23 Vector-based and Neural Models of Semantics

WILLEM ZUIDEMA AND PHONG LE

There is a rich tradition in modeling the meanings of words, phrases, sentences, and discourse, going back, in some cases, thousands of years. For most of that long history, the dominant approach has been an approach based on some form of symbolic logic. This approach has been very successful, in many respects, but it has proven difficult to directly relate such models of semantics to research in (experimental) psychology and cognitive neuroscience. In this chapter, we discuss an alternative approach to modeling meaning based on numerical vectors, using tools from some other branches of mathematics: linear algebra and differential calculus. These models are very attractive from the neural perspective, because numerical vectors are the natural representation for patterns of neural activity, as well as from the learning perspective, because such vectors are compatible with artificial neural network models, and, if certain conditions are met, rich sets of tools for optimization (learning) developed in this paradigm become available. We will get back to these attractive properties in section 1.4.

But are numerical vectors not too weak a representation to account for the intricacies of natural language semantics? Hasn't the rich tradition in lexical and compositional semantics revealed the need to have nontrivial, structured, symbolic models? The goal of this chapter is to show that we can have our cake and eat it: we can use vectorial representations that are compatible with neuroscience and that are optimized using techniques from the neural network toolbox, while still being able to account for nontrivial semantic phenomena. Vector-space models therefore form a class that is a key intermediary between neural and behavioral data and the rich modeling traditions in semantics (and are, we argue, complementary to symbolic approaches, rather than replacements for them).

1. Words: Vector-Space Models of Lexical Semantics

Many approaches that tackle the problem of how to represent word meaning by vectors, are based on the insight of Firth: "You shall know a word by the company it keeps" (Firth, 1957). It means that the meaning of a word is determined by (or at least is correlated with) the words that surround it, and word meanings can thus be computed by analyzing the *distribution* of words, their occurrence and co-occurrence frequencies, in large text corpora. We start our discussion of vectorial models of semantics by looking in some more detail at *distributional* methods; in section 1.3 we also briefly mention nondistributional ways to determine vectorial meaning representations.

1.1. An Example To illustrate the key ideas behind distributional semantics, consider the following example (used in from slides[1] of S. Evert). Imagine that we are reading an English book and encounter the strange word *bardiwac* in some sentences:

- He handed her her glass of *bardiwac*.
- Beef dishes are made to complement the *bardiwac*.
- Nigel staggered to his feet, face flushed from too much *bardiwac*.
- Malbec, one of the lesser-known *bardiwac* grapes, responds well to Australia's sunshine.
- I dined off bread and cheese and this excellent *bardiwac*.
- The drinks were delicious: blood-red *bardiwac* as well as light, sweet Rhenish.

Although we have no idea about the meaning of the word *bardiwac*, we can guess it using contexts. From the first sentence, we can guess that *bardiwac* is a kind of (drinkable) liquid. The second sentence suggests that it is drunk with beef dishes. With the third sentence, it seems that *bardiwac* can make us drunk. And so on. So finally, we can guess (with a very high confidence) that *bardiwac* is a kind of red strong wine.

Because we are human beings, we can guess the meaning of an unknown word easily. How can we simulate this process on a computer? The intuitions behind distributional semantics suggest a computational model can be based on counts of co-occurrences of words. In the following, we will work out a simple formalization of this intuition (before mentioning a number of recent

—-1
—0
—+1

edge of the sword each of this
Lets fall his sword before your Highness'
it me; my sword shall soon dispatch
Sooner this sword shall plough thy
claps me his sword upon the table
I see a sword out, my finger
What, the sword and the word!
Even with the sword that kill'd thee.
If to my sword his fate be
can shake my sword or hear the
He who the sword of heaven will
She takes a sword and runs at
I and my sword will earn our
It eats the sword it fights with.
But for thy sword and fortune, trod
For what the sword cuts down or

u	$\#(u\|w)$	$P(u\|w)$	$P(u\|w)/P(u)$
claps	1	0.002	1.333
cuts	1	0.002	1.333
dispatch	1	0.002	1.333
drink	0	0.0	0.0
down	1	0.002	1.333
earn	1	0.002	1.333
eats	1	0.002	1.333
fall	1	0.002	1.333
fate	1	0.002	1.333
fights	1	0.002	1.333
finger	1	0.002	1.333
fortune	1	0.002	1.333
hear	1	0.002	1.333
heaven	1	0.002	1.333
Highness	1	0.002	2.666
kill'd	1	0.002	1.333
plough	1	0.002	1.333
refuse	0	0.0	0.0
see	1	0.002	1.333
shake	1	0.002	1.333
shall	2	0.004	0.987
table	1	0.002	1.333
takes	1	0.002	1.333
trod	1	0.002	1.333

$$\begin{pmatrix} 1.8 \\ 1.3 \\ 0.8 \\ 0.9 \\ 0.2 \\ 2.8 \\ 2.0 \\ 0.1 \\ 1.7 \\ 1.8 \\ 2.5 \\ 0.5 \\ 0.7 \end{pmatrix}$$

FIGURE 23.1 (Left) An example toy corpus (a random selection of word sequences containing the word *sword* in the Shakespeare texts on Project Gutenberg, with a three-word context window). (Right) Co-occurrence frequencies with *sword* of various other words u in a window of size $k=3$, conditional probabilities, hypothetical PMI values (illustrating that a high-frequency function word such as *shall* is less informative than a low-frequency word such as *Highness*) and the resulting hypothetical word embedding vector (after dimensionality reduction).

variants, including neural network approaches that *predict* rather than *count*).

For simplicity, we define a *context* of a target word in a sentence as the set of all words near the target word, that is, within a *window* of k words before or after the target word (if $k=\infty$, the context is the entire sentence.) Let ν be a vocabulary. A very simple method is to count how many times a word $u \in \nu$ co-occurs with the target word $w \in \nu$ in the k-sized window. This is equivalent to estimating the conditional distribution $Pr(U=u|W=w)$ using maximum likelihood. An illustration is shown in figure 23.1. First, we collect sentences containing the target word w. We then for each word u count how many times u co-occurs with w within a defined window size k. Finally, we extract a vector containing all of those counts, which is used to represent the target word w. We estimate the conditional distribution simply with:

$$P(U=u|W=w) = \frac{\#u \text{ and } w \text{ co-occur}}{\#w} \quad (1)$$

However, in order to come up with useful vectors, there are some details that should be considered. First, which u's should be counted. It is clear that function words (e.g., *a*, *an*, *how*, *what*), which are extremely frequent, carry little information about the meanings of target words. By contrast, rare words tend to contribute more important information. We thus need one more step called a *weighting scheme*. One of popular schemes is weighting with pointwise mutual information (PMI), which corresponds to weighting with the inverse of the probability that u appears in the corpus, $P(U=u)$. Intuitively, this means that rare words get higher weights than frequent words do. The embedding $\vec{e}(w)$ of word w (before dimensionality reduction) is then given as:

$$\vec{e}(w)[u] = PMI(u,w) = \log\left(\frac{(P(U)=u \mid W=w)}{P(U=u)}\right) \quad (2)$$

In practice, the resulting vectors are high dimensional (easily many thousands of dimensions or more). Therefore, a dimensionality reduction method such as principal component analysis or non-negative matrix factorization is often employed as a final step to compute reduced embeddings $\vec{e'}$ (see Dinu & Lapata, 2010, for an example).

The model we just derived is a simple instance of a *vector-space model*, where every word is assigned a vector (based on co-occurrence statistics, or *distributional information*), and the set of vectors reflects similarities and differences in the contexts in which words find themselves. In distributional semantics this distributional similarity is, in turn, taken to correlate with semantic similarity (the *distributional hypothesis*).

1.2. A Variety of Methods in Distributional Semantics

Many variants of distributional semantics models exist (see Lowe, 2001, for an overview of the relevant dimensions at which they differ, and Erk, 2012, for a review of the state of the art a decade later). In cognitive science, the still most well-known variant is Latent Semantic Analysis (Landauer & Dumais, 1997), which differs from the model sketched in section 1.1 in that it counts how often a word is used in each of very many documents (i.e., a word–document matrix rather than a word–word matrix is computed; additionally tf/idf rather than PMI is the weight function, and singular value decomposition is used for dimensionality reduction).

Depending on how distributional context is defined (e.g., sentence context or document context) and on how different aspects of the context are weighted (e.g., with distance, or with position to the left or right of the target word), the configurations differ (as do their applicabilities in specific tasks).

In recent years, a class of alternative models has become popular where the key idea is that embeddings are optimized such that they allow prediction of missing or next words (e.g., Collobert et al., 2011, Mikolov, Sutskever, Chen, Corrado, & Dean, 2013). This approach can be illustrated with "filling in blanks" questions in Test of English as a Foreign Language and International English Language Testing System tests, as in the following example:

__________ is the study of numbers, equations, functions, and geometric shapes and their relationships.

a. Physics
b. Mathematics
c. Geography
d. Theology

To arrive at the correct answer in such a task (here: "mathematics"), a model needs (a) to summarize the relevant aspects of meaning of all the words in the given sentence and the way they are put together, (b) have a representation for each of the candidate answers, and (c) have a way to evaluate how well (b) matches (a). Success at this task thus implies knowledge of the meaning of all the words; in a vector-space model optimized for this task, vectors for words should thus reflect their meanings.

More formally: let us assume that each word $v \in \mathrm{V}$ is represented by a vector $\mathbf{v} \in \mathrm{R}^d$ and there is a mechanism for computing the probability $P(W=w|U=(u_1, u_2, \ldots, u_l))$ of the event that a target word w appears in a context $(u_1, u_2, \ldots, u_l)$. We are going to find a vector $\mathbf{v}$ for each word v such that those probabilities are as high as possible for each w and its context $(u_1, u_2, \ldots, u_l)$.

Bengio, Ducharme, Vincent, and Janvin (2003) provided, to our knowledge, the very first approach in this class. Mikolov, Karafiát, Burget, Černocký, and Khudanpur (2010) employed (simple) recurrent network networks. The very popular *word2vec* models, introduced by Mikolov, Chen, Corrado, & Dean (2013), are optimized to predict the target word given the context word embeddings (the continuous bag of words model), or vice versa, to predict context words given the target word embedding (the skipgram model). Pennington, Socher, and Manning (2014) presented a method called *GloVe* and showed that it performs on par with the word2vec method on many tasks. Interestingly, Levy and Goldberg (2014) argued that prediction and count-based approaches might, mathematically, not be so different after all. They derive a count-based objective function that the word2vec-skipgram method is approximating.

1.3. Successes: Proximity and Cognitive Semantic Similarity The most natural property of continuous vector spaces is that they immediately let us think about sets of words in terms of their mutual distances: for every two points in a vector space, we can compute the distance, and, inversely, proximity, using standard distance metrics from algebra and geometry. In vector-space models of semantics, we want *proximity* in that space to reflect semantic *similarity*. A popular distance metric is cosine, which equals 1 if two vectors are pointing in exactly the same direction and 0 if the two vectors are orthogonal. The cosine of two vectors w and v can be computed as the sum of products of the corresponding elements scaled by the vector lengths:

$$\cos(\vec{w}, \vec{v}) = \frac{\sum_{i=1}^{n} w_i v_i}{\|\vec{w}\| \cdot \|\vec{v}\|}$$

With a method for computing embeddings (say: word2vec–continuous bag of words), a corpus (say: English Wikipedia) and a distance metric (say: cosine) in place, we can start to investigate whether the proximity of the resulting vectors indeed corresponds to semantic similarity. In figure 23.2, we give a few examples of the nearest neighbors in such a word vector model of an arbitrary noun, verb, and adjective. These tables illustrate that words find themselves in the neighborhood of other words with, intuitively, strong semantic similarity.

And indeed, a large literature, from the 1990s onward, has demonstrated that word vectors from distributional semantics can (with the right setting of the parameters) strongly correlate with human similarity and typicality judgments and can be strongly predictive of semantic priming effects. Recent large-scale comparisons (Baroni & Lenci, 2010, Pereira, Gerhsman, Ritter, & Botvinick, 2016) confirm this point and moreover show that the GloVE and word2vec methods now so popular in natural language processing (NLP), correlate even better with human data of various sorts than classical methods such as Latent Semantic Analysis. (More recently,

dizziness	0.85	write	0.77	blue	0.76
nausea	0.81	learn	0.47	white	0.70
drowsiness	0.81	literacy	0.45	yellow	0.69
convulsions	0.78	print	0.40	red	0.65
vomiting	0.78	writing	0.39	purple	0.62
spasms	0.77	text	0.39	orange	0.60
constipation	0.77	edit	0.38	pink	0.56
migraine	0.77	how	0.38	magenta	0.54
cramps	0.77	readers	0.37	dark	0.52
duodenal	0.76	find	0.37	multicoloured	0.52
A NNs of "headache"		B NNs of "read"		C NNs of "green"	

FIGURE 23.2 Nearest neighbors (NNs) of three words with cosine similarities, using embeddings obtained by applying word2vec on the English Wikipedia corpus. This figure was created using data from the "Word2vec playground" (http://deeplearner.fz-qqq.net/) in June 1997 (site no longer available; a similar service is available at, e.g., https://rare-technologies.com/word2vec-tutorial/).

they have also been successfully employed in other areas of linguistics, including historical linguistics (e.g., Frermann & Lapata, 2016; Hamilton, Leskovec, & Jurafsky, 2016).

Mitchell et al. (2008) pioneered the use of distributional methods in analyzing neuroimaging data and showed that they can be used to predict, with some success, BOLD-responses of subjects exposed to word-picture pairs of concrete nouns such as *celery* or *airplane*. In the original study, a very simple distributional model was used, where the nouns used as stimuli are represented with a vector recording co-occurrence frequencies with 25 hand-picked verbs (such as *swallow* and *fly*). Later work (Murphy, Talukdar, & Mitchell, 2012) showed that more advanced techniques give even better results, with the best results for a model that makes use of the grammatical structure of sentences in which words find themselves (modeled using labeled dependency parses). These studies use a leave-2-out paradigm, where a model is evaluated on its ability to decide, based on the functional MRI (fMRI) data, which of two words were presented to the subject, given labeled fMRI data for all remaining words. The baseline performance for random guessing is 50%; the best models score around 83%.

Most successful demonstrations of vector-space models are thus based on distributional semantics, using language-internal statistics about words co-occurring with other words. However, it is clear that the meaning of words must also somehow be grounded in the senses and the world. Vector-space models are not restricted to distributional information. Vectors can also encode other types of information and are compatible with approaches based on embodiment and/or conceptual spaces, although it has been difficult to define this type of information at similar scales and levels of precision as the automatic, corpus-based methods for distributional semantics have made possible. Recent progress in vectorial models of vision and object recognition, however, provides a possible resource to fill this gap (see Baroni, 2016, for a recent review of the relevant literature in this area).

1.4. THEORETICAL ADVANTAGES OF VECTORIAL REPRESENTATIONS Vector representations of meaning are not only successfully used in practice for a variety of tasks, they are also attractive from a theoretical perspective as an intermediary among linguistic, cognitive science, and neuroscientific notions of meaning. This is first and foremost because numerical vectors are widely used to describe the activity of a layer of neurons, where the value of an element in the vector is representing the firing rate of a single neuron. But vector representations are not limited to such one-to-one mappings and to rate-coding. We can also think of an element in the vector as representing one of the other parameters of a neuron besides firing rate: oscillation phase, depletion of chemicals, changes in synaptic strength, and so on. Each neuron is then represented by multiple elements in the vector. Alternatively, one element might represent the average activity of a (large) group of neurons (e.g., a voxel in fMRI data) or of some other aggregate variable representing (part of) the information a group of neurons codes for. Interestingly, choosing between different neural interpretations is not always necessary (although it becomes very relevant when relating vectors to empirical neural data or trying to use neurobiological findings to constrain the model space; for example, when selecting an appropriate similarity metric or learning procedure).

For learning, numerical vectors are an attractive representation as they allow optimization using standard tools from machine learning. And, indeed, in recent *deep learning* work, many successful applications

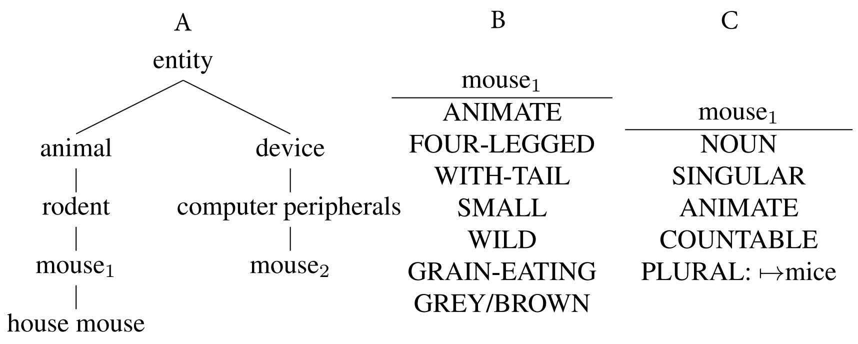

FIGURE 23.3 Examples of the typical information included in symbolic lexicons about words: (A) words might be polysemous, motivating multiple entries in a lexicon. Each entry can be placed in an ontology, detailing superset and subset or other relations it has with other entries. (B) The meaning of words can often be broken down into smaller meaning components/semantic primitives, allowing similarity relations between meanings to be made explicit. (C) The lexical entry often also contains syntactic information about the word form, regulating its behavior in when combined with other words to form sentences.

of this toolbox to questions relating to natural language meaning are reported, often exploiting the huge amount of texts available on the Internet.

1.5. LIMITATIONS OF SIMILARITY MODELS So far we have discussed successes in distributional semantics based on measuring proximity between word vectors, mostly treating semantic similarity as a scalar property, that is, as something that can be described with a single number. However, semantics, even at the word level, involves much more than similarity. Symbolic models of lexical semantics also try to account for phenomena such as the fact that one word might have multiple meanings (polysemy) and that the meanings of words have systematic *relations* with each other such as hypernymy (subset-superset relations, such as the relation between *animal* and *dog*), antonymy (being each other's opposite, such as *large* vs. *small*), and differing in degrees (such as in comparatives, *small, smaller, smallest*). Moreover, symbolic models of the lexicon also carry information about the role that a word plays in a sentence, such as its syntactic category (part-of-speech tag) and morphological information. Figure 23.3 illustrates some of the information about a word like *mouse* that a symbolic lexicon would specify.

1.6. LEXICAL RELATIONS IN VECTOR-SPACE MODELS Much current work is devoted to working out how such phenomena can be accounted for in vector-space models. An important reason for optimism in the field about the ability of vector-space models to account for such aspects of meaning, is the discovery of Mikolov and colleagues (Mikolov, Yih, & Zweig, 2013) that many of the mentioned relations between words, including the morphosyntactic relations between words, seem to be reflected by geometric relations (mostly linear shifts) in the vector space. Mikolov, Yih, and Zweig's by now most famous example is the demonstration that the linear shift in vector space to go from *king* to *queen* is very similar to the linear shift to go from *man* to *woman*, such that we can predict the vector for *queen* using addition and subtraction:

$$\overrightarrow{king} - \overrightarrow{man} + \overrightarrow{woman} = \overrightarrow{queen}$$

In figures 23.4 and 23.5, we show a number of similar predictions, using again the word2vec model. These examples try to capture several part-whole relations, the comparative transform, the past tense and present participle morphology, antonymy relations, and the regular plural morphology in English. These examples do not prove that the mentioned relations are consistently corresponding to linear shifts in vector space, but they do illustrate that the assignment of vectors to words is quite structured and that the high-dimensional vector space allows for many relations to be captured in this way.

Much more sophisticated analysis of these relation can be found in, for instance, Henderson and Popa (2016), who showed that word2vec is closely related to models that optimize for lexical entailment and Cotterell, Schutze, and Eisner (2016), who extended distributional methods to account for word morphology.

2. *Sentences: Vector-Space Models of Compositional Semantics*

Perhaps the biggest challenge for vector-space models of semantics is how to compute meaning representation for sentences and multiword phrases. In theory, the distributional statistics used for computing word representations can be applied to larger items such as phrases

berlin	0.64	bologna	0.56	legs	0.53	inches	0.56
munich	0.59	turin	0.55	toes	0.53	meters	0.55
leipzig	0.58	venice	0.53	feet	0.52	tall	0.53
vienna	0.55	milan	0.52	instep	0.52	weighs	0.53
dresden	0.53	florence	0.51	waist	0.51	weigh	0.52
holtenau	0.51	vienna	0.51	heel	0.51	backboard	0.50
frankfurt	0.50	nabucco	0.51	soles	0.50	metres	0.50
hamburg	0.49	palermo	0.50	shoulder	0.49	octaves	0.50
bonn	0.49	fossa	0.49	neck	0.49	centimetres	0.49
rheinische	0.48	teatro	0.48	shoulders	0.49	millimeter	0.49
A France:Paris::Germany:?		B France:Paris::Italy:?		C hand:fingers::foot:?		D hands:fingers::feet:?	

FIGURE 23.4 Nearest neighbors from the predicted vector in four analogy tasks. In the analogy *A is to B as C is to D*, a simple prediction for the vector for *D* is given by $C+B-A=-(A-B)+C$. The used embeddings are obtained by applying word2vec on the English Wikipedia corpus. This figure was created using data from the "Word2vec playground" (http://deeplearner.fz-qqq.net/) in June 1997 (site no longer available; a similar service is available at, e.g., https://rare-technologies.com/word2vec-tutorial/).

nicer	0.69	talk	0.74	talk	0.77
nice	0.67	talked	0.71	talking	0.76
happier	0.61	talking	0.63	talked	0.61
better	0.58	spoke	0.60	discussing	0.59
smarter	0.55	spoken	0.50	chatting	0.55
easier	0.53	discussing	0.48	telling	0.46
harder	0.52	approached	0.47	spoke	0.45
prettier	0.52	laughed	0.46	joking	0.45
bigger	0.51	speak	0.46	chatter	0.45
heck	0.50	chatting	0.44	spoken	0.44
A smart:smarter::nice:?		B walk:walked::talk:?		C walk:walking::talk:?	

good	0.61	good	0.85	headache	0.80
black	0.52	bad	0.62	headaches	0.74
great	0.48	great	0.62	nightmares	0.47
decent	0.47	decent	0.58	problems	0.42
bad	0.47	terrific	0.55	nightmare	0.41
terrific	0.44	tough	0.53	pain	0.38
poor	0.43	nice	0.52	anxiety	0.37
nice	0.43	terrible	0.52	distraction	0.35
terrible	0.43	excellent	0.52	symptoms	0.35
excellent	0.43	fantastic	0.51	cough	0.35
D green:black::good:?		E white:black::good:?		F message:messages::headache:?	

FIGURE 23.5 Nearest neighbors from the predicted vector in four analogy tasks. In the analogy *A is to B as C is to D*, a simple prediction for the vector for *D* is given by $C+B-A=-(A-B)+C$. Retrieved from Liu (2016)

and sentences as well. However, in practice, even in very large corpora, any particular sentence is extremely unlikely to occur multiple times. For words, phrases, or sentences occurring only once, we only have one chance to collect statistics on the items they co-occur with or that they can be predicted from, which is, in general, not enough to build useful semantic representations. Lack of data is therefore a serious problem for approaches that simply transfer the word-based techniques to multiword phrases and sentences.

The solution that formal semantics research found is to rely on the principle of compositionality, which says that "The meaning of a whole is a function of the meanings of the parts and of the way they are syntactically combined" (Partee, 1995). With the lambda calculus, composition turns out to be very elegant: lambda function application and reduction. For instance,

Lola:	*lola*
runs:	$\lambda x.run(x)$
Lola runs:	$(\lambda x.run(x))(lola) = run(lola)$

Many variants of formal semantics have been worked out, generally relying on a hierarchical, syntactic analysis of a sentence and an expressive logic to describe sentence meaning (including devices to represent temporal information and to represent thoughts or statements about thoughts or statements). In this approach, words get assigned potentially rather complex lambda expressions that are needed to compute meaning representations of sentences from those of words. For instance, a ditransitive verb *give* will get assigned an expression that regulates that the three arguments, giver, givee, and gift, will be labeled with the right variables, as well as with the information that the verb is in present tense. A word like *some* (in a sentence like *Some mammals swim*) will be assigned an expression that regulates that the sentence meaning will contain an existential quantifier (∃) and that the properties from its first argument (mammals) and from the second argument (swim) end up in the right place in that expression. On the other hand, the lexical meaning of content words typically remains unanalyzed, that is, it is assumed to be atomic. For instance, the word *mouse* just means MOUSE.

Vectorial lexical semantics, as reviewed in section 1, holds the promise of providing a more satisfactory account of the lexical semantics, but how to combine the strength of distributional and formal semantics is far from trivial. But since 2008 a number of promising approaches have been worked out to also compute useful vector representations for sentences. We will briefly review four approaches, of which the first three, like formal semantics take the *hierarchical* structure of a sentence as given. The fourth approach, in contrast, treats sentences as *sequences* of words.

2.1. Hierarchical Approaches We take as given that **x** and **y** are vectors representing two items x and y, respectively, and that **x** and **y** are to be combined according to the assumed hierarchical analysis of the sentence. The central task of vector-space models for phrases and sentences is then to find a composition function f so that the output is a vector representing constituent $p = xy$:

$$\mathbf{p} = f(\mathbf{x}, \mathbf{y}, R, K) \tag{3}$$

where R is the syntactic rule that combines x and y, and K is background knowledge (Mitchell & Lapata, 2008).

Computing a vector for a sentence is thus, like in formal semantics, carried out in a bottom-up manner on a given parse tree. An illustration is given in figure 23.6. Assuming that there are vectors representing words, we first compute a vector for the verb phrase:

$$\overrightarrow{like\ it} = f(\overrightarrow{like}, \vec{it}, VP \rightarrow ADJ\ PRP)$$

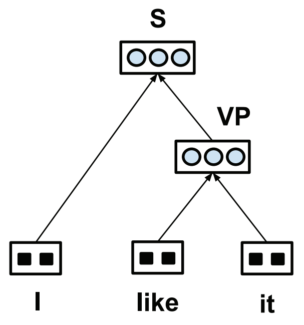

Figure 23.6 Compute sentence vector in a bottom-up manner. First, we compute a vector for the verb phrase (VP) and then compute a vector for the subject (S).

and then go higher up in the hierarchy to compute a vector for the whole sentence

$$\overrightarrow{I\ like\ it} = f(\vec{I}, \overrightarrow{like\ it}, S \rightarrow NP\ VP)$$

Approaches in this class mainly differ by the composition functions they make use of. Next, we will discuss three approaches.

2.1.1. Approach 1: Additive and multiplicative composition The simplest approach is to use vector element-wise operations such as addition and multiplication (Mitchell & Lapata, 2008)

- Additive: $\mathbf{p} = \mathbf{x} + \mathbf{y}$
- Multiplicative: $\mathbf{p} = \mathbf{x} \odot \mathbf{y}$ (i.e., $p_i = x_i \times y_i$)

where $p = xy$ is formed by combined x and y.

Despite its simplicity, this approach works surprisingly well in composing noun-noun and adjective-noun phrases (Blacoe & Lapata, 2012). However, from the theoretical point of view, this approach is unsatisfactory. Because those operations are commutative, word order and hierarchical structure of a sentence become irrelevant, that is,

$$\overrightarrow{man\ bit\ dog} = \overrightarrow{dog\ bit\ man}$$

A more sophisticated way is to linearly combine those two operations:

$$\mathbf{p} = \alpha\mathbf{x} + \beta\mathbf{y} + \gamma\mathbf{x}_i \odot \mathbf{y}$$

2.1.2. Approach 2: Tensor-product–based composition The idea is to resemble composition in formal semantics. In formal semantics, words such as adjectives and verbs are *functors* that are to modify *arguments* representing other words such as nouns, and composition is simply function application. Based on this, with the slogan "composition is largely a matter of *function application*" Baroni, Bernardi, and Zamparelli (2013) proposed using tensors[2] to represent words:

- first-order tensors (i.e., vectors $\in \mathbb{R}^n$) are to represent nouns,
- second-order tensors (i.e., matrices $\in \mathbb{R}^{n\times m}$) are to represent adjectives, intransitive verbs,
- third-order tensors (i.e., $\in \mathbb{R}^{n\times m\times k}$) are for transitive verbs, and so on.

Composition is carried out through tensor production. For example, composing an adjective and a noun is a matrix-vector multiplication like:

$$\overrightarrow{old\ dog} = f_{old}(\overrightarrow{dog}) = \mathbf{OLD} \times \overrightarrow{dog} \tag{4}$$

Training this model corresponds to finding proper tensors for words. Given the fact that approaches for word vectors are not applicable to second- and higher-order tensors, Baroni et al. (2013) and Grefenstette, Dinu, Zhang, Sadrzadeh, and Baroni (2013) employed linear regression with co-occurrence counts as training signals. First, co-occurrence count is used to find vectors for nouns like *dog*, vectors for noun phrases *old X* such as *old dog, old cat*. From those vectors representing *old X* and *X*, linear regression is used to estimate a matrix for *old* (see figure 23.7). Higher-order tensors are estimated with the same mechanism but with a technique called multistep linear regression.

From a theoretical point of view, the function-application–based approach is a beautiful model, which could be considered as inheriting the philosophy of formal semantics. However, learning tensors for lexical items is its most serious weakness. This is because the model requires a very large number of parameters (a single transitive verb needs n^3, i.e., $1M$ if $n=100$,

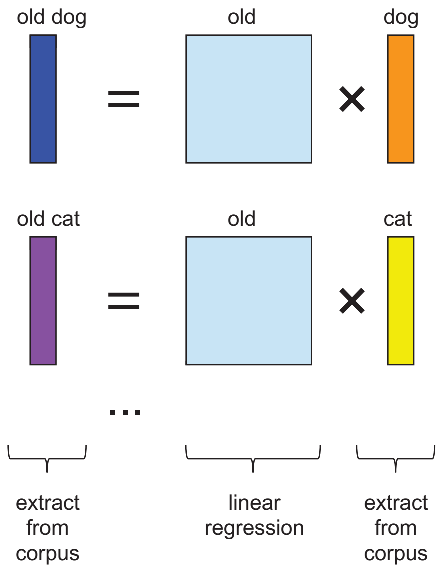

FIGURE 23.7 Learning a matrix for an adjective using linear regression.

parameters). Besides, we need to collect context distributions of phrases like *old dog* as in the preceding example. This becomes problematic when phrases are rare, and phrases tend to get very rare rapidly if they are composed of more than two words.

The most successful application of this idea, according to Baroni, is to composition in morphology (Marelli & Baroni, 2015). Marelli and Baroni found, for instance, that the composition of representations for *column* and *–ist* predicts that it's similar to *journalist* and thus correctly picks out the relevant sense (as *column* is similar to *pillar*).

2.1.3. Approach 3: Recursive neural networks Because two-layer feedforward neural networks are universal approximators (Cybenko, 1989), they can theoretically approximate *true* composition functions (if they exist and are measurable). Therefore, it is natural to think of using a feedforward neural net as a composition function. This leads to a model called Recursive neural network (RxNN). This idea was first proposed by Pollack (1990) and Goller and Küchler (1996). It then became popular in NLP thanks to the work series of Socher and colleagues (Socher, Manning, & Ng, 2010; Socher, Pennington, Huang, Ng, & Manning, 2011; Socher, Huval, Manning, & Ng, 2012).

A composition function for this model is a network consisting of two layers (figure 23.8A): an input layer with $2n$ nodes for pairs of two children and an output layer with n nodes for parents. Technically speaking, given a production $p \rightarrow x\ y$, the computation is given by

$$\mathbf{p} = a(\mathbf{W}_1\mathbf{x} + \mathbf{W}_2\mathbf{y} + \mathbf{b}) \tag{5}$$

where $\mathbf{x}$, $\mathbf{y}$, $\mathbf{p} \in \mathbb{R}^n$ are vectors representing x, y, p; $\mathbf{W}_1$, $\mathbf{W}_2 \in \mathbb{R}^{n\times n}$, $\mathbf{b} \in \mathbb{R}^n$ are weight matrices; and bias vector a is an activation function (e.g., *tanh*). The RxNN model differs from approaches 1 and 2 in that it is nonlinear (if the activation function a in equation 5 is nonlinear, e.g., *sigmoid*, *tanh*).

Although it is not clear whether the true composition functions are linear or nonlinear, the nonlinearity obviously makes the RxNN model more expressive and a wide range of choices for a make it more flexible. Although the RxNN model is simple, its elegance in terms of modeling and training makes it popular in the NLP community. A bundle of its extensions have been proposed, mainly along two directions: (i) proposing more expressive composition functions (e.g., Le & Zuidema, 2015a; Socher et al., 2012; Socher et al., 2013; Tai, Socher, & Manning, 2015) and (ii) extending the topology (Irsoy & Cardie, 2013, Le & Zuidema, 2014, 2015b).

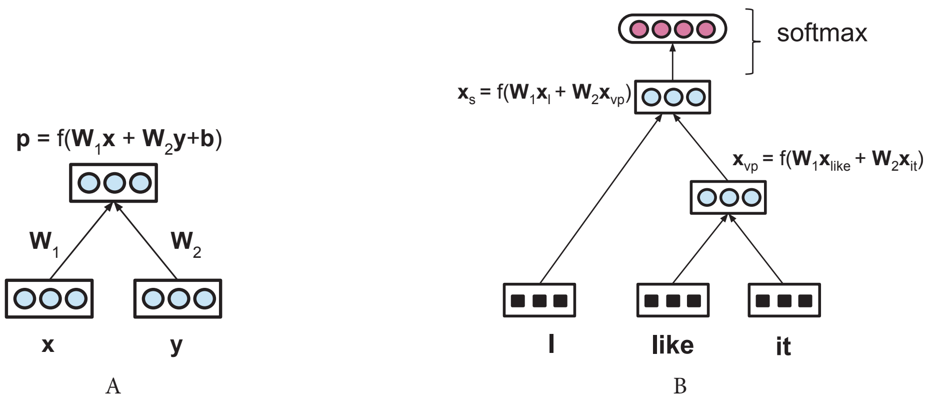

FIGURE 23.8 (A) A feedforward neural network as a composition function. (B) RxNN for sentence classification.

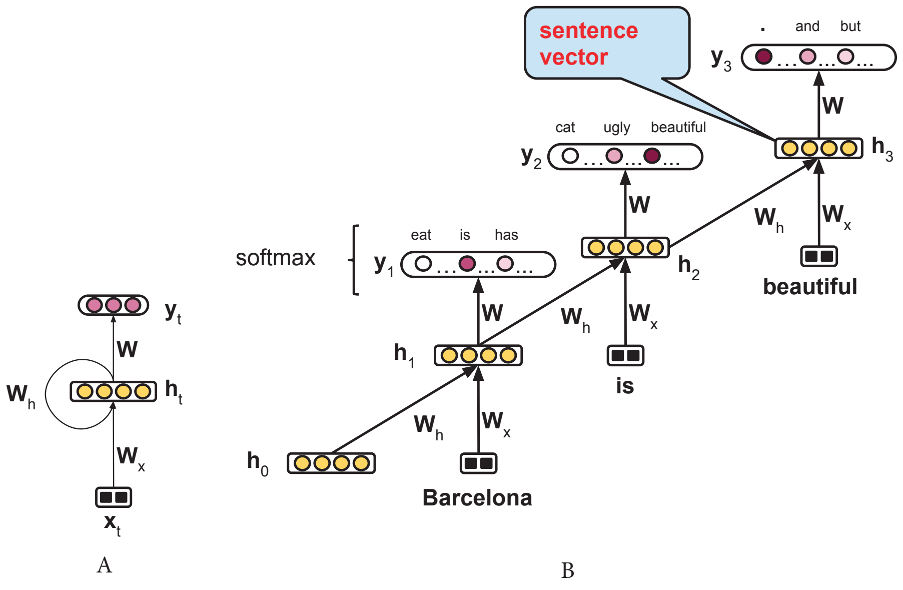

FIGURE 23.9 SRN (A) and the SRN when we unfold it (B). Notice that $\mathbf{h}_0$ can be simply $\vec{0}$.

2.2. Sequential Models of Sentence Semantics

2.2.1. Approach 4: Recurrent neural networks A fourth approach tries to compute vectors for sentences without (explicitly) using syntactic structures as glue to combine subcomponents. Successful versions of this approach are recurrent network and convolutional networks (which we will not discuss here, but see Collobert et al., 2011; Kalchbrenner, Grefenstette, & Blunsom, 2014; Kim, 2014).

A neural network is *recurrent* if it has at least one directed ring in its structure. In the simple recurrent neural network (SRN) model proposed by Elman (1990) (see figure 23.9A), an input $\mathbf{x}_t$ is fed to the network at each time t. The hidden layer **h**, which has activation $\mathbf{h}_{t-1}$ right before $\mathbf{x}_t$ comes in, plays a role as a memory capturing the whole history $(\mathbf{x}_0, \ldots, \mathbf{x}_{t-1})$. When $\mathbf{x}t$ comes in, the hidden layer updates its activation by:

$$\mathbf{h}_t = g(\mathbf{W}_x\mathbf{x}_t + \mathbf{W}_h \mathrm{h}_{t-1} + \mathbf{b})$$

where $\mathrm{W}_x \in \mathbb{R}^{n\times dim(\mathrm{x}_t)}$, $\mathrm{W}_h \in \mathbb{R}^{n\times n}$, $\mathrm{b} \in \mathbb{R}^n$ are weight matrices and bias vector g is an activation function. This network model thus, in theory, can be used to estimate conditional probabilities with long histories.

Because the SRN can handle sequences with different lengths, we can use it to process sentences. For instance, figure 23.9B shows how we can use the SRN model for language modeling: the idea is to try to predict which word will come next. In this way, when we reach the end of a sentence, the hidden layer activation captures the whole sentence. In other words, we can use the activation of the hidden layer at that time point to represent the whole sentence. Figure 23.9B also shows that the SRN model is in fact a special case of the RxNN model: the input tree is always left-branching.

The network is efficiently trained using the back-propagation through time algorithm (Werbos, 1990). Although in theory recurrent neural networks are, at least, as expressiveness as Turing machines (Hyötyniemi, 1996), and training them is difficult because of the vanishing gradient problem and the problem of how to capture long-range dependencies (Hochreiter, Bengio, Frasconi, & Schmidhuber, 2001). Two alternatives, which have become popular recently, are the *Long short-term memory architecture* (Hochreiter & Schmidhuber, 1997) and *gated recurrent neural networks* (Chung, Gulcehre, Cho, & Bengio, 2014).

3. Conclusion

How can we represent the meaning of words and sentences in a computational model? In this chapter we have discussed foundational work in distributional semantics, where word meanings are represented with (high-dimensional) numerical vectors, and recent extensions of this approach that generalize the vector representations to also apply to sentences. We have briefly contrasted these approaches to alternative modeling traditions. We have shown that vectorial models are surprisingly versatile and account for a wide variety of semantic phenomena often claimed to necessitate symbolic formalisms, while offering the additional benefits of being much more compatible with both neurobiological modeling and existing machine learning frameworks.

NOTES

1. See, for instance, http://esslli2016.unibz.it/?page_id =242.
2. Tensors describe linear relations among vectors, scalars, and other tensors. An n-dim vector is a first-order n-shape tensor, and an $n \times m$ matrix is a second-order $n \times m$-shape tensor. A third-order $(n \times m) \times k$–shape tensor **T** describes a linear map from a k-dim vector **v** to an $n \times m$ matrix **M**

$$\mathbf{M} = \mathbf{T} \times \mathbf{v}$$

where

$$M_{m,n} = \textstyle\sum_k T_{m,n,k^{v}k}$$

Generally, a $(I_1 \times \ldots \times I_m) \times (J_1 \times \ldots \times J_n)$–shape tensor **T** describes a linear map from a $J_1 \times \ldots \times J_n$-shape tensor **V** to an $I_1 \times \ldots \times I_m$-shape tensor **M** by

$$M_{i_1,\ldots,i_m} = \sum_{j_1,\ldots,j_n} T_{i_1,\ldots,i_m,j_1,\ldots,j_n} V_{j_1,\ldots,j_n}$$

REFERENCES

Baroni, M. (2016). Grounding distributional semantics in the visual world. *Language and Linguistics Compass, 10*(1), 3–13.

Baroni, M., Bernardi, R., & Zamparelli, R. (2013). Frege in space: A program for compositional distributional semantics. *LiLT (Linguistic Issues in Language Technology), 9*, 241–346.

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

Bengio, Y., Ducharme, R., Vincent, P., & Janvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, *3*, 1137–1155.

Blacoe, W., & Lapata, M. (2012). A comparison of vector-based representations for semantic composition. In J. Tsujii, J. Henderson, & M. Paşca (Eds.), *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning* (pp. 546–556). Stroudsburg, PA: Association for Computational Linguistics.

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *ArXiv Preprint*. Advanced online publication. arXiv:1412.3555.

Collobert, R., Weston, J., Bottou, L., Karlen, M, Kavukcuoglu, K., & Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research, 12*, 2493–2537.

Cotterell, R., Schütze, H., & Eisner, J. (2016). Morphological smoothing and extrapolation of word embeddings. In K. Erk & N. A. Smith (Eds.), *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 1651–1660). Stroudsburg, PA: Association for Computational Linguistics.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, *2*(4), 303–314.

Dinu, G., & Lapata, M. (2010). Measuring distributional similarity in context. In H. Li & L. Màrquez (Eds.), *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing* (pp. 1162–1172). Stroudsburg, PA: Association for Computational Linguistics.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science, 14*(2), 179–211.

Erk, K. (2012). Vector space models of word meaning and phrase meaning: A survey. *Language and Linguistics Compass, 6*(10), 635–653.

Firth, J. R. (1957). A synopsis of linguistic theory, 1930–1955. In *Studies in Linguistic Analysis (special volume of the Philological Society), Vol. 1952–59* (pp. 1–32). Oxford: The Philological Society.

Fremann, L., & Lapata, M. (2016). A Bayesian model of diachronic meaning change. *Transactions of the Association of Computational Linguistics*, *4*, 31–45.

Goller, C., & Küchler, A. (1996). Learning task-dependent distributed representations by backpropagation through structure. In *IEEE International Conference on Neural Networks* (pp. 347–352). New York: IEEE Press.

Grefenstette, E., Dinu, G., Zhang, Y.-Z., Sadrzadeh, M., & Baroni, M. (2013). Multi-step regression learning for compositional distributional semantics. In K. Erk & A. Koller (Eds.), *Proceedings of the 10th International Conference on*

Computational Semantics (IWCS). Stroudsburg, PA: Association for Computational Linguistics.

Hamilton, W. L., Leskovec, J., & Jurafsky, D. (2016). Diachronic word embeddings reveal statistical laws of semantic change. In K. Erk & N. A. Smith (Eds.), *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 1489–1501). Stroudsburg, PA: Association for Computational Linguistics.

Henderson, J., & Popa, D. N. (2016). A vector space for distributional semantics for entailment. *ArXiv Preprint*. Advanced online publication. arXiv:1607.03780.

Hochreiter, S., Bengio, Y., Frasconi, P., & Schmidhuber, J. (2001). Gradient flow in recurrent nets: The difficulty of learning long-term dependencies. In J. F. Kolen & S. C. Kolen (Eds.), *A field guide to dynamical recurrent neural networks* (pp. 237–244). New York: IEEE Press.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Hyötyniemi, H. (1996). Turing machines are recurrent neural networks. In J. Alander, T. Honkela, and M. Jakobsson (Eds.), *Proceedings of STeP'96, Finnish Artificial Intelligence Conference*. Vaasa, Finland: Finnish Artificial Intelligence Society.

Irsoy, O., & Cardie, C. (2013). Bidirectional recursive neural networks for token-level labeling with structure. *ArXiv Preprint*. Advanced online publication. arXiv:1312.0493.

Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014). A convolutional neural network for modelling sentences. In K. Toutanova & H. Wu (Eds.), *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 655–665), Stroudsburg, PA: Association for Computational Linguistics.

Kim, Y. (2014). Convolutional neural networks for sentence classification. In A. Moschitti, B. Pang, & W. Daelemans (Eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1746–1751). Stroudsburg, PA: Association for Computational Linguistics.

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

Le, P., & Zuidema, W. (2014). Inside-outside semantics: A framework for neural models of semantic composition. In *Proceedings of the NIPS 2014 Workshop on Deep Learning and Representation Learning*. Montreal, Canada.

Le, P. & Zuidema, W. (2015a). Compositional distributional semantics with long short term memory. In C. Gardent, R. Bernardi, & I. Titov (Eds.), *Proceedings of the Joint Conference on Lexical and Computational Semantics (*SEM)*. Stroudsburg, PA: Association for Computational Linguistics.

Le, P., & Zuidema, W. (2015b). The forest convolutional network: Compositional distributional semantics with a neural chart and without binarization. In L. Màrquez, C. Callison-Burch, & J. Su (Eds.), *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (pp. 1155–1164). Stroudsburg, PA: Association for Computational Linguistics.

Levy, O., & Goldberg, Y. (2014). Neural word embedding as implicit matrix factorization. *Advances in Neural Information Processing Systems*, *27*, 2177–2185.

Liu, A. (2016). Word to Vec JS demo. Retrieved from http://turbomaze.github.io/word2vecjson/.

Lowe, W. (2001). Towards a theory of semantic space. In J. D. Moore & K. Stenning (Eds.), *Proceedings of the 23rd Annual Meeting of the Cognitive Science Society*. (pp. 576–581). Mahwah, NJ: Lawrence Erlbaum Associates.

Marelli, M., & Baroni, M. (2015). Affixation in semantic space: Modeling morpheme meanings with compositional distributional semantics. *Psychological Review*, *122*(3), 485–515.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *ArXiv Preprint*. Advanced online publication. arXiv: 1301.3781.

Mikolov, T., Karafiát, M., Burget, L., Černocký, J, & Khudanpur, S. (2010). Recurrent neural network based language model. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)* (pp. 1045–1048). Curran Associates, Inc.: Red Hook, NY.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, *26*, 3111–3119.

Mikolov, T., Yih, W., & Zweig, G. (2013). Linguistic regularities in continuous space word representations. In L. Vanderwende, H. Daumé III, & K. Kirchhoff (Eds.), *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 746–751). Stroudsburg, PA: Association for Computational Linguistics.

Mitchell, J., & Lapata, M. (2008). Vector-based models of semantic composition. In *Proceedings of 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies* (pp. 236–244). Stroudsburg, PA: Association for Computational Linguistics.

Mitchell, T. M., Shinkareva, S. V., Carlson, A., Chang, K.-M., Malave, V. L., Mason, R. A., & Just, M. A. (2008). Predicting human brain activity associated with the meanings of nouns. *Science*, *320*(5880), 1191–1195.

Murphy, B., Talukdar, P., & Mitchell, T. (2012). Selecting corpus-semantic models for neurolinguistic decoding. In E. Agirre, J. Bos, M. Iab, S. Manandhar, Y. Marton, & D. Yuret (Eds.),**SEM 2012: The First Joint Conference on Lexical and Computational Semantics—Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)* (pp. 114–123). Stroudsburg. PA: Association for Computational Linguistics.

Partee, B. (1995). Lexical semantics and compositionality. In L. R. Gleitman & M. Liberman (Eds.), *An invitation to cognitive science. Vol. 1: Language* (pp. 311–360). Cambridge, MA: MIT Press.

Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global vectors for word representation. In A. Moschitti, B. Pang, & W. Daelemans (Eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1532–1543). Stroudsburg, PA: Association for Computational Linguistics.

Pereira, F., Gershman, S., Ritter, S., & Botvinick, M. (2016). A comparative evaluation of off-the-shelf distributed semantic representations for modelling behavioural data. *Cognitive Neuropsychology*, *33*(3–4), 175–190.

Pollack, J. B. (1990). Recursive distributed representations. *Artificial Intelligence*, *46*(1), 77–105.

—-1
—0
—+1

Socher, R., Huval, B., Manning, C. D., & Ng, A. Y. (2012). Semantic compositionality through recursive matrix-vector spaces. In J. Tsujii, J. Henderson & M. Pasca (Eds.), *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning* (pp. 1201–1211). Stroudsburg, PA: Association for Computational Linguistics.

Socher, R., Manning, C. D., & Ng, A. Y. (2010). Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS 2014 Workshop on Deep Learning and Representation Learning*. Montreal, Canada.

Socher, R., Pennington, J., Huang, E. H., Ng, A. Y., & Manning, C. D. (2011). Semi-supervised recursive autoencoders for predicting sentiment distributions. In R. Barzilay & M. Johnson (Eds.), *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (pp. 151–161). Stroudsburg, PA: Association for Computational Linguistics.

Socher, R., Perelygin, A., Wu, J. Y., Chuang, J., Manning, C. D., Ng, A. Y., & Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In D. Yarowsky, T. Baldwin, A. Korhonen, K. Livescu, & S. Bethard (Eds.), *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1631–1642). Stroudsburg, PA: Association for Computational Linguistics.

Tai, K. S., Socher, R., & Manning. C. D. (2015). Improved semantic representations from tree-structured long short-term memory networks. In C. Zong & M. Strube (Eds.), *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (pp. 1556–1566). Stroudsburg, PA: Association for Computational Linguistics.

Werbos, P. (1990). Backpropagation through time: What it does and how to do it. *Proceedings of the IEEE*, *78*(10), 1550–1560.