Springer Science Reviews (2014) 2:161–177
DOI 10.1007/s40362-014-0017-z

STUDENT REVIEW

Compositional Operators in Distributional Semantics

Dimitri Kartsaklis

Received: 22 May 2013 / Revised: 18 January 2014 / Accepted: 4 March 2014 / Published online: 12 April 2014

Abstract This survey presents in some detail the main advances that have been recently taking place in Computational Linguistics towards the unification of the two prominent semantic paradigms: the compositional formal semantics view and the distributional models of meaning based on vector spaces. After an introduction to these two approaches, I review the most important models that aim to provide compositionality in distributional semantics. Then I proceed and present in more detail a particular framework [7] based on the abstract mathematical setting of category theory, as a more complete example capable to demonstrate the diversity of techniques and scientific disciplines that this kind of research can draw from. This paper concludes with a discussion about important open issues that need to be addressed by the researchers in the future.

Keywords Natural language processing · Distributional semantics · Compositionality · Vector space models · Formal semantics · Category theory · Compact closed categories

Introduction

The recent developments on the syntactical and morphological analysis of natural language text constitute the first step towards a more ambitious goal of assigning a proper form of *meaning* to arbitrary text compounds. Indeed, for certain really 'intelligent' applications, such as machine translation, question-answering systems, paraphrase detection or automatic essay scoring, to name just a few, there will always exist a gap between raw linguistic information (such as part-of-speech labels, for example) and the knowledge of the real world that is needed for the completion of the task in a satisfactory way. Semantic analysis has exactly this role, aiming to close (or reduce as much as possible) this gap by linking the linguistic information with semantic representations that embody this elusive real-world knowledge.

The traditional way of adding semantics to sentences is a syntax-driven compositional approach: every word in the sentence is associated with a primitive symbol or a predicate, and these are combined to larger and larger logical forms based on the syntactical rules of the grammar. At the end of the syntactical analysis, the logical representation of the whole sentence is a complex formula that can be fed to a theorem prover for further processing. Although such an approach seems intuitive, it has been shown that it is rather inefficient for any practical application (for example, Bos and Markert [5] get very low recall scores for a textual entailment task). Even more importantly, the meaning of the atomic units (words) is captured in an axiomatic way, namely by ad-hoc unexplained primitives that have nothing to say about the real semantic value of the specific words.

On the other hand, distributional models of meaning work by building co-occurrence vectors for every word in a corpus based on its context, following Firth's intuition that 'you should know a word by the company it keeps' [12]. These models have been proved useful in many natural language tasks (see section From Words to Sentence) and can provide concrete information for the words of a sentence, but they do not scale up to larger constituents of text, such as phrases or sentences. Given the complementary nature of these two distinct approaches, it is not a surprise that compositional abilities of distributional models have

D. Kartsaklis (✉)
Department of Computer Science, University of Oxford, Oxford, UK
e-mail: dimitri.kartsaklis@cs.ox.ac.uk

been the subject of much discussion and research in recent years. Towards this purpose researchers exploit a wide variety of techniques, ranging from simple mathematical operations like addition and multiplication to neural networks and even category theory. The purpose of this paper is to provide a concise survey of the developments that have been taking place towards the goal of equipping distributional models of meaning with compositional abilities.

The plan is the following: In section Compositional Semantics and Distributional Semantics I provide an introduction to compositional and distributional models of meaning, respectively, explaining the basic principles and assumptions on which they rely. Then I proceed to review the most important methods aiming towards their unification (section Compositionality in Distributional Approaches). As a more complete example of such a method (and as a demonstration of the multidisciplinarity of Computational Linguistics), section A Categorical Framework for Natural Language describes the framework of Coecke et al. [7], based on the abstract setting of category theory. Section Verb and Sentence Spaces provides a closer look to the form of a sentence space, and how our sentence-producing functions (i.e. the verbs) can be built from a large corpus. Finally, section Challenges and Open Questions discusses important philosophical and practical open questions and issues that form part of the current and future research.

Compositional Semantics

Compositionality in semantics offers an elegant way to address the inherent property of natural language to produce infinite structures (phrases and sentences) from finite resources (words). The *principle of compositionality* states that the meaning of a complex expression can be determined by the meanings of its constituents and the rules used for combining them. This idea is quite old, and glimpses of it can be spotted even in works of Plato. In his dialogue *Sophist*, Plato argues that a sentence consists of a noun and a verb, and that the sentence is true if the verb denotes the action that the noun is currently performing. In other words, Plato argues that (a) a sentence has a structure, (b) the parts of the sentence have different functions and (c) the meaning of the sentence is determined by the function of its parts. Nowadays, this intuitive idea is often attributed to Gottlob Frege, who expresses similar views in his 'Foundations of Mathematics', originally published in 1884. In an undated letter to Philip Jourdain, included in 'Philosophical and Mathematical Correspondence' [14], Frege provides an explanation for the reason this idea seems so intuitive:

> The possibility of our understanding propositions which we have never heard before rests evidently on this, that we can construct the sense of a proposition out of parts that correspond to words.

This forms the basis of the *productivity* argument, often used as a proof for the validity of the principle: humans only know the meaning of words, and the rules to combine them in larger constructs; yet, being equipped with this knowledge, we are able to produce new sentences that we have never uttered or heard before. Indeed, this task seems natural even for a 3-year-old child—however, its formalization in a way reproducible by a computer has been proven nothing but trivial. The modern compositional models owe a lot to the seminal work of Richard Montague (1930–1971), who has managed to present a systematic way of processing fragments of the English language in order to get semantic representations capturing their 'meaning' [30–32]. In his 'Universal Grammar' (1970b), Montague states that

> There is in my opinion no important theoretical difference between natural languages and the artificial languages of logicians.

Montague supports this claim by detailing a systematization of the natural language, an approach which became known as Montague grammar. To use Montague's method, one would need two things: first, a resource which will provide the logical forms of each specific word (a lexicon); and second, a way to determine the correct order in which the elements in the sentence should be combined in order to end up with a valid semantic representation. A natural way to address the latter, and one traditionally used in computational linguistics, is to use the syntactic structure as a means of driving the semantic derivation (an approach called *syntax-driven semantic analysis*). In other words, we assume that there exists a mapping from syntactic to semantic types, and that the composition in the syntax level implies a similar composition in the semantic level. This is known as the *rule-to-rule hypothesis* [1].

In order to provide an example, I will use the sentence 'Every man walks'. We begin from the lexicon, the job of which is to provide a grammar type and a logical form to every word in the sentence:

$$\begin{aligned} (1)\quad &\text{a.}\quad \text{every} \vdash Det : \lambda P.\lambda Q.\forall x[P(x) \rightarrow Q(x)] \\ &\text{b.}\quad \text{man} \vdash N : \lambda y.man(y) \\ &\text{c.}\quad \text{walks} \vdash Verb_{IN} : \lambda z.walks(z). \end{aligned}$$

The above use of formal logic (especially higher order) in conjunction with λ-calculus was first introduced by Montague, and from then on it constitutes the standard way of providing logical forms to compositional models. In the above lexicon, predicates of the form $man(y)$ and $walks(z)$

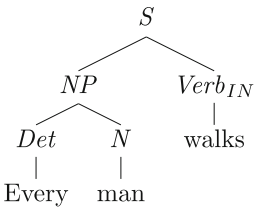
are true if the individuals denoted by y and z carry the property (or, respectively, perform the action) indicated by the predicate. From an extensional perspective, the semantic value of a predicate can be seen as the set of all individuals that carry a specific property: *walks*(*john*) will be true if the individual *john* belongs to the set of all individuals who perform the action of walking. Furthermore, λ-terms like λx or λQ have the role of placeholders that remain to be filled. The logical form $\lambda y.man(y)$, for example, reflects the fact that the entity which is going to be tested for the property of manhood is still unknown and it will be later specified based on the syntactic combinatorics. Finally, the form in (1a) reflects the traditional way for representing a universal quantifier in natural language, where the still unknown part is actually the predicates acting over a range of entities.

In λ-calculus, function application is achieved via the process of β-reduction: given two logical forms $\lambda x.t$ and s, the application of the former to the latter will produce a version of t, where all the free occurrences of x in t have been replaced by s. More formally,

$$(\lambda x.t)s \rightarrow_\beta t[x := s]\,. \tag{1}$$

Let us see how we can apply the principle of compositionality to get a logical form for the above example sentence, by repeatedly applying β-reduction between the semantic forms of text constituents following the grammar rules. The parse tree in (2) below provides us a syntactic analysis:

(2)



Our simple context-free grammar (CFG) consists of two rules:

(3) $NP \rightarrow Det\ N$ a noun phrase consists of a determiner and a noun.
$S \rightarrow NP\ Verb_{IN}$ a sentence consists of a noun phrase and an intransitive verb.

These rules will essentially drive the semantic derivation. Interpreted from a semantic perspective, the first rule states that the logical form of a noun phrase is derived by applying the logical form of a determiner to the logical form of a noun. In other words, P in (1a) will be substituted by the logical form for *man* (1b). The details of this reduction are presented below:

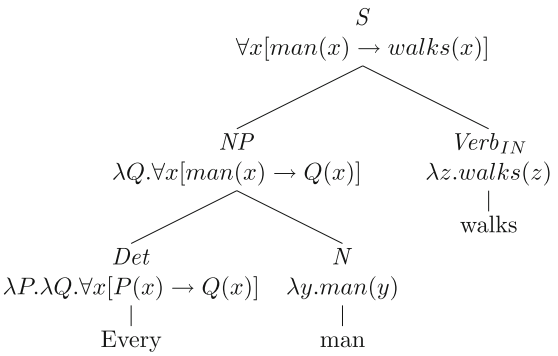
$$\begin{aligned}(4)\quad & \lambda P.\lambda Q.\forall x[P(x)\rightarrow Q(x)](\lambda y.man(y)) \\ & \rightarrow_\beta \lambda Q.\forall x[(\lambda y.man(y))(x)\rightarrow Q(x)] \quad P:=\lambda y.man(y) \\ & \rightarrow_\beta \lambda Q.\forall x[man(x)\rightarrow Q(x)] \quad y:=x\end{aligned}\,.$$

Similarly, the second rule signifies that the logical form of the whole sentence is derived by the combination of the logical form of the noun phrase as computed in (4) above with the logical form of the intransitive verb (1c):

$$\begin{aligned}(5)\quad & \lambda Q.\forall x[man(x)\rightarrow Q(x)](\lambda z.walks(z)) \\ & \rightarrow_\beta \forall x[man(x)\rightarrow(\lambda z.walks(z))(x)] \quad Q:=\lambda z.walks(z) \\ & \rightarrow_\beta \forall x[man(x)\rightarrow walks(x)] \quad z:=x\end{aligned}\,.$$

Thus we have arrived at a logical form which can be seen as a semantic representation of the whole sentence. The tree below provides a concise picture of the complete semantic derivation:

(6)



A logical form such as $\forall x[man(x) \rightarrow walks(x)]$ simply states the truth (or falseness) of the expression given the sub-expressions and the rules for composing them. It does not provide any quantitative interpretation of the result (e.g. grades of truth) and, even more importantly, leaves the meaning of words as unexplained primitives (*man*, *walks*, etc.). In the next section we will see how distributional semantics can fill this gap.

Distributional Semantics

The Distributional Hypothesis

The distributional paradigm is based on the *distributional hypothesis* [18], stating that words that occur in the same context have similar meanings. Various forms of this popular idea keep recurring in the literature: Firth [12] calls it collocation, while Frege himself states that 'never ask for the meaning of a word in isolation, but only in the context of a proposition'. [13]. The attraction of this principle in the context of Computational Linguistics is that it provides a way of concretely representing the meaning of a word via mathematics: each word is a vector whose elements show how many times this word occurred in some corpus at the same context with every other word in the vocabulary. If, for example, our basis is $\{cute, sleep, finance, milk\}$, the vector for word 'cat' could have the form $(15, 7, 0, 22)$ meaning that 'cat' appeared 15 times together with 'cute',

7 times with 'sleep' and so on. More formally, given an orthonormal basis $\{\overrightarrow{n_i}\}_i$ for our vector space, a word is represented as:

$$\overrightarrow{word} = \sum_i c_i \overrightarrow{n_i} \tag{2}$$

where c_i is the coefficient for the ith basis vector. As mentioned above, in their simplest form these coefficients can be just co-occurrence counts, although in practice a function on counts is often used in order to remove some of the unavoidable frequency bias. A well-known measure is the information-theoretic *point-wise mutual information* (PMI), which can reflect the relationship between a context word c and a target word t as follows:

$$\mathrm{PMI}(c,t) = \log \frac{p(c|t)}{p(t)}. \tag{3}$$

On the contrary with compositional semantics which leaves the meaning of lexical items unexplained, a vector like the one used above for 'cat' provides some concrete information about the meaning of the specific word: cats are cute, sleep a lot, they really like milk, and they have nothing to do with finance. Additionally, this quantitative representation allows us to compare the meanings of two words, e.g. by computing the cosine distance of their vectors, and evaluate their semantic similarity. The cosine distance is a popular choice for this task (but not the only one) and is given by the following formula:

$$\mathrm{sim}(\overrightarrow{v}, \overrightarrow{u}) = \cos(\overrightarrow{v}, \overrightarrow{u}) = \frac{\langle \overrightarrow{v} | \overrightarrow{u} \rangle}{\| \overrightarrow{v} \| \| \overrightarrow{u} \|}, \tag{4}$$

where $\langle \overrightarrow{v} | \overrightarrow{u} \rangle$ denotes the dot product between $\overrightarrow{v}$ *and* $\overrightarrow{u}$, and $\| \overrightarrow{v} \|$ the magnitude of $\overrightarrow{v}$.

As an example, in the 2-dimensional vector space of Fig. 1 we see that 'cat' and 'puppy' are close together (and both of them closer to the basis 'cute'), while 'bank' is closer to basis vector 'finance'.

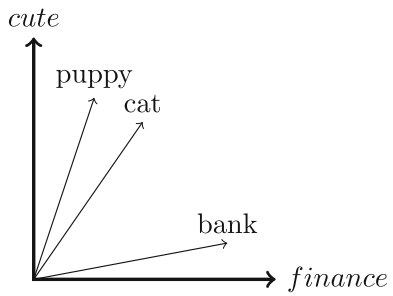


Fig. 1 A toy 'vector space' for demonstrating semantic similarity between words

A more realistic example is shown in Fig. 2. The points in this space represent real distributional word vectors created from British National Corpus (BNC),[1] originally 2,000-dimensional and projected onto two dimensions for visualization. Note how words form distinct groups of points according to their semantic correlation. Furthermore, it is interesting to see how ambiguous words behave in these models: the ambiguous word 'mouse' (with the two meanings to be that of a rodent and of a computer pointing device), for example, is placed almost equidistantly from the group related to IT concepts (lower left part of the diagram) and the animal group (top left part of the diagram), having a meaning that can be indeed seen as the average of both senses.

Forms of Word Spaces

In the simplest form of a word space (a vector space for words), the context of a word is a set containing all the words that occur within a certain distance from the target word, for example within a 5-word window. Models like these have been extensively studied and implemented in the past years, see for example the work of Lowe [26] and Lowe and McDonald [27]. However, although such an approach is simple, intuitive, and computationally efficient, it is not optimal, since it assumes that every word within this window will be semantically relevant to the target word, while it treats every word outside of the window as irrelevant. Unfortunately, this is not always the case. Consider for example the following phrase:

(7) The movie I saw and John said he really likes.

Here, a word-based model cannot efficiently capture the long-ranged dependency between 'movie' and 'likes'; on the other hand, since 'said' is closer to 'movie' it is more likely to be considered as part of its context, despite the fact that the semantic relationship between the two words is actually weak. Cases like the above suggest that a better approach for the construction of the model would be to take into account not just the surface form of context words, but also the specific grammatical relations that hold between them and the target word. A model like this, for example, will be aware that 'movie' is the object of 'likes', so it could safely consider the latter as part of the context for the former and vice versa. This kind of observations motivated many researchers to experiment with vector spaces based not solely on the surface forms of words but also on various morphological and syntactical properties of the text.

[1] BNC is a 100 million-word text corpus consisting of samples of written and spoken English. It can be found online at http://www.natcorp.ox.ac.uk/.

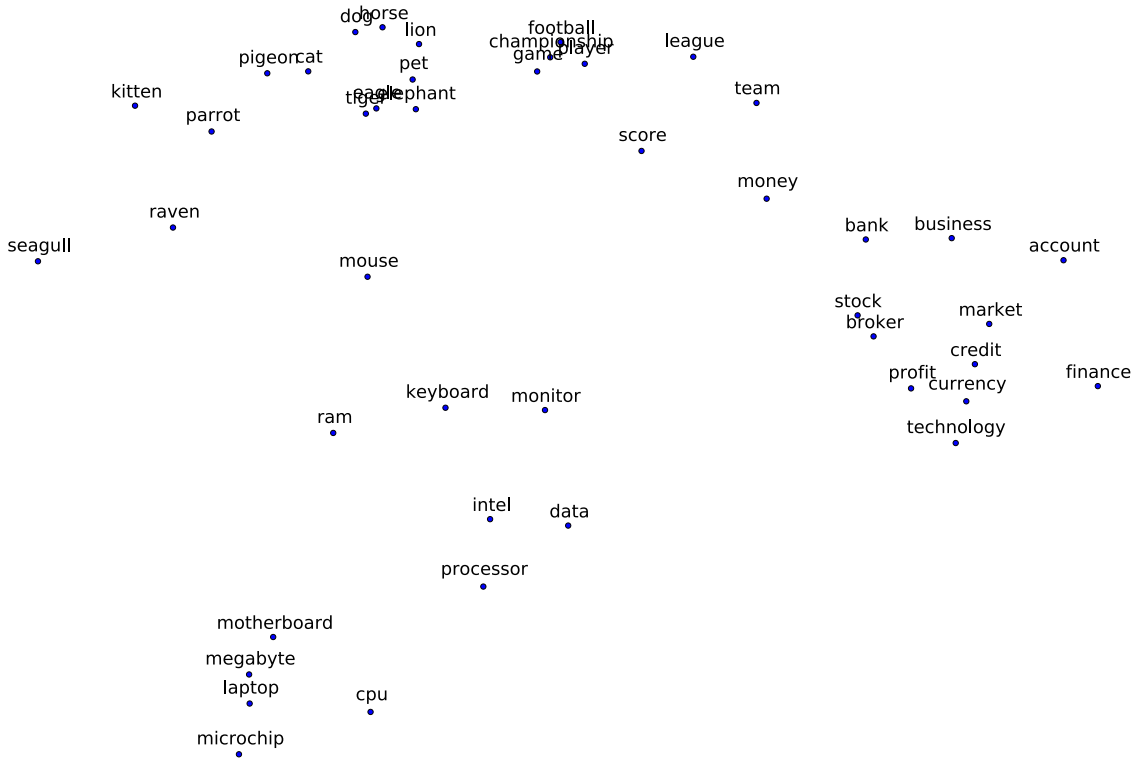


Fig. 2 Visualization of a word space in two dimensions (original vectors are 2000-dimensional vectors created from BNC)

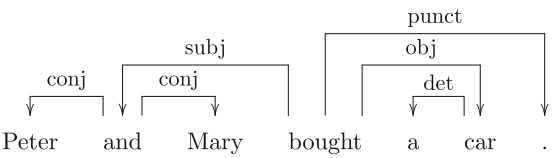
One of the earliest attempts to add syntactical information in a word space was that of Grefenstette [17], who used a structured vector space with a basis constructed by grammatical properties such as 'subject-of-buy' or 'argument-of-useful', denoting that the target word has occurred in the corpus as subject of the verb 'buy' or as argument of the adjective 'useful'. The weights of a word vector were binary, either 1 for at least one occurrence or 0 otherwise. Lin [25] moves one step further, replacing the binary weights with frequency counts. Following a different path, Erk and Padó [10] argue that a single vector is not enough to capture the meaning of a word; instead, the vector of a word is accompanied by a set of vectors R representing the lexical preferences of the word for its arguments positions, and a set of vectors R^{-1}, denoting the inverse relationship; that is, the usage of the word as argument in the lexical preferences of other words. Subsequent works by Thater et al. [45, 46] present a version of this idea extended with the inclusion of grammatical dependency contexts.

In an attempt to provide a generic distributional framework, Padó and Lapata [34] presented a model based on dependency relations between words. The interesting part of this work is that, given the proper parametrization, it is indeed able to essentially subsume a large amount of other works on the same subject. For this reason, it is worth of a more detailed description which I provide in the next section.

A Dependency-Based Model

The generic framework of Padó and Lapata [34] treats each sentence as a directed graph, the nodes of which are the words and the edges the dependency relations that hold between them. In (8) below we can see a typical dependency diagram; each arrow starts from a head word and ends on a dependent, whereas the labels denote the specific relationships that hold between the words:

(8)



The context of a target word consists of all dependency paths that start from this specific word and end to some other word within a given distance. A valid path should not be cyclic, in order to avoid linguistically meaningless situations such as paths of infinite length or paths consisting of unconnected fragments. Let us see an example for the simple sentence 'dogs chase cats', whose dependency relations set is $\{object(chase, cats), subject(chase, dogs)\}$. The set of paths for the target word 'dogs', then, would be $\{dogs—chase, dogs—chase—cats\}$ (note that the direction information is dropped at this point—the paths are treated as *undirected* in order to catch relationships

between e.g. a subject and an object, which otherwise would be impossible). The creation of a vector space involves the following steps:

1. For each target word t, collect all the undirected paths that start from this word. This will be the initial context for t, denoted by Π_t.
2. Apply a context selection function $cont$: $W \rightarrow 2^{\Pi_t}$ (where W is the set of all tokens of type t), which assigns to t a subset of the initial context. Given a word-window k, for example, this function might be based on the absolute difference between the position of t and the position of the end word for each path.
3. For every path $\pi \in cont(t)$, specify a relative importance value by applying a path value function of the form v: $\Pi \rightarrow \mathbb{R}$.
4. For every $\pi \in cont(t)$, apply a basis-mapping function μ: $\Pi \rightarrow B$, which maps each path π to a specific basis element.
5. Calculate the co-occurrence frequency of t with a basis element b by a function f: $B \times T \rightarrow \mathbb{R}$, defined as:
$$f(b,t) = \sum_{w \in W(t)} \sum_{\pi \in cont(w) \wedge \mu(\pi) = b} v(\pi). \quad (5)$$
6. Finally, and in order to remove potential frequency bias due to raw counts, calculate the log-likelihood ratio G^2 [9] for all basis elements.

An appealing future of this framework is that it is fully parametrized by using different forms of the functions $cont$, v and μ. In order to avoid sparsity problems, the authors chose to use for their experiments a fixed basis-mapping function that maps every dependency path to its ending word. This results in a vector space with words as basis elements, quite similar to the word-based setting described in section Forms of Word Spaces. There is, however, the important difference that the inclusion criterion of a word into the context of another word is not simply the co-occurrence of the two words within a given window, but the fact that they are related through some dependency path. The authors experimented with various similarity measures and different forms of the functions $cont$ and v, producing similarity scores for the well-known benchmark dataset of Rubenstein and Goodenough [38]. This dataset consists of 65 pairs of nouns; the task is to estimate the similarity between the two nouns of each pair and compare the results with a gold reference prepared by human annotators. Their best results (expressed as the Pearson correlation of model scores with human scores) came from the information-theoretic similarity measure of Lin [25], a context selection function that retains paths with length ≤ 3, and a simple path value function that assigns 1 to paths of length 1 (since these paths correspond to the most direct and strong dependencies), and fractions to longer paths. More formally, the best-performing model has the following form (including the selected basis mapping function):

$$cont(t) = \{\pi \in \Pi_t | \|\pi\| \leq 3\}, v(\pi) = \frac{1}{\|\pi\|}, \mu(\pi) = end(\pi) \quad (6)$$

where $end(\pi)$ denotes the ending word of path π.

From Words to Sentence

Distributional models of meaning have been widely studied and successfully applied on a variety of language tasks, especially during the last decade with the availability of large-scale corpora, like Gigaword [15] and ukWaC [11], which provide a reliable resource for training the vector spaces. For example, Landauer and Dumais [24] use vector space models in order to model and reason about human learning rates in language; Schütze [40] performs word sense induction and disambiguation; Curran [8] shows how distributional models can be applied to automatic thesaurus extraction and Manning et al. [28] discuss possible applications in the context of information retrieval.

However, due to the infinite capacity of a natural language to produce new sentences from finite resources, no corpus, regardless its size, can be used for providing vector representations to anything but very small text fragments, usually only to words. Under this light, the provision of distributional models with compositional abilities similar to what was described in section Compositional Semantics seems a very appealing solution that could offer the best of both worlds in a unified manner. The goal of such a system would be to use the compositional rules of a grammar, as described in section Compositional Semantics, in order to combine the context vectors of the words to vectors of larger and larger text constituents, up to the level of a sentence. A sentence vector, then, could be compared with other sentence vectors, providing a way for assessing the semantic similarity between sentences as if they were words. The benefits of such a feature are obvious for many natural language processing tasks, such as paraphrase detection, machine translation, information retrieval, and so on, and in the following section I am going to review all the important approaches and current research towards this challenging goal.

Compositionality in Distributional Approaches

Vector Mixture Models

The transition from word vectors to sentence vectors implies the existence of a composition operation that can be applied between text constituents according to the rules

of the grammar: the composition of 'red' and 'car' into the adjective–noun compound 'red car', for example, should produce a new vector derived from the composition of the context vectors for 'red' and 'car'. Since we work with vector spaces, the candidates that first come to mind is vector addition and vector (point-wise) multiplication. Indeed, Mitchell and Lapata [29] present and test various models, where the composition of vectors is based on these two simple operations. Given two word vectors $\overrightarrow{w_1}$ and $\overrightarrow{w_2}$ and assuming an orthonormal basis $\{\overrightarrow{n_i}\}_i$, the multiplicative model computes the meaning vector of the new compound as follows:

$$\overrightarrow{w_1 w_2} = \overrightarrow{w_1} \odot \overrightarrow{w_2} = \sum_i c_i^{w_1} c_i^{w_2} \overrightarrow{n_i}. \tag{7}$$

Similarly, the meaning according to the additive model is:

$$\overrightarrow{w_1 w_2} = \alpha \overrightarrow{w_1} + \beta \overrightarrow{w_2} = \sum_i (\alpha c_i^{w_1} + \beta c_i^{w_2}) \overrightarrow{n_i}, \tag{8}$$

where α and β are optional weights denoting the relative importance of each word. The main characteristic of these models is that all word vectors live into the same space, which means that in general there is no way to distinguish between the type-logical identities of the different words. This fact, in conjunction with the element-wise nature of the operators, makes the output vector a kind of *mixture* of the input vectors. Figure 3 demonstrates this; each element of the output vector can be seen as an 'average' of the two corresponding elements in the input vectors. In the additive case, the components of the result are simply the cumulative scores of the input components. So in a sense the output element embraces both input elements, resembling a union of the input features. On the other hand, the multiplicative version is closer to intersection: a zero element in one of the input vector will eliminate the corresponding feature in the output, no matter how high the other component was.

Vector mixture models constitute the simplest compositional method in distributional semantics. Despite their simplicity, though, (or because of it) these approaches have been proved very popular and useful in many NLP tasks, and they are considered hard-to-beat baselines for many of the more sophisticated models we are going to discuss next.

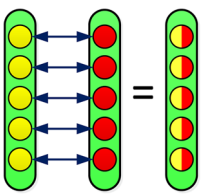


Fig. 3 Vector mixture models. Each element in the output vector is a mixture of the corresponding elements in the input vectors

In fact, the comparative study of Blacoe and Lapata [4] suggests something really surprising: that, for certain tasks, additive and multiplicative models can be almost as much effective as state-of-the-art deep learning models, which will be the subject of section Deep Learning Models.

Tensor Product and Circular Convolution

The low complexity of vector mixtures comes with a price, since the produced composite representations disregard grammar in many different ways. For example, an obvious problem with these approaches is the commutativity of the operators: the models treat a sentence as a 'bag of words' where the word order does not matter, equating for example the meaning of sentence 'dog bites man' with that of 'man bites dog'. This fact motivated researchers to seek solutions on non-commutative operations, such as the tensor product between vector spaces. Following this suggestion, which was originated by Smolensky [41], the composition of two words is achieved by a structural mixing of the basis vectors, which results in an increase in dimensionality:

$$\overrightarrow{w_1} \otimes \overrightarrow{w_2} = \sum_{i,j} c_i^{w_1} c_j^{w_2} (\overrightarrow{n_i} \otimes \overrightarrow{n_j}). \tag{9}$$

Clark and Pulman [6] take this original idea further and propose a concrete model in which the meaning of a word is represented as the tensor product of the word's context vector with another vector that denotes the grammatical role of the word and comes from a different abstract vector space of grammatical relationships. As an example, the meaning of the sentence 'dog bites man' is given as

$$\overrightarrow{dog\ bites\ man} = (\overrightarrow{dog} \otimes \overrightarrow{subj}) \otimes \overrightarrow{bites} \otimes (\overrightarrow{man} \otimes \overrightarrow{obj}). \tag{10}$$

Although tensor product models solve the bag-of-words problem, unfortunately they introduce a new very important issue: given that the cardinality of the vector space is d, the space complexity grows exponentially as more constituents are composed together. With $d = 300$, and assuming a typical floating-point machine representation (8 bytes per number), the vector of Eq. 10 would require $300^5 \times 8 = 1.944 \times 10^{13}$ bytes (≈ 19.5 terabytes). Even more importantly, the use of tensor product as above only allows the comparison of sentences that share the same structure, i.e. there is no way for example to compare a transitive sentence with an intransitive one, a fact that severely limits the applicability of such models.

Using a concept from signal processing, Plate [35] suggests the replacement of tensor product by circular convolution. This operation carries the appealing property that its application on two vectors results in a vector of the

same dimensions as the operands. Let $\overrightarrow{v}$ and $\overrightarrow{u}$ be vectors of d elements, the circular convolution of them will result in a vector $\overrightarrow{c}$ of the following form:

$$\overrightarrow{c} = \overrightarrow{v} \circledast \overrightarrow{u} = \sum_{i=0}^{d-1} \left(\sum_{j=0}^{d-1} v_j u_{i-j} \right) \overrightarrow{n}_{i+1}, \tag{11}$$

where $\overrightarrow{n_i}$ represents a basis vector, and the subscripts are modulo-d, giving to the operation its circular nature. This can be seen as a compressed outer product of the two vectors. However, a successful application of this technique poses some restrictions. For example, it requires a different interpretation of the underlying vector space, in which 'micro-features', such as PMI weights for each context word, are replaced by what Plate calls 'macro-features', i.e. features that are represented by whole vectors drawn from a normal distribution. Furthermore, circular convolution is commutative, re-introducing the bag-of-words problem.[2]

Tensor-Based Models

A weakness of vector mixture models and the generic tensor product approach is the symmetric way in which they treat all words, ignoring their special roles in the sentence. An adjective, for example, lives in the same space with the noun it modifies, and both will contribute equally to the output vector representing the adjective–noun compound. However, a treatment like this seems unintuitive; we tend to see relational words, such as verbs or adjectives, as functions acting on a number of arguments rather than entities of the same order as them. Following this idea, a recent line of research represents words with special meaning as linear maps (tensors[3] of higher order) that apply on one or more arguments (vectors or tensors of lower order). An adjective, for example, is not any more a simple vector but a matrix (a tensor of order 2) that, when matrix-multiplied with the vector of a noun, will return a modified version of it (Fig. 4).

This approach is based on the well-known *Choi–Jamiołkowsky isomorphism*: every linear map from V to W (where V and W are finite-dimensional Hilbert spaces) stands in one-to-one correspondence with a tensor living in the tensor product space $V \otimes W$. For the case of a multilinear map (a function with more than one argument), this can be generalized to the following:

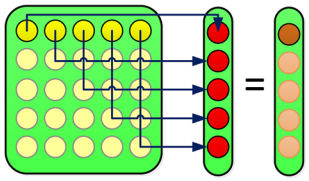


Fig. 4 Tensor-based models. The ith element in the output vector is computed as the linear combination of the input vector with the ith row of the matrix representing the linear map

$$f : V_1 \to \ldots \to V_j \to V_k \cong V_1 \otimes \ldots \otimes V_j \otimes V_k \,. \tag{12}$$

In general, the order of the tensor is equal to the number of arguments plus one order for carrying the result; so a unary function (such as an adjective) is a tensor of order 2, while a binary function (e.g. a transitive verb) is a tensor of order 3. The composition operation is based on the inner product and is nothing more than a generalization of matrix multiplication in higher dimensions, a process known as *tensor contraction*. Given two tensors of orders m and n, the tensor contraction operation always produces a new tensor of order $n + m - 2$. Under this setting, the meaning of a simple transitive sentence can be calculated as follows:

$$\overline{subj\, verb\, obj} = \overrightarrow{subj}^{\mathsf{T}} \times \overline{verb} \times \overrightarrow{obj}, \tag{13}$$

where the symbol $\times$ denotes tensor contraction. Given that $\overrightarrow{subj}$ and $\overrightarrow{obj}$ live in N and $\overline{verb}$ lives in $N \otimes S \otimes N$, the above operation will result in a tensor in S, which represents the sentence space of our choice (for a discussion about S see section Verb and Sentence Spaces).

Tensor-based models provide an elegant solution to the problems of vector mixtures: they are not bag-of-words approaches and they respect the type-logical identities of special words, following an approach very much aligned with the formal semantics perspective. Furthermore, they do not suffer from the space complexity problems of models based on raw tensor product operations (section Tensor Product and Circular Convolution), since the tensor contraction operation guarantees that every sentence will eventually live in our sentence space S. On the other hand, the highly linguistic perspective they adopt has also a downside: in order for a tensor-based model to be fully effective, an appropriate mapping to vector spaces should have been devised for every functional word, such as prepositions, relative pronouns or logical connectives. As we will see in section Treatment of Functional Words and Treatment of Logical Words, this problem is far from trivial; actually it constitutes one of the most important open issues, and at the moment restricts the application of these models on well-defined text structures (for example, simple transitive sentences of the form 'subject-verb-object' or adjective–noun compounds).

[2] Actually, Plate proposes a workaround for the commutativity problem; however, this is not quite satisfactory and not specific to his model, since it can be used with any other commutative operation such as vector addition or point-wise multiplication.

[3] Here, the word *tensor* refers to a geometric object that can be seen as a generalization of a vector in higher dimensions. A matrix, for example, is an order-2 tensor.

The notion of a framework where relational words act as linear maps on noun vectors has been formalized by Coecke et al. [7] in the abstract setting of category theory and compact closed categories, a topic we are going to discuss in more detail in section Categorical Framework for Natural Language. Baroni and Zamparelli's [2] composition method for adjectives and nouns also follows the very same principle.

Deep Learning Models

A recent trend in compositionality of distributional models is based on *deep learning* techniques, a class of machine learning algorithms (usually neural networks) that approach models as multiple layers of representations, where the higher-level concepts are induced from the lower-level ones. For example, Socher et al. [42–44] use recursive neural networks in order to produce compositional vector representations for sentences, with very promising results in a number of tasks. In its most general form, a neural network like this takes as input a pair of word vectors $\overrightarrow{w_1}, \overrightarrow{w_2}$ and returns a new composite vector $\overrightarrow{y}$ according to the following equation:

$$\overrightarrow{y} = g(\mathbf{W}[\overrightarrow{w}_1; \overrightarrow{w}_2] + \overrightarrow{b}), \tag{14}$$

where $[\overrightarrow{w}_1; \overrightarrow{w}_2]$ denotes the concatenation of the two child vectors, $\mathbf{W}$ and $\overrightarrow{b}$ are the parameters of the model, and g is a non-linear function such as tanh. This output, then, will be used in a recursive fashion again as input to the network for computing the vector representations of larger constituents. The general architecture of this model is presented in Fig. 5.

Neural networks can vary in design and topology. Kalchbrenner and Blunsom [20], for example, model sentential compositionality using a *convolutional* neural network in an element-wise fashion. Specifically, the input of the network is a vector representing a single feature, the elements of which are collected across all the word vectors in the sentence. Each layer of the network applies convolutions of kernels of increasing size, producing at the output a single value that will form the corresponding feature in the resulting sentence vector. This method was used for providing sentence vectors in the context of a discourse model, and was tested with success in a task of recognizing dialogue acts of utterance within a conversation. Furthermore, it has been used as a sentence generation apparatus in a machine translation model with promising results [19].

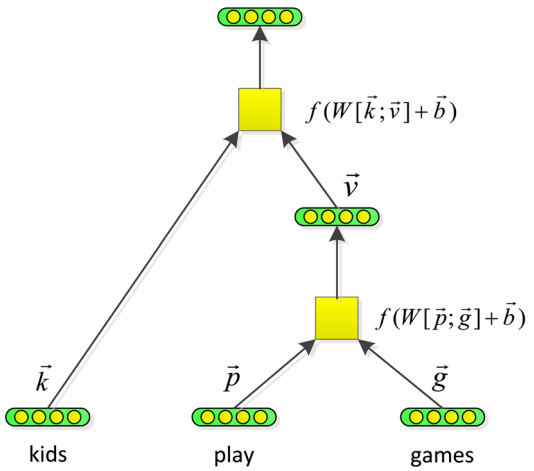


Fig. 5 A recursive neural network with a single layer for providing compositionality in distributional models

On the contrary to the previously discussed compositional approaches, deep learning methods are based on a large amount of pre-training: the parameters $\mathbf{W}$ and $\overrightarrow{b}$ in the network of Fig. 5 must be learned through an iterative algorithm known as *backpropagation*, a process that can be very time-consuming and in general cannot guarantee optimality. However, the non-linearity in combination with the layered approach in which neural networks are based provides these models with great power, allowing them to simulate the behaviour of a range of functions much wider than the linear maps of tensor-based approaches. Indeed, the work of Socher et al. [42–44] has been tested in various paraphrase detection and sentiment analysis tasks, delivering results that by the time of this writing remain state-of-the-art.

A Categorical Framework for Natural Language

Tensor-based models stand in between the two extremes of vector mixtures and deep learning methods, offering an appealing alternative that can be powerful enough and at the same time fully aligned with the formal semantics view of natural language. Actually, it has been shown that the linear-algebraic formulas for the composite meanings produced by a tensor-based model emerge as the natural consequence of a structural similarity between a grammar and finite-dimensional vector spaces. In this section I will review the most important points of this work.

Unifying Grammar and Meaning

Using the abstract mathematical framework of category theory, Coecke et al. [7] managed to equip the distributional models of meaning with compositionality in a way that every grammatical reduction corresponds to a linear map defining mathematical manipulations between vector spaces. In other words, given a sentence $s = w_1 w_2 \cdots w_n$ there exists a syntax-driven linear map f from the context vectors of the individual words to a vector for the whole sentence defined as follows:

$$\overrightarrow{s} = f(\overrightarrow{w_1} \otimes \overrightarrow{w_2} \otimes \cdots \otimes \overrightarrow{w_n}). \tag{15}$$

This result is based on the fact that the base type-logic of the framework, a pregroup grammar [23], shares the same abstract structure with finite-dimensional vector spaces, that of a compact closed category. Mathematically, the transition from grammar types to vector spaces has the form of a strongly monoidal functor, that is, of a map that preserves the basic structure of compact closed categories. The following section provides a short introduction to the category theoretic notions above.

Introduction to Categorical Concepts

Category theory is an abstract area of mathematics, the aim of which is to study and identify universal properties of mathematical concepts that often remain hidden by traditional approaches. A *category* is a collection of objects and morphisms that hold between these objects, with composition of morphisms as the main operation. That is, for two morphisms $A \xrightarrow{f} B \xrightarrow{g} C$, we have $g \circ f : A \to C$. Morphism composition is associative, so that $(f \circ g) \circ h = f \circ (g \circ h)$. Furthermore, every object A has an identity morphism $1_A : A \to A$; for $f : A \to B$ we moreover require that

$$f \circ 1_A = f \quad \text{and} \quad 1_B \circ f = f\,. \tag{16}$$

A *monoidal category* is a special type of category equipped with another associative operation, a monoidal tensor $\otimes : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ (where $\mathcal{C}$ is our basic category). Specifically, for each pair of objects (A, B) there exists a composite object $A \otimes B$, and for every pair of morphisms $(f : A \to C, g : B \to D)$ a parallel composite $f \otimes g : A \otimes B \to C \otimes D$. For a *symmetric monoidal category*, it is also the case that $A \otimes B \cong B \otimes A$. Furthermore, there is a unit object I which satisfies the following isomorphisms:

$$A \otimes I \cong A \cong I \otimes A\,. \tag{17}$$

A monoidal category is *compact closed* if every object A has a left and right adjoint, denoted as A^l, A^r, respectively, for which the following special morphisms exist:

$$\eta^l : I \to A \otimes A^l \quad \eta^r : I \to A^r \otimes A \tag{18}$$

$$\epsilon^l : A^l \otimes A \to I \quad \epsilon^r : A \otimes A^r \to I\,. \tag{19}$$

For the case of a *symmetric compact closed category*, the left and right adjoints collapse into one, so that $A^* := A^l = A^r$.

Both a pregroup grammar and the category of finite-dimensional vector spaces over a base field $\mathbb{R}$ conform to this abstract definition of a compact closed category. A *pregroup grammar* [23] is a type-logical grammar built on the rigorous mathematical basis of a pregroup algebra, i.e. a partially ordered monoid with unit 1, whose each element p has a left adjoint p^l and a right adjoint p^r. These elements of the monoid are the objects of the category, the partial orders are the morphisms, the pregroup adjoints correspond to the adjoints of the category, while the monoid multiplication is the tensor product with 1 as unit. In a context of a pregroup, Eqs. 18 and 19 are transformed to the following:

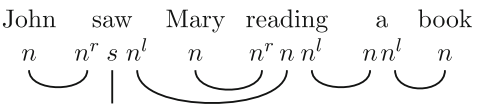
$$p^l p \le 1 \le p p^l \quad \text{and} \quad p p^r \le 1 \le p^r p\,, \tag{20}$$

where the juxtaposition of elements denotes the monoid multiplication. Each element p represents an atomic type of the grammar, for example n for noun phrases and s for sentences. Atomic types and their adjoints can be combined to form compound types, e.g. $n^r s n^l$ for a transitive verb. This type reflects the fact that a transitive verb is something that expects for a noun at its right (the object) and a noun at its left (the subject) in order to return a sentence. The rules of the grammar are prescribed by the mathematical properties of pregroups, and specifically by the inequalities in (20) above. A partial order in the context of a logic denotes implication, so from (20) we derive

$$p^l p \to 1 \quad \text{and} \quad p p^r \to 1\,. \tag{21}$$

These cancellation rules to the unit object correspond to the ϵ-maps of a compact closed category. It also holds that $1p = p = p1$, satisfying the isomorphism of the monoidal unit in (17). A derivation in a pregroup grammar has the form of a reduction diagram, where the cancellation rules (i.e. ϵ-maps) are depicted by lines connecting elements:

(9)



We refer to the compact closed category formed by a pregroup freely generated over a set of basic types (here $\{n, s\}$) by $\mathbf{Preg}_F$. On the other hand, finite-dimensional vector spaces also form a compact closed category, where the vector spaces are the objects, linear maps are the morphisms, the main operation is the tensor product between vector spaces, and the field over which the spaces are formed, in our case $\mathbb{R}$, is the unit. The adjoints are the dual spaces, which are isomorphic to the space itself (hence, on the contrary with $\mathbf{Preg}_F$, finite-dimensional vector spaces form a *symmetric* compact closed category); the ϵ-maps correspond to the inner product between the involved context vectors, as follows:

$$\epsilon^l = \epsilon^r : W \otimes W \to \mathbb{R} :: \sum_{ij} c_{ij}(\overrightarrow{w_i} \otimes \overrightarrow{w_j}) \mapsto \sum_{ij} c_{ij} \langle \overrightarrow{w_i} | \overrightarrow{w_j} \rangle\,. \tag{22}$$

Let us refer to the category of finite-dimensional vector spaces and linear maps over a field as $\mathbf{FVect}_W$, where W is

our basic distributional vector space with an orthonormal basis $\{w_i\}_i$. The transition from grammar type reductions to vector spaces, then, is accomplished by a *strongly monoidal functor* $\mathcal{F}$ of the form:

$$\mathcal{F} : \mathbf{Preg}_F \rightarrow \mathbf{FVect}_W \tag{23}$$

which preserves the compact structure between the two categories so that $\mathcal{F}(A^l) = \mathcal{F}(A)^l$ and $\mathcal{F}(A^r) = F(A)^r$. The categorical framework is agnostic regarding the form of our sentence space S, so in general $\mathcal{F}(n)$ can be different than $\mathcal{F}(s)$. However, in order to keep this presentation simple let us have our functor assigning the same basic vector space W to both of the basic types, as follows:

$$\mathcal{F}(n) = \mathcal{F}(s) = W \tag{24}$$

Furthermore, the complex types are mapped to tensor products of vector spaces:

$$\mathcal{F}(nn^l) = \mathcal{F}(n^r n) = W \otimes W \qquad \mathcal{F}(n^r s n^l) = W \otimes W \otimes W. \tag{25}$$

Similarly, the type reductions are mapped to the compositions of tensor products of identity and ϵ-maps of $\mathbf{FVect}_W$. I will use the case of a transitive sentence as an example. Here, the subject and the object have the type n, whereas the type of the verb is $n^r s n^l$, as described above. The derivation proceeds as follows:

$$n(n^r s n^l)n = (nn^r)s(n^l n) \rightarrow 1s1 = s$$

which corresponds to the following map:

$$\begin{aligned} &\mathcal{F}(n(n^r s n^l)n) = \mathcal{F}(\epsilon_n^r \otimes 1_s \otimes \epsilon_n^l) \\ &= \epsilon_W \otimes 1_W \otimes \epsilon_W : W \otimes (W \otimes W \otimes W) \otimes W \rightarrow W. \end{aligned} \tag{26}$$

The function f in Eq. 15, suggested as a way for calculating the meaning of a sentence, now takes a concrete form strictly based on the grammar rules that connect the individual words. Let us work on the transitive example a bit further: the map of pregroup types to tensors prescribes that the subject and object are vectors (Eq. 24) while the verb is a tensor of order 3 (Eq. 25). So for a simple sentence such as 'dogs chase cats' we have the following geometric entities:

$$\begin{aligned} \overrightarrow{dogs} &= \sum_i c_i^{dogs} \overrightarrow{w_i} \\ \overline{chase} &= \sum_{ijk} c_{ijk}^{chase} (\overrightarrow{w_i} \otimes \overrightarrow{w_j} \otimes \overrightarrow{w_k}) \\ \overrightarrow{cats} &= \sum_k c_k^{cats} \overrightarrow{w_k} \end{aligned}$$

where c_i^v denotes the ith component in vector $\overrightarrow{v}$ and $\overrightarrow{w_i}$ a basis vector of W. Applying Eq. 15 on this sentence will give

$$\begin{aligned} &\mathcal{F}(\epsilon_n^r \otimes 1_n \otimes \epsilon_n^l)(\overrightarrow{dogs} \otimes \overline{chase} \otimes \overrightarrow{cats}) \\ &= (\epsilon_W \otimes 1_W \otimes \epsilon_W)(\overrightarrow{dogs} \otimes \overline{chase} \otimes \overrightarrow{cats}) \\ &= \sum_{ijk} c_{ijk}^{chase} \langle \overrightarrow{dogs} | \overrightarrow{w_i} \rangle \overrightarrow{w_j} \langle \overrightarrow{w_k} | \overrightarrow{cats} \rangle \\ &= \overrightarrow{dogs}^{\mathsf{T}} \times \overline{chase} \times \overrightarrow{cats} \end{aligned} \tag{27}$$

where the symbol $\times$ denotes tensor contraction. Thus we have arrived at Eq. 13, presented in section Tensor-Based Models as a means for calculating the vector of a transitive sentence in the context of a tensor-based model.

The significance of the categorical framework lies exactly in this fact, that it provides an elegant mathematical counterpart of the formal semantics perspective as expressed by Montague [30], where words are represented and interact with each other according to their type-logical identities. Furthermore, it seems to imply that approaching the problem of compositionality in a tensor-based setting is a step towards the right direction, since the linear-algebraic manipulations come as a direct consequence of the grammatical derivation. The framework itself is a high-level recipe for composition in distributional environments, leaving a lot of room for further research and experimental work. Concrete implementations have been provided, for example, by Grefenstette and Sadrzadeh [16] and Kartsaklis et al. [21]. For more details on pregroup grammars and their type dictionary, see [23]. The functorial passage from a pregroup grammar to finite-dimensional vector spaces is described in detail in [22].

Verb and Sentence Spaces

Until now I have been deliberately vague when talking about the sentence space S and the properties that a structure like this should bring. In this section I will try to discuss this important issue in more detail, putting some emphasis on how it is connected to another blurry aspect of the discussion so far, the form of relational words such as verbs. From a mathematical perspective, the decisions that need to be taken regard (a) the dimension of S, that is, the cardinality of its basis; and (b) the form of the basis. In other words, how many and what kind of features will comprise the meaning of a sentence?

This question finds a trivial answer in the setting of vector mixture models; since everything in that approach lives into the same base space, a sentence vector has to share the same size and features with words. It is instructive to pause for a moment and consider what this really mean in practice. What a distributional vector for a word actually shows us is to what extent all other words in the vocabulary are related to this specific target word. If our

target word is a verb, then the components of its vector can be thought as related to the *action* described by the verb: a vector for the verb 'run' reflects the degree to which a 'dog' can run, a 'car' can run, a 'table' can run and so on. The element-wise mixing of vectors $\overrightarrow{dog}$ and $\overrightarrow{run}$ then in order to produce a compositional representation for the meaning of the simple intransitive sentence 'dogs run', finds an intuitive interpretation: the output vector will reflect the extent to which things that are related to dogs can also run; in other words, it shows how *compatible* the verb is with the specific subject.

A tensor-based model, on the other hand, goes beyond a simple compatibility check between the relational word and its arguments; its purpose is to *transform* the noun into a sentence. Furthermore, the size and the form of the sentence space become tunable parameters of the model, which can depend on the specific task in hand. Let us assume that in our model we select sentence and noun spaces such that $S \in \mathbb{R}^s$ and $N \in \mathbb{R}^n$, respectively; here, s refers to the number of distinct features that we consider appropriate for representing the meaning of a sentence in our model, while n is the corresponding number for nouns. An intransitive verb, then, like 'play' in 'kids play', will live in $N \otimes S \in \mathbb{R}^{n \times s}$, and will be a map $f : N \to S$ built (somehow) in a way to take as input a noun and produce a sentence; similarly, a transitive verb will live in $N \otimes S \otimes N \in \mathbb{R}^{n \times s \times n}$ and will correspond to a map $f_{tr} : N \otimes N \to S$. We can now provide some intuition of how the verb space should be linked to the sentence space: the above description clearly suggests that, in a certain sense, the verb tensor should be able to somehow encode the meaning of *every* possible sentence that can be produced by the specific verb, and emit the one that matches better the given input.

Let us demonstrate these ideas using a concrete example, where the goal is to simulate the truth-theoretic nature of formal semantics view in the context of a tensor-based model (perhaps for the purposes of a textual entailment task). In that case, our sentence space will be nothing more than the following:

$$S = \left\{ \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right\} \tag{28}$$

with the two vectors representing $\top$ and $\bot$, respectively. Each individual in our universe will correspond to a basis vector of our noun space; with just three individuals (Bob, John and Mary), we get the following mapping:

$$\overrightarrow{bob} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \quad \overrightarrow{john} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \quad \overrightarrow{mary} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}. \tag{29}$$

In this setting, an intransitive verb will be a matrix formed as a series of truth values, each one of which is associated with some individual in our universe. Assuming for example that only John performs the action of sleeping, then the meaning of the sentence 'John sleeps' is given by the following computation:

$$\overrightarrow{john}^{\mathsf{T}} \times \overline{sleep} = \begin{pmatrix} 0 & 1 & 0 \end{pmatrix} \times \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \top. \tag{30}$$

The matrix for *sleep* provides a clear intuition of a verb structure that encodes all possible sentence meanings that can be emitted from the specific verb. Notice that each row of $\overline{sleep}$ corresponds to a potential sentence meaning given a different subject vector; the role of the subject is to specify which row should be selected and produced as the output of the verb.

It is not difficult to imagine how this situation scales up when we move on from this simple example to the case of highly dimensional real-valued vector spaces. Regarding the first question we posed in the beginning of this section, this implies some serious practical limitations on how large s can be. Although it is generally true that the higher the dimension of sentence space, the subtler the differences we would be able to detect from sentence to sentence, in practice a verb tensor must be able to fit in a computer's memory to be of any use to us; with today's machines, this could roughly mean that $s \leq 300$.[4] Using very small vectors is also common practice in deep learning approaches, aiming to reduce the training times of the expensive optimization process; Socher et al. [43], for example, use 50-dimensional vectors for both words and phrases/sentences.

The second question posed in this section, regarding *what kind* of properties should comprise the meaning of a sentence, seems more philosophical than technical. Although in principle model designers are free to select whatever features (that is, basis vectors of S) they think might serve better the purpose of the model, in practice an empirical approach is usually taken in order to sidestep deeper philosophical issues regarding the nature of a sentence. In a deep learning setting, for example, the sentence vector emerges as a result of an *objective function* based on which the parameters of the model have been optimized. In [42], the objective function assesses the quality of a parent vector (e.g. vector $\overrightarrow{v}$ in Fig. 5) by how faithfully it can be deconstructed to the two original children vectors. Hence, the sentence vector at the top of the diagram in Fig. 5 is a vector constructed in a way to allow the optimal reconstruction of the vectors for its two children, the noun phrase 'kids' and the verb phrase 'play games'. The important

[4] To understand why, imagine that a ditransitive verb is a tensor of order 4 (a ternary function); by taking $s = n = 300$ this means that the required space for just one ditransitive verb would be $300^4 \times 8$ bytes per number $\approx$ 65 gigabytes.

point here is that no attempt has been made to interpret the components of the sentence vector individually; the only thing that matters is how faithfully the resulting vector fulfils the adopted constraints.

In the context of a tensor-based model, the question regarding the form of a sentence space can be recast in the following form: How should we build the sentence-producing maps (i.e. our verbs) in order to output the appropriate form of sentence vector or tensor? Baroni and Zamparelli [2] propose a method for building adjective matrices, which is generally applicable to any relational word. Assuming we want to create a matrix for the intransitive verb 'run', we can collect all instances of this verb occurring together with some subject in the corpus and create a distributional vector for these two-word constructs based on their contexts as if they were single words; each one of these vectors is paired with the vector of the corresponding subject to create a set of the form: $\langle \overrightarrow{dog}, \overrightarrow{dogs\ run} \rangle$, $\langle \overrightarrow{people}, \overrightarrow{people\ run} \rangle$, $\langle \overrightarrow{car}, \overrightarrow{cars\ run} \rangle$ and so on. We can now use linear regression in order to produce an appropriate matrix for 'run' based on these exemplars. Specifically, the goal of the learning process is to find the matrix $\overline{run}$ that minimizes the following quantity:

$$\frac{1}{2m}\sum_{i=1}^{m}\left(\overline{run} \times \overrightarrow{subj_i} - \overrightarrow{subj_i\ run}\right)^2 \tag{31}$$

which represents the total error for all nouns occurred as subjects of the specific intransitive verb. Notice that this time our objective function has a different form from the reconstruction error of Socher et al., but it still exists: the verb must be able to produce a sentence vector that, given an arbitrary subject, will approximate the distributional behaviour of all those two-word elementary exemplars on which the training was based.

Challenges and Open Questions

The previous sections hopefully provided a concise introduction to the important developments that have been noted in the recent years on the topic of compositional distributional models. Despite this progress, however, the provision of distributional models with compositionality is an endeavour that still has a long way to go. This section outlines some important issues that current and future research should face in order to provide a more complete account to the problem.

Evaluating the Correctness of Distributional Hypothesis

The idea presented in section Verb and Sentence Spaces for creating distributional vectors of constructions larger than single words has its roots to an interesting thought experiment, the purpose of which is to investigate the potential distributional behaviour of large phrases and sentences and the extent to which such a distributional approach is plausible or not for longer-than-words text constituents. The argument goes like this: if we had an infinitely large corpus of text, we could create a purely distributional representation of the phrase or sentence, exactly as we do for words, by taking into account the different contexts within which this text fragment occurs. Then the assumption would be that the vector produced by the composition of the individual word vectors should be a synthetic 'reproduction' of this distributional sentence vector. This thought experiment poses some interesting questions: First of all, it is not clear if the distributional hypothesis does indeed scale up to text constituents larger than words; second, even if we assume it does, what 'context' would mean in this case? For example, what would an appropriate context be of a 20-word sentence?

Although there is no such a thing as an 'infinitely large corpus', it would still be possible to get an insight about these important issues if we restrict ourselves to small constructs—say, two-word constituents—for which we can still get reliable frequency counts from a large corpus. In the context of their work with adjectives (shortly discussed in section Verb and Sentence Spaces), Baroni and Zamparelli [2] performed an interesting experiment along these lines using the ukWaC corpus, consisting of 2.8 billion words. As we saw, their work follows the tensor-based paradigm where adjectives are represented as linear maps learnt using linear regression and act on the context vectors of nouns. The composite vectors were compared with observed vectors of adjective–noun compounds, created by the contexts of each compound in the corpus. The results, although perhaps encouraging, are far from perfect: for 25 % of the composed vectors, the observed vector was not even in the top 1,000 of their nearest neighbours, in 50 % of the cases the observed vector was in the top 170, while only for a 25 % of the cases the observed vector was in the top 17 of the nearest neighbours.

As the authors point out, one way to explain the performance is as a result of data sparseness. However, the fact that a 2.8 billion-word corpus is not sufficient for modelling elementary two-word constructs would be really disappointing. As Pulman [37] mentions:

> It is worth pointing out that a corpus of 2.83 billion is already thousands of times as big as the number of words it is estimated a 10-year-old person would have been exposed to [33], and many hundreds of times larger than any person will hear or read in a lifetime.

If we set aside for a moment the possibility of data sparseness, then we might have to start worrying about the

validity of our fundamental assumptions, i.e. that of distributional hypothesis and principle of compositionality. Does the result mean that the distributional hypothesis holds only for individual words such as nouns, but it is not very effective for larger constituents such as adjective–noun compounds? Doubtful, since a noun like 'car' and an adjective–noun compound like 'red car' represent similar entities, share the same structure and occur within similar contexts. Is this then an indication that the distributional hypothesis suffers from some fundamental flaw that limits its applicability even for the case of single words? That would be a very strong and rather unjustified claim to make, since it is undoubtedly proven that distributional models can capture the meaning of words, at least to some extent, for many real-world applications (see examples in section From Words to Sentence). Perhaps we should seek the reason of the sub-optimal performance to the specific methods used for the composition in that particular experiment. The adjectives, for example, are modelled as linear functions over their arguments (nouns they modify), which raises another important question: Is *linearity* an appropriate model for composition in natural language? Further research is needed in order to provide a clearer picture of the expectations we should have regarding the true potential of compositional distributional models, and all the issues raised in this section are very important towards this purpose.

What is 'Meaning'?

Even if we accept that the distributional hypothesis is correct, and a context vector can indeed capture the 'meaning' of a word, it would be far too simplistic to assume that this holds for *every* kind of word. The meaning of some words can be determined by their denotations; it is reasonable to claim, for example, that the meaning of the word 'tree' is the set of all trees, and we are even able to answer the question 'what is a tree?' by pointing to a member of this set, a technique known as *ostensive* definition. But this is not true for all words. In 'Philosophy' (published in 'Philosophical Occasions: 1912–1951', [50]), Ludwig Wittgenstein notes that there exist certain words, like 'time', the meaning of which is quite clear to us until the moment we have to explain it to someone else; then we realize that suddenly we are not able any more to express in words what we certainly know—it is like we have forgotten what that specific word really means. Wittgenstein claims that 'if we have this experience, then we have arrived at the limits of language'. This observation is related to one of the central ideas of his work: that the meaning of a word does not need to rely on some kind of definition; what really matters is the way we use this word in our everyday communications. In 'Philosophical Investigations' [49], Wittgenstein presents a thought experiment:

> Now think of the following use of language: I send someone shopping. I give him a slip marked five red apples. He takes the slip to the shopkeeper, who opens the drawer marked 'apples'; then he looks up the word 'red' in a table and finds a colour sample opposite it; then he says the series of cardinal numbers—I assume that he knows them by heart—up to the word five and for each number he takes an apple of the same colour as the sample out of the drawer. It is in this and similar ways that one operates with words.

For the shopkeeper, the meaning of words 'red' and 'apples' was given by ostensive definitions (provided by the colour table and the drawer label). But what was the meaning of word 'five'? Wittgenstein is very direct on this:

> No such thing was in question here, only how the word 'five' is used.

If language is not expressive enough to describe certain fundamental concepts of our world, then the application of distributional models is by definition limited. Indeed, the subject of this entire section is the concept of 'meaning'—yet, how useful would be for us to use this resource in order to construct a context vector for this concept? To what extent would this vector be an appropriate semantic representation for the word 'meaning'?

Treatment of Functional Words

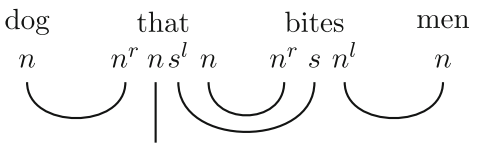
In contrast with content words, like nouns, verbs and adjectives, functional words such as prepositions, determiners or relative pronouns are considered semantically vacuous. In the context of a distributional model, the problem arises from the ubiquitous nature of these words, which means that creating a context vector for preposition 'in', for example, would not be especially useful, since this word can be encountered in almost every possible context. Vector mixture models 'address' this problem by just ignoring all these functional words, a solution which seems questionable given that these words signify specific relations between different text constituents. Under the formal semantics view, for example, a noun phrase such as 'man in uniform' would be represented by the following logical form:

$$(10) \quad \exists x \exists y.[man(x) \wedge uniform(y) \wedge in(x, y)],$$

where the predicate $in(x, y)$ denotes that between the two entities there holds a specific kind of relation. At the other end of the scale, deep learning techniques rely on their brute force and make no distinction between functional words and content words, hoping that the learning process will eventually capture at least some of the correct semantics. To what extent this can be achieved, though, it is still not quite clear.

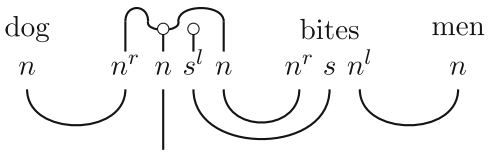
Compositional models that are based on deep semantic analysis, as the categorical framework of Coecke et al. [7], do not allow us to arbitrarily drop a number of words from a sentence based on subjective criteria regarding their meaning, and for a good reason: there is simply no linguistic justification behind such decisions. However, since in that case a functional word is a multi-linear map, we can still manually 'engineer' the inner workings of this structure in a way that conforms to the linguistic intuition we have for the role of the specific word within the sentence. For example, Sadrzadeh et al. [39] use Frobenius algebras over the category of finite-dimensional vector spaces in order to detail the 'anatomy' of subjective and objective relative pronouns. Given that the type of a subjective relative pronoun is $n^r n s^l n$, a typical derivation (here using a pregroup grammar) gets the following form:

(11)



In these cases, the part that follows the relative pronoun acts as a modifier on the head noun. It is possible, then, to define the inner structure of tensor $\overline{that}$ in a way that allows us to pass the information of the noun from the left-hand part of the phrase to the right-hand part, in order to let it properly interact with the modifier part. Schematically, this is depicted by

(12)



In the above diagram, the 'cups' ($\cup$) correspond to ϵ-maps as usual, the 'caps' ($\cap$) represent η-maps (see section A Categorical Framework for Natural Language and Eq. 18), and the dots denote Frobenius operations. Explaining the exact manipulations that take place here is beyond the scope of this article, but intuitively the following things happen:

1. The sentence dimension of the verb is 'killed'; this can be seen as a collapse of the order-3 tensor into a tensor of order 2 along the sentence dimension.
2. The new version of verb, now retaining information only from the dimensions linked to the subject and the object, interacts with the object and produces a new vector.
3. This new vector is 'merged' with the subject, in order to modify it appropriately.

From a linear-algebraic perspective, the meaning of the noun phrase is given as the computation $(\overrightarrow{verb} \times \overrightarrow{obj}) \odot \overrightarrow{subj}$, where $\times$ denotes matrix multiplication and $\odot$ point-wise multiplication. Note that this final equation does not include any tensor for 'that'; the word solely acts as a 'router', moving information around and controlling the interactions between the content words. The above treatment of relative pronouns is flexible and possibly opens a door for modelling other functional words, such as prepositions. What remains to be seen is how effective it can be in an appropriate large-scale evaluation, since the current results, although promising, are limited to a small example dataset.

Treatment of Logical Words

Logical words, like 'not', 'and ', 'or', constitute a different category of functional words that need to be addressed. In the context of distributional semantics, a natural tool for achieving this is *quantum logic*, originated by Birkho and von Neumann [3], where the logical connectives operate on linear subspaces of a vector space. Under this setting, the negation of a subspace is given by its orthogonal complement. Given two vector spaces A and V, with A to be a subspace of V (denoted by $A \leq V$), the orthogonal complement of A is defined as follows:

$$A^{\perp} = \{\overrightarrow{v} \in V : \forall \overrightarrow{a} \in A, \langle \overrightarrow{a} | \overrightarrow{v} \rangle = 0\}\,. \tag{32}$$

The negation of a word vector $\overrightarrow{w}$, then, can be defined as the orthogonal complement of the vector space $\langle w \rangle = \{\lambda \overrightarrow{w} : \lambda \in \mathbb{R}\}$ generated by $\overrightarrow{w}$. Widdows [48] applies this idea on keywords, in the context of an information retrieval query. Specifically, he models negation by using projection to the orthogonal subspace, with very good results compared to traditional Boolean operators. Suppose for example that we need to include in our query an ambiguous word like 'rock', but we also wish to restrict its meaning to the geological sense (so we want the vector space counterpart of the Boolean query `rock NOT music`). We can achieve this by projecting the vector for 'rock' onto the orthogonal subspace of the vector for 'music', an operation that eventually will retain the components of the vector of 'rock' that are not related to music. For two arbitrary words w_1 and w_2, this can be achieved as follows:

$$\overrightarrow{w_1} \wedge \neg \overrightarrow{w_2} = \frac{\langle \overrightarrow{w_1} | \overrightarrow{w_2} \rangle}{|\overrightarrow{w_2}|^2} \overrightarrow{w_2}\,. \tag{33}$$

In quantum logic, disjunction is modelled as the vector sum of subspaces: $V + W$ is the smallest subspace that includes both V and W. That is, an expression like $w_1 \, OR \, w_2$ is represented by the subspace:

$$\langle w_1 \rangle + \langle w_2 \rangle = \{\lambda_1 \overrightarrow{w_1} + \lambda_2 \overrightarrow{w_2} : \lambda_1, \lambda_2 \in \mathbb{R}\}\,, \tag{34}$$

where again $\langle w \rangle$ is the vector space generated by vector $\overrightarrow{w}$. Finally, the conjunction of two subspaces is their intersection—the set of vectors belonging to both of them. Widdows [48] discusses some of the subtleties of a

practical implementation based on these concepts, such as difficulties on evaluating the similarity of a vector with a subspace. Furthermore, it should be mentioned that quantum logic has some important differences from classical logic, notably it does not adhere to the distributive law, so in general:

$$A \wedge (B \vee C) \neq (A \wedge B) \vee (A \wedge C)\,. \tag{35}$$

The failure of distributivity law produces another unwelcome side effect: quantum logic has no way to represent material implication, which in the context of a propositional logic is given by the following rule:

$$P \rightarrow Q \Leftrightarrow \neg P \vee Q\,. \tag{36}$$

An excellent introduction to quantum logic, specifically oriented to information retrieval, can be found in [47].

Quantification

Perhaps the single most important obstacle for constructing proper compositional vectors for phrases and sentences is quantification—a concept that is incompatible with distributional models. A quantifier operates on a number of individual entities by counting or enumerating them: *all* men, *some* women, *three* cats and *at least* four days. This fits nicely in the logical view of formal semantics. Consider for example the sentence 'John loves every woman', which has the following logical form:

$$(13) \quad \forall x(woman(x) \rightarrow loves(john, x)).$$

Given that the set of women in our universe is $\{alice, mary, helen\}$, the above expression will be true if relation *loves* includes the pairs $(john, alice), (john, mary), (john, helen)$. Unfortunately, this treatment does not make sense in vector space models since they lack any notion of individuality; furthermore, creating a context vector for a quantifier is meaningless. As a result, quantifiers are just considered 'noise' and ignored in the current practice, producing unjustified equalities such as $\overrightarrow{every\,man} = \overrightarrow{some\,man} = \overrightarrow{man}$. The extent to which vector spaces can be equipped with quantification (and whether this is possible or not) remains another open question for further research. Preller [36] provides valuable insights on the topic of moving from functional to distributional models and how these two approaches are related.

Some Closing Remarks

Compositional distributional models of meaning constitute a technology with great potential, which can drastically influence and improve the practice of natural language processing. Admittedly our efforts are still in their infancy, which should be evident from the discussion in section Challenges and Open Questions. For many people, the ultimate goal of capturing the meaning of a sentence in a computer's memory might currently seem rather utopic, something of theoretical only interest for the researchers to play with. That would be wrong; computers are already capable of doing a lot of amazing things: they can adequately translate text from one language to another, they respond to vocal instructions, they score to TOEFL[5] tests at least as well as human beings do (see, for example, [24]), and—perhaps the most important of all—they offer us all the knowledge of the world from the convenience of our desk with just few mouse clicks. What at least I hope is apparent from the current presentation is that compositional distributional models of meaning is a technology that slowly but steadily evolves to a *useful tool*, which is after all the ultimate purpose of every scientific research.

Acknowledgments This paper discusses topics that I study and work on the past two and a half years, an endeavour that would be doomed to fail without the guidance and help of Mehrnoosh Sadrzadeh, Stephen Pulman and Bob Coecke. I would also like to thank my colleagues in Oxford Nal Kalchbrenner and Edward Grefenstette for all those insightful and interesting discussions, as well as Anne Preller for her useful comments on quantification. The contribution of the two anonymous reviewers to the final form of this article was invaluable; their suggestions led to a paper which has been tremendously improved compared to the original version. Last, but not least, support by EPSRC Grant EP/F042728/1 is gratefully acknowledged.

References

1. Bach E (1976) An extension of classical transformational grammar. In: Proceedings of the conference at Michigan State University on problems in linguistic metatheory, Michigan State University, Lansing, p 183–224
2. Baroni M, Zamparelli R (2010) Nouns are vectors, adjectives are matrices. In: Proceedings of conference on empirical methods in natural language processing (EMNLP), Seattle, p 1427–1432
3. Birkhoff G, von Neumann J (1936) The logic of quantum mechanics. Ann Math 37:823–843
4. Blacoe W, Lapata M (2012) A comparison of vector-based representations for semantic composition. In: Proceedings of the joint conference on empirical methods in natural language processing and computational natural language learning. Association for Computational Linguistics, Jeju Island, Korea, p 546–556
5. Bos J, Markert K (2006) When logical inference helps determining textual entailment (and when it doesn't). In: Proceedings of the second PASCAL challenges workshop on recognizing textual entailment, Citeseer, Venice, Italy
6. Clark S, Pulman S (2007) Combining symbolic and distributional models of meaning. In: Proceedings of the AAAI spring symposium on quantum interaction, p 52–55, Stanford, California, March 2007
7. Coecke B, Sadrzadeh M, Clark S (2010) Mathematical foundations for distributed compositional model of meaning. Lambek Festschrift. Ling Anal 36:345–384

[5] Test of English as a Foreign Language.

8. Curran J (2004) From distributional to semantic similarity. PhD thesis, School of Informatics, University of Edinburgh
9. Dunning T (1993) Accurate methods for the statistics of surprise and coincidence. Comput Ling 19(1):61–74
10. Erk K, Padó S (2008) A structured vector–space model for word meaning in context. In: Proceedings of conference on empirical methods in natural language processing (EMNLP), p 897–906
11. Ferraresi A, Zanchetta E, Baroni M, Bernardini S (2008) Introducing and evaluating ukWaC, a very large web-derived corpus of English. In: Proceedings of the 4th web as corpus workshop (WAC-4) Can we beat Google, p 47–54
12. Firth J (1957) A synopsis of linguistic theory 1930–1955. In: Studies in linguistic analysis, Philological Society, Oxford, p 1–32
13. Frege G (1980a) The foundations of arithmetic: a logico-mathematical enquiry into the concept of number (Translation: Austin, J.L.). Northwestern Univ Press, Evanston
14. Frege G (1980b) Letter to jourdain. In: Gabriel G (ed) Philoshopical and mathematical correspondence. Chicago University Press, Chicago, pp 78–80
15. Graff D, Kong J, Chen K, Maeda K (2003) English gigaword. Linguistic Data Consortium, Philadelphia
16. Grefenstette E, Sadrzadeh M (2011) Experimental support for a categorical compositional distributional model of meaning. In: Proceedings of conference on empirical methods in natural language processing (EMNLP), p 1394–1404
17. Grefenstette G (1994) Explorations in automatic thesaurus discovery. Springer, Heidelberg, Germany
18. Harris Z (1968) Mathematical structures of language. Wiley, New York
19. Kalchbrenner N, Blunsom P (2013a) Recurrent continuous translation models. In: Proceedings of the conference on empirical methods in natural language processing (EMNLP), Association for Computational Linguistics, Seattle, USA
20. Kalchbrenner N, Blunsom P (2013b) Recurrent convolutional neural networks for discourse compositionality. In: Proceedings of the workshop on continuous vector space models and their compositionality, Bulgaria, Sofia
21. Kartsaklis D, Sadrzadeh M, Pulman S (2012) A unified sentence space for categorical distributional-compositional semantics: theory and experiments. In: Proceedings of 24th international conference on computational linguistics (COLING 2012): Posters, The COLING 2012 Organizing Committee, Mumbai, India, p 549–558
22. Kartsaklis D, Sadrzadeh M, Pulman S, Coecke B (2014) Reasoning about meaning in natural language with compact closed categories and frobenius algebras. In: Chubb J, Eskandarian A, Harizanov V (eds.) Logic and algebraic structures in quantum computing and information, Association for Symbolic Logic Lecture Notes in Logic, Cambridge University Press, Cambridge (To appear)
23. Lambek J (2008) From word to sentence. Polimetrica, Milan
24. Landauer T, Dumais S (1997) A solution to Plato's problem: the latent semantic analysis theory of acquision, induction, and representation of knowledge. Psychol Rev 104(2):211–240
25. Lin D (1998) Automatic retrieval and clustering of similar words. In: Proceedings of the 17th international conference on Computational linguistics, vol. 2. Association for Computational Linguistics, Morristown, p 768–774
26. Lowe W (2001) Towards a theory of semantic space. In: Proceedings of the 23rd Annual conference of the Cognitive Science Society, Edinburgh, p 576–581
27. Lowe W, McDonald S (2000) The direct route: mediated priming in semantic space. In: Proceedings of the 22nd Annual conference of the Cognitive Science Society. Philadelphia, PA, p 675–680
28. Manning C, Raghavan P, Schütze H (2008) Introduction to information retrieval. Cambridge University Press, Cambridge
29. Mitchell J, Lapata M (2008), Vector-based models of semantic composition. In: Proceedings of the 46th Annual meeting of the Association for Computational Linguistics, Columbus, p 236–244
30. Montague R (1970a) English as a formal language. Linguaggi nella società e nella tecnica. Edizioni di Comunità, Milan
31. Montague R (1970b) Universal grammar. Theoria 36:373–398
32. Montague R (1973) The proper treatment of quantification in ordinary english. In: Hintikka J et al (eds.) Approaches to natural language, Reidel, Dordrecht, p 221–242
33. Moore RK (2003) A comparison of the data requirements of automatic speech recognition systems and human listeners. In: Interspeech: 8th European conference of speech communication and technology, ISCA, vol 3. Geneva, Switzerland, p 2582–2584
34. Padó S, Lapata M (2007) Dependency-based construction of semantic space models. Comput Ling 33(2):161–199
35. Plate T (1991) Holographic reduced representations: convolution algebra for compositional distributed representations. In: Proceedings of the 12th international joint conference on artificial intelligence, Morgan Kaufmann, San Mateo, CA, p 30–35
36. Preller A (2013) From functional to compositional models. In: Proceedings of the 10th conference of quantum physics and logic (QPL 2013), Barcelona, Spain
37. Pulman S (2013) Combining compositional and distributional models of semantics. In: Heyden C, Sadrzadeh M, Grefenstette E (eds) Quantum physics and linguistics: a compositional, diagrammatic discourse. Oxford University Press, Oxford
38. Rubenstein H, Goodenough J (1965) Contextual correlates of synonymy. Commun ACM 8(10):627–633
39. Sadrzadeh M, Clark S, Coecke B (2013) The frobenius anatomy of word meanings I: subject and object relative pronouns. J Logic Comput 23(6):1293–1317
40. Schütze H (1998) Automatic word sense discrimination. Comput Ling 24:97–123
41. Smolensky P (1990) Tensor product variable binding and the representation of symbolic structures in connectionist systems. Artif Intell 46:159–216
42. Socher R, Huang E, Pennington J, Ng A, Manning C (2011) Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. Adv Neural Inf Process Syst 24:801–809
43. Socher R, Huval B, Manning CD, Ng AY (2012) Semantic compositionality through recursive matrix-vector spaces. In: Conference on empirical methods in natural language processing, p 1201–1211
44. Socher R, Manning C, Ng A (2010) Learning continuous phrase representations and syntactic parsing with recursive neural networks. In: Proceedings of the NIPS-2010 deep learning and unsupervised feature learning workshop, p 1–9
45. Thater S, Fürstenau H, Pinkal M (2010) Contextualizing semantic representations using syntactically enriched vector models. In: Proceedings of the 48th Annual meeting of the Association for Computational Linguistics, Uppsala, Sweden, p 948–957
46. Thater S, Fürstenau H, Pinkal M (2011) Word meaning in context: a simple and effective vector model. In: Proceedings of the 5th international joint conference of natural language processing. Asian Federation of Natural Language Processing, Chiang Mai, Thailand, p 1134–1143
47. van Rijsbergen K (2004) The geometry of information retrieval. Cambridge University Press, Cambridge
48. Widdows D (2003) Orthogonal negation in vector spaces for modelling word-meanings and document retrieval. In: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics, vol. 1. Association for Computational Linguistics, p 136–143
49. Wittgenstein L (1963) Philosophical investigations. Blackwell, Oxford
50. Wittgenstein L (1993) Philosophy. In: Klagge J, Nordmann A (eds.) Philosophical occasions 1912–1951, Hackett, Indianapolis, p 171