with its arguments is, by the design of this data set, a semantically coherent sentence (e.g., *the head bowed* and *the government bowed* both make sense). We therefore expect language models to treat this sort of bigram as having a higher probability than bigrams that are not semantically coherent sentences (and therefore unlikely to be observed in the corpus). A similar (relative) high probability is to be expected when the sentence formed by taking the second verb in each entry and combining it with the verb yields a sentence similar to the first in meaning (e.g., as would be the case with *the head stooped*), whereas the probability of a semantically incoherent sentence (e.g., *the government stooped*) is expected to be low relative to that of the first sentence. By taking the sum of log probabilities of sentences, we compute the log of the product of the probabilities, which we expect to be low when the probability of the second sentence is low, and high when it is high, with the probability of the first sentence acting as a normalizing factor. To summarize, while this bigram-based measure only tracks a tangential aspect of semantic similarity, it can be one which plays an artificially important role in experiments with a predetermined structure such as the one described in this section. For this reason, we use this bigram baseline for this experiment and all that follow.

The upper bound of the data set, **UpperBound**, was taken to be the inter-annotator agreement: the average of how each annotator's score aligns with other annotator scores, using Spearman's ρ.

Results. The results[5] of the first experiment are shown in Table 4. As expected from the fact that **Multiply** and **Categorical** differ only in how the verb vector is learned, the results of these two models are virtually identical, outperforming both baselines and the **Additive** model by a significant margin. However, the distance from these models to the upper bound is even greater, demonstrating that there is still a lot of progress to be made.

5.2 Second Experiment

The first experiment was followed by a second similar experiment, in Mitchell and Lapata (2010), covering different sorts of composition operations for binary combinations of syntactic types (adjective-noun, noun-noun, verb-object). Such further experiments are interesting, but rather than continue down this binary road, we now turn to the development of our second experiment, involving sentences with larger syntactic structures, to examine how well various compositional models cope with more complex syntactic and semantic relations.

[5] The results were state of the art when the experiments were run in 2011. The binary composition data set used here are popular; now there are a host of new state-of-the-art systems available in the literature. We invite the reader to check references to Mitchell and Lapata (2008) to find the current state of the art.

Table 4
Model correlation coefficients with human judgments, first experiment. $p < 0.05$ for each ρ.

Model	ρ
Verb Baseline	0.08
Bigram Baseline	0.02
Add	0.04
Multiply	0.17
Categorical	**0.17**
UpperBound	0.40

This second experiment, which we initially presented in Grefenstette and Sadrzadeh (2011a), is an extension of the first in the case of sentences centered around transitive verbs, composed with a subject and an object. The results of the first experiment did not demonstrate any difference between the multiplicative model, which takes into account no syntactic information or word ordering, and our syntactically motivated categorical compositional model. By running the same experiment over a new data set, where the relations expressed by the verb have a higher arity than in the first, we hope to demonstrate that added structure leads to better results for our syntax-sensitive model.

Data Set Description. The construction procedure for this data set[6] is almost exactly as for the first data set, with the following differences:

- Verbs are transitive instead of intransitive.
- We arbitrarily took 10 verbs from the most frequent verbs in the BNC, and for each verb, took two maximally distant synonyms (again, using WordNet) to obtain 10 pairs of pairs.
- For each pair of verb pairs, we selected a set of subject and object nouns to use as context, as opposed to just a subject noun.
- Each subject-object pair was manually chosen so that one of the verb pairs would have high similarity in the context of that subject and object, and the other would have low similarity. We used these choices to annotate the entries with HIGH and LOW tags.
- Each combination of a verb pair with a subject-object pair constitutes an entry of our data set, of which there are 200.

As a form of quality control, we inserted "gold standard" sentences in the form of identical sentence pairs and rejected annotators who did not score these gold standard

[6] The data set, reannotated by Turkers in 2013, is available at `http://www.cs.ox.ac.uk/activities/compdistmeaning/GS2013data.txt`.

Table 5
Example entries from the transitive data set without annotator score, second experiment.

Sentence 1	Sentence 2	HIGH-LOW Tag
man draw sword	man attract sword	LOW
report draw attention	report attract attention	HIGH
man draw sword	man depict sword	HIGH
report draw attention	report depict attention	LOW

sentences with a high score of 6 or 7.[7] Lemmatized sentences from sample entries of this data set and our HIGH-LOW tags for them are shown in Table 5.

The data set was passed to a group of 50 annotators on Amazon Mechanical Turk, as for the previous data set. The annotators were shown, for each entry, a pair of sentences created by adding *the* in front of the subject and object nouns and putting the verbs in the past tense,[8] and were instructed to score each pair of sentences on the same scale of 1 (not similar in meaning) to 7 (similar in meaning), based on how similar in meaning they believed the sentence pair was.

Evaluation Methodology. The methodology for this experiment is exactly that of the previous experiment. Models compositionally construct sentence representations, and compare them using a distance metric (all vector-based models once again used cosine similarity). The rank correlation of models scores with annotator scores is calculated using Spearman's ρ, which is in turn used to rank models.

Models Compared. The models compared in this experiment are those of the first experiment, with the addition of an extra trigram-based baseline (trained with SRILM, using the addition of log-probability of the sentence as a similarity metric), and a variation on our categorical model, presented subsequently. With W as the distributional semantic space for all words in the corpus, trained using the same parameters as in the first experiment, and $\overrightarrow{subj}, \overrightarrow{verb}, \overrightarrow{object} \in W$ as the vectors for subject, verb, and object of a sentence, respectively, and with $verb_{cat}$ as the compact representation of a transitive verb learned using the algorithm presented in this paper, we have the following compositional methods:

$$\textbf{Add}: \ \overrightarrow{subject\ verb\ object} = \overrightarrow{subject} + \overrightarrow{verb} + \overrightarrow{object}$$

$$\textbf{Multiply}: \ \overrightarrow{subject\ verb\ object} = \overrightarrow{subject} \odot \overrightarrow{verb} \odot \overrightarrow{object}$$

$$\textbf{Categorical}: \ \overrightarrow{subject\ verb\ object} = \overrightarrow{verb_{cat}} \odot \left(\overrightarrow{subject} \otimes \overrightarrow{object}\right)$$

$$\textbf{Kronecker}: \ \overrightarrow{subject\ verb\ object} = \left(\overrightarrow{verb} \otimes \overrightarrow{verb}\right) \odot \left(\overrightarrow{subject} \otimes \overrightarrow{object}\right)$$

[7] During the 2013 reannotation of this data set, we rejected no Turker contributions, as the answers to the gold standard sentence pairs were aligned with our expectations. We attribute this to our adding the requirement that Turkers be based in the US or the UK and have English as the first language.

[8] For example, the entry *draw table eye depict* would yield the sentences *The table drew the eye* and *The table depicted the eye.*

All of these models have been explained earlier, with the exception of **Kronecker**. We first presented this new addition in Grefenstette and Sadrzadeh (2011b), where we observed that the compact representation of a verb in the DisCoCat framework, under the assumptions presented in Section 4, can be viewed as $dim(N) \times dim(N)$ matrices in $N \otimes N$. We considered alternatives to the algorithm presented earlier for the construction of such matrices, and were surprised by the results of the Kronecker method, wherein we replaced the matrix learned by our algorithm with the Kronecker product of the lexical semantic vectors for the verb. Further analysis performed since the publication of that paper can help to understand why this method might work. Using the following property, for any vectors $\overrightarrow{a}, \overrightarrow{b}, \overrightarrow{c}, \overrightarrow{d}$ in a vector space

$$(\overrightarrow{a} \otimes \overrightarrow{b}) \odot (\overrightarrow{c} \otimes \overrightarrow{d}) = (\overrightarrow{a} \odot \overrightarrow{c}) \otimes (\overrightarrow{b} \odot \overrightarrow{d})$$

we can see that the **Kronecker** model's composition operation can be expressed as

$$\textbf{Kronecker}: \overrightarrow{subject\ verb\ object} = \left(\overrightarrow{verb} \odot \overrightarrow{subject}\right) \otimes \left(\overrightarrow{verb} \odot \overrightarrow{object}\right)$$

Bearing in mind that the cosine measure we are using as a similarity metric is equivalent to the inner product of two vectors normalized by the product of their length

$$cosine(\overrightarrow{a}, \overrightarrow{b}) = \frac{\langle \overrightarrow{a} | \overrightarrow{b} \rangle}{\| \overrightarrow{a} \| \times \| \overrightarrow{b} \|}$$

and the following property of the inner product of kronecker products

$$\langle \overrightarrow{a} \otimes \overrightarrow{b} | \overrightarrow{c} \otimes \overrightarrow{d} \rangle = \langle \overrightarrow{a} | \overrightarrow{c} \rangle \times \langle \overrightarrow{b} | \overrightarrow{d} \rangle$$

we finally observe that comparing two sentences under **Kronecker** corresponds to the following computation:

$$\begin{aligned}
&cosine(\overrightarrow{subject\ verb_1\ object}, \overrightarrow{subject\ verb_2\ object}) \\
&\quad = \alpha \left\langle \left(\overrightarrow{verb_1} \otimes \overrightarrow{verb_1}\right) \odot \left(\overrightarrow{subject} \otimes \overrightarrow{object}\right) \middle| \left(\overrightarrow{verb_2} \otimes \overrightarrow{verb_2}\right) \odot \left(\overrightarrow{subject} \otimes \overrightarrow{object}\right) \right\rangle \\
&\quad = \alpha \left\langle \left(\overrightarrow{verb_1} \odot \overrightarrow{subject}\right) \otimes \left(\overrightarrow{verb_1} \odot \overrightarrow{object}\right) \middle| \left(\overrightarrow{verb_2} \odot \overrightarrow{subject}\right) \otimes \left(\overrightarrow{verb_2} \odot \overrightarrow{object}\right) \right\rangle \\
&\quad = \alpha \left\langle \left(\overrightarrow{verb_1} \odot \overrightarrow{subject}\right) \middle| \left(\overrightarrow{verb_2} \odot \overrightarrow{subject}\right) \right\rangle \left\langle \left(\overrightarrow{verb_1} \odot \overrightarrow{object}\right) \middle| \left(\overrightarrow{verb_2} \odot \overrightarrow{object}\right) \right\rangle
\end{aligned}$$

where α is the normalization factor

$$\alpha = \frac{1}{\|\overrightarrow{subject\ verb_1\ object}\| \times \|\overrightarrow{subject\ verb_2\ object}\|}$$

We note here that the **Kronecker** is effectively a parallel application of the **Multiply** model, combining the subject and verb, and object and verb separately. Within the context of this task, the comparison of two sentences boils down to how well each verb combines with the subject multiplied by how well it combines with the object, with the joint product forming the overall model score. In short, it more or less constitutes the introduction of some mild syntactic sensitivity into the multiplicative model of Mitchell and Lapata (2008), although it remains to be seen how well this scales with syntactic complexity (e.g., extended to ditransitive verbs, or other relations of higher arity).

The **UpperBound** here is, again, the inter-annotator agreement, calculated by computing the pairwise alignment of each annotator with every other, and averaging the resulting rank correlation coefficients.

Results. The results of the second experiment are shown in Table 6. The baseline scores fall in the 0.14–0.16 range, with best results being obtained for the **Bigram Baseline** and **Trigram Baseline**, the difference between both models not being statistically significant. The additive model **Add** performs on par with the first experiment. The multiplicative model **Multiply** and our **Categorical** model perform on par with the version used for the intransitive experiment, but obtain a score comparable to the baselines. The best performing model here is our newly introduced **Kronecker** model, which leads the pack by a steady margin, with a score of 0.26. The inter-annotator agreement **UpperBound** is much higher in this experiment than in the previous experiment, indicating even more room for improvement.

We therefore learn here that the **Categorical** model continues to operate on par with the multiplicative model, and that the **Kronecker** model provides the highest results, while being as simple in its construction as the multiplicative model, requiring only the learning of lexical vectors for the verb.

5.3 Third Experiment

The third and final experiment we present is a modified version of the second data set presented earlier, where the nouns in each entry are under the scope of adjectives applied to them.The intuition behind the data sets presented in Section 5.1 and Section 5.2 was that ambiguous verbs are disambiguated through composition with nouns. These nouns themselves may also be ambiguous, and a good compositional model will be capable of separating the noise produced by other meanings through its compositional mechanism to produce unambiguous phrase representations. The intuition behind this data set is similar, in that adjectives provide both additional information for disambiguation of the nouns they apply to, but also additional semantic noise. Therefore, a

Table 6
Model correlation coefficients with human judgments, second experiment. $p < 0.05$ for each ρ.

Model	ρ
Verb Baseline	0.13
Bigram Baseline	0.16
Trigram Baseline	0.15
Add	0.10
Multiply	0.16
Categorical	0.16
Kronecker	**0.26**
UpperBound	0.62

Table 7
Example entries from the adjective-transitive data set without annotator score, third experiment.

Sentence 1	Sentence 2
statistical table show good result	statistical table express good result
statistical table show good result	statistical table depict good result

good model will also be able to separate the useful information of the adjective from its semantic noise when composing it with its argument, in addition to doing this when composing the noun phrases with the verb.

Data Set Description. The construction procedure for this data set was to take the data set from Section 5.2, and, for each entry, add a pair of adjectives from those most frequently occurring in the corpus. The first adjective from the pair is applied to the first noun (subject) of the entry when forming the sentences, and the second adjective is applied to the second noun (object). For each entry, we chose adjectives which best preserved the meaning of the phrase constructed by combining the first verb with its subject and object.

This new data set[9] was then annotated again by a group of 50 annotators using Amazon's Mechanical Turk service. The annotators were shown, for each entry, a pair of sentences created by adding *the* in front of the subject and object noun phrases and putting the verbs in the past tense,[10] and asked to give each sentence pair a meaning similarity score between 1 and 7, as for the previous data sets. We applied the same quality control mechanism as in the second experiment. Some 94 users returned annotations, of which we kept 50 according to our gold standard tests. We are unaware of whether or not it was applied in the production of the first data set, but believe that this can only lead to the production of higher quality annotations.

Sample sentences from this data set are shown in Table 7.

Evaluation Methodology. The evaluation methodology in this experiment is identical to that of the previous experiments.

Models Compared. In this experiment, in lieu of simply comparing compositional models "across the board" (e.g., using the multiplicative model for both adjective-noun composition and verb-argument composition), we experimented with different *combinations of models*. This evaluation procedure was chosen because we believe that adjective-noun composition need not necessarily be the same kind of compositional process as subject-verb-object composition, and also because different models may latch onto different semantic features during the compositional process, and it would be interesting to see what model mixtures work well together. Naturally, this is not a viable approach to *selecting* composition operations in general, as we will not have the luxury of trying every combination of composition operations for every combination of syntactic types, but it is worthwhile performing these tests to at least verify the hypothesis that

[9] Available at `http://www.cs.ox.ac.uk/activities/compdistmeaning/GS2012data.txt`.

[10] For example, the entry *statistical table show express good result* would yield the sentences *The statistical table showed the good result* and *The statistical table expressed the good result*.

operation-specific composition operations (or parameters) is a good thing. Notably, this idea has been explored very recently, within the context of deep learning networks, by Chen et al. (2013).

Each mixed model has two components: a verb-argument composition model and a adjective-noun composition model. For verb-argument composition, we used the three best models from the previous experiment, namely, **Multiply**, **Categorical**, and **Kronecker**. For adjective composition we used three different methods of adjective-noun composition. With $\overrightarrow{adjective}$ and $\overrightarrow{noun}$ being the vectors for an adjective and a noun in our distributional lexical semantic space W (built using the same procedure as the previous experiments) and $\overrightarrow{adj_{cat}}$ being the compact representation in the **Categorical** model, built according to the algorithm from Section 4.3, we have the following models:

$$\textbf{AdjMult}: \ \overrightarrow{adjective\ noun} = \overrightarrow{adjective} \odot \overrightarrow{noun}$$

$$\textbf{Categorical}: \ \overrightarrow{adjective\ noun} = \overrightarrow{adjective_{cat}} \odot \overrightarrow{noun}$$

The third model, **AdjNoun**, is a holistic (non-compositional) model, wherein the adjective-noun compound was treated as a single token, as its semantic vector $\overrightarrow{(adjective\ noun)_{lex}} \in W$ was learned from the corpus using the same learning procedure applied to construct other vectors in W. Hence, the model defines adjective-noun "composition" as:

$$\textbf{AdjNoun}: \ \overrightarrow{adjective\ noun} = \overrightarrow{(adjective\ noun)_{lex}}$$

In addition to these models, we also evaluated three baselines: **Verb Baseline**, **Bigram Baseline**, and **Trigram Baseline**. As in previous experiments, the verb baseline uses the verb vector as sentence vector, ignoring the information provided by other words. The bigram and trigram baselines are calculated from the same language model as used in the second experiment. In both cases, the log-probability of each sentence is calculated using SRLIM, and the sum of log-probabilities of two sentences is used as a similarity measure.

Finally, for comparison, we also considered the full additive model:

$$\textbf{Additive}: \ \overrightarrow{sentence} = \overrightarrow{adjective_1} + \overrightarrow{noun_1} + \overrightarrow{verb} + \overrightarrow{adjective_2} + \overrightarrow{noun_2}$$

Results. The results for the third experiment are shown in Table 8. The best performing adjective-noun combination operations for each verb-argument combination operation are shown in bold. Going through the combined models, we notice that in most cases the results stay the same whether the adjective-noun combination method is **AdjMult** or **CategoricalAdj**. This is because, as was shown in the first experiment, composition of a unary-relation such as an adjective or intransitive verb with its sole argument, under the categorical model with reduced representations, is mathematically equivalent to the multiplicative model. The sole difference is the way the adjective or intransitive verb vector is constructed. We note, however, that with **Categorical** as a verb-argument composition method, the **CategoricalAdj** outperforms **AdjMult** by a non-negligible margin (0.19 vs. 0.14), indicating that the difference in learning procedure can lead to different results depending on what other models it is combined with.

Table 8
Model correlation coefficients with human judgments, third experiment. $p < 0.05$ for each ρ.

Model	ρ
Verb Baseline	0.20
Bigram Baseline	0.14
Trigram Baseline	0.16
Additive	0.10
Multiplicative	
AdjMult	**0.20**
AdjNoun	0.05
CategoricalAdj	**0.20**
Categorical	
AdjMult	0.14
AdjNoun	0.16
CategoricalAdj	**0.19**
Kronecker	
AdjMult	**0.26**
AdjNoun	0.17
CategoricalAdj	**0.27**
Upperbound	0.48

Overall, the best results are obtained for **AdjMult+Kronecker** ($\rho = 0.26$) and **CategoricalAdj+Kronecker** ($\rho = 0.27$). Combinations of the adjective composition methods with other composition methods at best matches the best-performing baseline, **Verb Baseline**. In all cases, the holistic model **AdjNoun** provides the worst results.

6. Discussion

In this article, we presented various approaches to compositionality in distributional semantics. We discussed what mathematical operations could underlie vector combination, and several different ways of including syntactic information into the combinatory process. We reviewed an existing compositional framework that leverages the ability to communicate information across mathematical structures provided by category theory in order to define a general way of linking syntactic structure to syntax-sensitive composition operations. We presented concrete ways to apply this framework to linguistic tasks and evaluate it, and developed algorithms to construct semantic representations for words and relations within this framework. In this section, we first briefly comment upon the combined results of all three experiments, and then conclude by discussing what aspects of compositionality require further attention, and how experiment design should adapt towards this goal.

Results Commentary. We evaluated this framework against other unsupervised compositional distributional models, using non-compositional models and *n*-gram language models as baselines, within the context of three experiments. These experiments show that the concrete categorical model developed here, and the Kronecker-based variant presented alongside it, outperform all other models in each experiment save the first, where they perform on par with what was the leading model at the time this experiment was performed (namely, 2011). As the experiments involved progressively more syntactically complicated sentences, the increased reliability of our categorical approaches relative to competing models as sentence complexity rises seems to indicate that both the categorical and Kronecker model successfully leverage the added information provided by additional terms and syntactic structures.

The third experiment also served to show that using different combinations of composition operations, depending on the syntactic type of the terms being combined, can yield better results, and that some models combine better than others. Notably, the adjective-noun combination models **AdjMult** and **CategoricalAdj**, despite their mathematical similarity, produce noticeably different results when combined with the categorical verb-argument composition operation, while they perform equally with most other verb-argument composition operations. We can conclude that different models combine different semantic aspects more prominently than others, and that through combination we can find better performance by assuming that different kinds of composition play on different semantic properties. For example, predicates such as intersective adjectives add information to their argument (a red ball is a ball that is also red). This raises the question of how to design models of composition that systematically select which operations will match the semantic aspects of the words being combined based on their syntactic type. This is an open question, which we believe warrants further investigation.

Overall, we observe two advantages of these models over those presented in the early part of this article, and those evaluated in the later part. First, we have shown that a linguistically motivated and mathematically sound framework can be implemented and learned. Second, we showed that simple methods such as the Kronecker model perform well, especially when combined with the multiplicative model (effectively a unary instance of the Kronecker model) for sentences with adjectives. Considering the "simple is best" position recently argued for experimentally by Blacoe and Lapata (2012), this is an interesting candidate for dealing with binary relation composition.

Future Work. These experiments showcased the ability of various compositional models to produce sentence representations that were less ambiguous than the words that formed them. They also more generally demonstrated that concrete models could be built from the general categorical framework and perform adequately in simple paraphrase detection tasks. However, various aspects of compositionality were not evaluated here. First, syntax sensitivity is not as important in these experiments as it might be in "real" language. For instance, whereas models that treat adjectives, nouns, and verbs differently tended to perform better than those that did not, the actual capacity of a model to *use* the syntactic structure was not tested. For instance, models that ignore word order and syntax altogether were not particularly penalized, as might be done if some sentences were simply the reverse of another sentence they are paired with (where syntax insensitive models would erroneously give such sentence pairs a high similarity score). One way of testing word order while expanding the data set was suggested by Turney (2012), who for every entry in a phrase similarity test such as Mitchell and Lapata's, discussed herein, creates a new entry where one of the sentences has the order of its words reversed, and is assigned an artificial annotator score of 1 (the minimum). This is an interesting approach, but we would prefer to see such reversals designed into the data set and seen by annotators, for instance, in the recent data set presented in Pham et al. (2013). This data set has entries such as *guitar played man* and *man played guitar*, where the subjects and objects of verbs are indeed reversed but the two sentences do not necessarily express opposite meanings; we will be looking into expanding such data sets to ones where both sentences make sense and have opposite meanings, such as in the pair *man bites dog* and *dog bites man*.

Furthermore, sentences all have the exact same sentence structure, and therefore words are aligned. Models that might indirectly benefit from this alignment, such as **Kronecker**, may have been given an advantage due to this, as opposed to models such as **Categorical**, which are designed to compare sentences of different length or syntactic structure, when resolving the difference between intransitive and transitive sentence spaces as has been done in Grefenstette et al. (2011). Future experiments should aim to do away with this automated alignment, and include sentence comparisons that would penalize models which do not leverage syntactic information.

Furthermore, each of these experiments dealt with the comparison of a certain type of sentence (transitive, intransitive). This was convenient for our concrete categorical model, as we defined the sentence space differently based on the valency of the head verb. However, sentences with different sorts of verbs should be able to be directly compared. Not only do several models, both non-syntax sensitive (additive, multiplicative) and syntax-sensitive (Baroni and Zamparelli 2010; Socher et al. 2012), not face this problem, as the product of composition is either naturally in the same space as the vectors being composed or is projected back into it, but the categorical framework the concrete categorical models were derived from does not commit us to different sentence spaces either. The topic of how to solve this problem for the concrete models developed here is beyond the scope of this article, but various options exist, such as the projection of S_T into S_I, the embedding of S_I into S_T, the use of a combined sentence space $S = S_I \oplus S_T$, and so on. It is clear that future experiments should not give this degree of convenience to the models being evaluated (and their designers), by comparing sentences of different types, lengths, and structure so as to more fairly evaluate the capacity of models to produce meaningful semantic representations in a single space, which can be directly compared regardless of syntactic structure and verb-type.

Finally, we mentioned at the beginning of Section 5 that evaluations need to be application-oriented. The class of experiments presented in this article specifically see how well disambiguation occurs as a byproduct of composition. We presented concrete models that would offer ways of combining relations underlying verbs and adjectives, and tested how well these relations disambiguated their arguments and were in turn disambiguated by their arguments. We did not address other aspects of language, such as quantification, logical operations, relative clauses, intensional clauses, embedded sentences, and many other linguistic aspects of spoken and written language that have complex syntactic structure and complicated semantic roles. Addressing these sorts of problems requires determining how negation, conjunction, and other logical relations should be modeled within compositional distributional formalisms, a problem we share with other similar approaches. The development of newer models within the categorical framework we based our work on, and the further development of other approaches mentioned here, must be driven by new experimental goals different from those offered by the task and experiments discussed in this article. Thinking not only about how to address these aspects of language within compositional models, but how to evaluate them, should be a priority for all those interested in further developing this field.

Acknowledgments

We thank the EPSRC for funding the research leading to this article via EPSRC grant EP/J002607/1. Furthermore, we would like to thank John Harding and the equational theorem prover "Prover 9"[11] for checking the identities occurred as a result of translating CCG's rules to the language of a pregroup grammar. We would also like to thank Stephen Clark, Stephen Pulman, Bob Coecke, Karl Moritz Hermann, Richard Socher, and Dimitri Kartsaklis for valuable discussions and comments.

References

Abramsky, S. and B. Coecke. 2004. A categorical semantics of quantum protocols. In *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science*, pages 415–425, Turku, Finland.

Alshawi, H., editor. 1992. *The Core Language Engine*. MIT Press.

Baroni, M. and R. 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 1,183–1,193, Boston, MA.

Béchet, D., A. Foret, and I. Tellier. 2007. Learnability of Pregroup Grammars. *Studia Logica*, 87(2–3), pages 225–252.

Blacoe, W. and M. Lapata. 2012. A comparison of vector-based representations for semantic composition. *Proceedings of the 2012 Conference on Empirical Methods in Natural Language Processing*, Jeju Island.

Buszkowski, W. 2001. Lambek grammars based on pregroups. In P. de Groote, G. Morrill, and C. Retoré, editors, Logical Aspects of Computational Linguistics. *Lecture Notes in Computer Science, volume 2099*. Springer, Berlin Heidelberg, pages 95–109.

Chen, Danqi, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2013. Learning new facts from knowledge bases with neural tensor networks and semantic word vectors. *arXiv preprint arXiv:1301.3618*.

Clark, S. 2013. Type-driven syntax and semantics for composing meaning vectors. In *Quantum Physics and Linguistics: A Compositional, Diagrammatic Discourse.* Oxford University Press.

Clark, S., B. Coecke, and M. Sadrzadeh. 2008. A compositional distributional model of meaning. In *Proceedings of the Second Quantum Interaction Symposium (QI-2008)*, Oxford.

Clark, S. and J. R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33:493–552.

Clark, S. and S. Pulman. 2007. Combining symbolic and distributional models of meaning. In *AAAI Spring Symposium on Quantum Interaction*, Stanford, USA.

Clarke, Daoud. 2009. Context-theoretic semantics for natural language: An overview. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*, pages 112–119, Edinburgh.

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

Coecke, B. and É. O. Paquette. 2011. Categories for the practising physicist. In B. Coecke, editor, *New Structures for Physics. Lecture Notes in Physics*, volume 813, pages 173–286, Springer, Berlin Heidelberg.

Coecke, B., M. Sadrzadeh, and S. Clark. 2010. Mathematical Foundations for a Compositional Distributional Model of Meaning. *Linguistic Analysis*, 36:345–384.

Curran, James, Stephen Clark, and Johan Bos. 2007. Linguistically motivated large-scale NLP with C&C and boxer. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 33–36, Prague.

Curran, J. R. 2004. *From distributional to semantic similarity*. Ph.D. thesis, School of Informatics, University of Edinburgh.

Dean, Jeffrey and Sanjay Ghemawat. 2008. MapReduce: Simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113.

Erk, Katrin and Sebastian Padó. 2008. A structured vector space model for word meaning in context. *Proceedings of the Conference on Empirical Methods in Natural Language Processing - EMNLP '08*, pages 897–906, Edinburgh.

Finkelstein, L., E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin. 2001. Placing search in

[11] http://www.cs.unm.edu/∼mccune/mace4/.

context: The concept revisited. In *Proceedings of the 10th International Conference on World Wide Web*, pages 406–414, Hong Kong.

Firth, J. R. 1957. A synopsis of linguistic theory 1930-1955. *Studies in linguistic analysis*.

Fowler, T. A. D. and G. Penn. 2010. Accurate context-free parsing with combinatory categorial grammar. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 335–344, Uppsala.

Frege, G. 1892. Uber sinn und bedeutung. *Zeitschrift für Philosophie und philosophische Kritik*, 100(1):25–50.

Genkin, Daniel, Nissim Francez, and Michael Kaminski. 2010. Mildly context-sensitive languages via buffer augmented pregroup grammars. In Z. Manna and D. A. Peled, editors, *Time for Verification*, pages 144–166, Springer-Verlag, Berlin, Heidelberg.

Girard, Jean-Yves. 1987. Linear logic. *Theoretical Computer Science*, 50:1–102.

Grefenstette, E. 2009. Analysing document similarity measures. Master's thesis, University of Oxford.

Grefenstette, E., G. Dinu, Y. Zhang, M. Sadrzadeh, and M. Baroni. 2013. Multi-step regression learning for compositional distributional semantics. In *Proceedings of the Tenth International Conference on Computational Semantics*, Potsdam.

Grefenstette, E. and M. 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 1,394–1,404, Edinburgh.

Grefenstette, E. and M. Sadrzadeh. 2011b. Experimenting with transitive verbs in a DisCoCat. In *Proceedings of the 2011 EMNLP Workshop on Geometric Models of Natural Language Semantics*, pages 62–66, Edinburgh.

Grefenstette, E., M. Sadrzadeh, S. Clark, B. Coecke, and S. Pulman. 2011. Concrete sentence spaces for compositional distributional models of meaning. In *Proceedings of the Ninth International Conference on Computational Semantics*, pages 125–134, Oxford.

Grefenstette, G. 1992. Use of syntactic context to produce term association lists for text retrieval. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 89–97, Copenhagen.

Grefenstette, G. 1994. *Explorations in automatic thesaurus discovery*.

Guevara, E. 2010. A regression model of adjective-noun compositionality in distributional semantics. In *Proceedings of the 2010 Workshop on Geometrical Models of Natural Language Semantics*, pages 33–37, Uppsala.

Harris, Z. S. 1968. *Mathematical structures of language*. Wiley.

Hockenmaier, Julia. 2003. *Data and Models for Statistical Parsing with Combinatory Categorial Grammar*. Ph.D. thesis, School of Informatics, University of Edinburgh.

Hockenmaier, Julia and Mark Steedman. 2007. CCGBank: A corpus of CCG derivations and dependency structures extracted from the Penn treebank. *Computational Linguistics*, 33(3):355–396.

Joshi, A. K., K. Vijay-Shanker, and D. J. Weir. 1989. The convergence of mildly context-sensitive grammar formalisms. Working paper, University of Pennsylvania, School of Engineering and Applied Science, Dept. of Computer and Information Science.

Lambek, J. 1958. The mathematics of sentence structure. *The American Mathematical Monthly*, 65(3):154–170.

Lambek, J. 1999. Type grammar revisited. *Logical Aspects of Computational Linguistics*, pages 1–27.

Lambek, J. 2008. *From word to sentence. A computational algebraic approach to grammar*. Milan, Polimetrica.

Lambek, J., 2010. *Compact Monoidal Categories from Linguistics to Physics*, pages 451–469.

Landauer, T. K. 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*.

Mac Lane, S. 1998. *Categories for the Working Mathematician*. Springer Verlag.

Manning, C. D., P. Raghavan, and H. Schütze. 2011. Introduction to information retrieval. Cambridge University Press, New York, NY.

Minnen, G., J. Carroll, and D. Pearce. 2001. Applied morphological processing of English. *Natural Language Engineering*, 7(03):207–223.

Mitchell, J. and M. Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of ACL*, volume 8, pages 236–244, Columbus, OH.

Mitchell, J. and M. Lapata. 2010. Composition in Distributional Models of Semantics. *Cognitive Science* 34(8):1388–1429.

Montague, R. 1974. English as a Formal Language. In R. H. Thomason, editor, *Formal Semantics: The Essential Readings*.

Moortgat, M. 1997. Categorial type logics. In H. van Ditmarsch and L. S. Moss, editors, *Handbook of Logic and Language*. Elsevier, pages 93–177.

Padó, Sebastian and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.

Pham, N., R. Bernardi, Y.-Z. Zhang, and M. Baroni. 2013. Sentence paraphrase detection: When determiners and word order make the difference. In *Proceedings of the Towards a Formal Distributional Semantics Workshop at IWCS 2013*, pages 21–29, Potsdam.

Plate, T. A. 1991. Holographic reduced representations: Convolution algebra for compositional distributed representations. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 30–35, Hyderabad.

Preller, A. 2010. Polynomial pregroup grammars parse context sensitive languages. *Linguistic Analysis*, 36:483–516.

Preller, A. and M. Sadrzadeh. 2010. Bell states and negative sentences in the distributed model of meaning. In P. Selinger, B. Coecke, P. Panangaden, editors, *Electronic Notes in Theoretical Computer Science, Proceedings of the 6th QPL Workshop on Quantum Physics and Logic*. University of Oxford.

Selinger, P. 2010. A survey of graphical languages for monoidal categories. *New Structures for Physics*, pages 275–337.

Smolensky, P. 1990. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, 46(1-2):159–216.

Smolensky, P. and G. Legendre. 2006. *The Harmonic Mind: From Neural Computation to Optimality-Theoretic Grammar Volume I: Cognitive Architecture*. MIT Press.

Socher, R., B. Huval, C. D. Manning, and A. Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Conference on Empirical Methods in Natural Language Processing*, pages 1,201–1,211, Jeju Island.

Steedman, M. 2001. *The Syntactic Process*. The MIT Press.

Steedman, M. and J. Baldridge. 2011. *Combinatory Categorial Grammar*. Wiley-Blackwell.

Stolcke, A. 2002. SRILM—An extensible language modeling toolkit. In *Seventh International Conference on Spoken Language Processing*.

Turney, P. D. and P. Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37(1):141–188.

Turney, Peter D. 2012. Domain and function: A dual-space model of semantic relations and compositions. *Journal of Artificial Intelligence Research*, 44:533–585.

Van Rijsbergen, C. J. 2004. *The Geometry of Information Retrieval*. Cambridge University Press.

Walters, R. F. 1991. *Categories and Computer Science*. Cambridge University Press.

Widdows, D. 2005. *Geometry and Meaning*. University of Chicago Press.

Widdows, D. 2008. Semantic vector products: Some initial investigations. In *Proceedings of the Second Quantum Interaction Symposium (QI-2008). College Publications*. CITESEER, Oxford.

Wittgenstein, L. 1953. *Philosophical Investigations*. Blackwell.

Zanzotto, F. M., I. Korkontzelos, F. Fallucchi, and S. Manandhar. 2010. Estimating linear models for compositional distributional semantics. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1,263–1,271, Beijing.