9

Distributional Semantics beyond the Lexicon

The Distributional Hypothesis is mainly a conjecture about the representation and learning of lexical meaning from co-occurrence statistics (cf. Chapter 1.1). However, distributional semantics cannot be claimed to contribute to a general theory of meaning, unless it provides a satisfactory model of the content of complex expressions such as phrases and sentences. The simplest solution might consist in the **holistic approach** (Turney, 2012): Representing complex expressions as single units to which standard DSMs assign a distributional vector, just like with any other lexical item. This way, phrases such as *red car* or *The dog chases the cat* would be treated as kinds of "words with spaces" and added to the set of DSM targets. However, the holistic solution is not satisfactory (with the possible exception of some idiomatic expressions),[1] because it does not account for a key property of natural language: **productivity**.

HOLISTIC REPRESENTATIONS

PRODUCTIVITY

Productivity is the ability to produce and interpret a potentially infinite number of novel linguistic expressions.

Natural language contains grammatical and lexical processes that are productive because they can be applied to open-ended classes of expressions that satisfy certain properties. For instance, morphological affixes allow us to create an endless number of new lexical items: If we know that *gorp* is a verb, we can infer that *gorper* refers to the agent of gorping, whatever the meaning of *gorp*. Similarly, it is the productivity of natural language that allows us to use its expressions to communicate novel situations, like in the following sentence:

(1) A dog wearing sun glasses is surfing in the ocean.

[1] Actually, even idioms cannot always be treated holistically from the semantic point of view (Nunberg et al., 1994). Idioms are a very heterogeneous class of linguistic expressions that differ in their degree of structural variability. Despite their idiosyncratic meaning, they cannot be regarded as just "words with spaces" (Cacciari, 2014).

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

Productivity is constitutive of the "magic of language" (Pelletier, 2006) and is usually considered one of the main arguments that semantic competence includes some general and systematic procedure to build the interpretation of an expression by combining the meaning of its components. Such central feature of natural language semantics is called **compositionality**.

COMPOSITIONALITY

> The meaning of a linguistic expression is **compositional** if it can be obtained by combining the meaning of its parts.

The notion of compositionality has been the subject of an intense debate in philosophy and cognitive science (Werning et al., 2012). Though not all aspects of natural language are compositional (e.g., idiomatic expressions), it is a fact that the meanings of complex expressions can be derived from the meanings of their parts (Dowty, 2007; Martin and Baggio, 2019). This explains our ability to understand (1), even if we have never seen it before. The key question is to identify and model the mechanisms that are used to compose meanings. Here, we focus on a more specific aspect of this general issue: *the computational processes to compose distributional representations*.

In this chapter, we present current research in **compositional distributional semantics**, which aims at designing methods to construct the interpretation of complex linguistic expressions from the distributional representations of the lexical items they contain. This theme includes two major questions that we are going to explore in the following sections: (i) *what is the distributional representation of a phrase or sentence and to what extent it is able to encode key aspects of its meaning?* (ii) *how can we build such representations compositionally?* Addressing these issues is crucial to prove the ability of distributional semantics to provide a general model of natural language interpretation, but also for NLP and AI applications that need distributional representations for phrases and sentences, to be used in tasks such as textual semantic similarity, paraphrase detection, machine translation, natural language inference, question answering, among many others.

COMPOSITIONAL DISTRIBUTIONAL SEMANTICS

The mainstream approach in compositional distributional semantics consists in representing phrase and sentence meanings with vectors:

PHRASE AND SENTENCE VECTORS

> The meaning of a phrase (sentence) is a **phrase (sentence) vector**.

After introducing the classical symbolic paradigm of compositionality based on function-argument structures and function application (Section 9.1), we review different methods to create phrase and sentence vectors – simple vector operations (Section 9.2), linear-algebraic models of function application

(Section 9.3), neural networks trained to learn sentence embeddings (Section 9.4) – and the main tasks and datasets to evaluate compositional DSMs (Section 9.5). Then, we investigate how distributional semantics addresses two issues that are strictly related to compositionality: the context-sensitive nature of semantic representations, with a particular focus on the last generation of **contextual embeddings** (Section 9.6), and the semantic constraints governing the combination of lexical items, also known as selectional preferences (Section 9.7). We end with some general considerations about compositionality, semantic structures, and vector models of meaning (Section 9.8).

CONTEXTUAL EMBEDDINGS

9.1 Semantic Representations and Compositionality

The philosopher and logician **GOTTLOB FREGE** (1848–1925) claims that the productivity of natural language would be impossible

> were we not able to distinguish parts in the thought corresponding to the parts of a sentence, so that the structure of the sentence serves as an image of the structure of the thought. [. . .] If, then, we look upon thoughts as composed of simple parts, and take these, in turn, to correspond to the simple parts of sentences, we can understand how a few parts of sentences can go to make up a great multitude of sentences, to which, in turn, there correspond a great multitude of thoughts. (Frege, 1923, p. 1)

PRINCIPLE OF COMPOSITIONALITY

In logic and linguistics, this idea has been expressed as the **Principle of Compositionality**, usually stated in the following way (Partee, 1984):

> The meaning of a linguistic expression is a function of the meaning of its lexical items and of the way they are syntactically combined.

This is also named **Frege's Principle** or **Fregean Compositionality**.[2] According to the Principle of Compositionality, the interpretation of linguistic expressions is built with a recursive process that combines the meaning of lexical items through semantic operations associated with syntactic structures.

FUNCTIONS AND ARGUMENTS

PREDICATES

In symbolic semantic models, compositional meaning construction is formalized with **function-argument structures**, which represent the fact that it is possible to analyze meaning into parts that can then be recombined one independently of the other to produce new meanings. Linguistic expressions are divided into **predicates** and **arguments**. Predicates express properties, events, and relations involving one or more arguments. The number of arguments

[2] Whether this principle was explicitly endorsed by Frege is still a matter of dispute (Werning et al., 2012; Pelletier, 2016), but it undoubtedly resonates with his words quoted above.

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

required by a predicate is also called its **valence** or **arity**. In (2), *run* is the predicate and *Tom* is the argument:

VALENCE

(2) Tom runs.

The operation of applying a predicate to an argument is called **predication**. The meaning of a predicate is treated as an "unsaturated" entity (Frege, 1891) and is formalized as a **function** with **variables**, each representing an open slot to be filled by an argument. In fact, "every theory of semantics back to Frege acknowledges that word meanings may contain variables that are satisfied by arguments expressed elsewhere in the sentence" (Jackendoff, 2002, p. 360). Using the **lambda notation**, the meaning of the unary predicate *run* is represented by the following function symbol:

PREDICATION

FUNCTIONS WITH VARIABLES

LAMBDA NOTATION

(3) $\lambda \mathrm{x}[\mathrm{run}(\mathrm{x})]$

The "λ" marks the variable "x" as the unsaturated element of the function "run." A function is a **mapping** between two sets, the **domain** and the **range** of the function. The function $\lambda \mathrm{x}[\mathrm{f}(\mathrm{x})] : A \rightarrow B$ has domain A and range B:

FUNCTION DOMAIN AND RANGE

> The **function** $\lambda \mathrm{x}[\mathrm{f}(\mathrm{x})] : A \rightarrow B$ maps an argument $a \in A$ to one and only one element $b \in B$ called the **value** for the function given a.

Predication is formally modeled as **function application**, the process of saturating (i.e., filling) the variable of a function with an argument:

PREDICATION AS FUNCTION APPLICATION

> The **application** of the function $\lambda \mathrm{x}[\mathrm{f}(\mathrm{x})]$ to an argument a, $\lambda \mathrm{x}[\mathrm{f}(\mathrm{x})](\mathrm{a})$, turns the function into the saturated expression $\mathrm{f}(\mathrm{a})$, obtained by replacing the variable with a; $\mathrm{f}(\mathrm{a})$ is the value of the function $\lambda \mathrm{x}[\mathrm{f}(\mathrm{x})]$ applied to a.

The semantic representation of (2) is built by applying (3) to "Tom":

(4) $\lambda \mathrm{x}[\mathrm{run}(\mathrm{x})](\mathrm{Tom}) \Rightarrow \mathrm{run}(\mathrm{Tom})$

Function application is the general operation to combine the meaning of lexical items into the interpretation of complex linguistic expressions.

In **formal semantics**, the meaning of a sentence is its **truth conditions**, "something that determines the conditions under which the sentence is true or false" (Lewis, 1970, p. 22). In this framework, predicates correspond to functions that are true or false of their arguments, and function application is the operation to build the truth conditions of sentences. One of the most influential

TRUTH CONDITIONS

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

MONTAGUE SEMANTICS
LOGICAL FORMS
MODEL
DOMAIN
INTERPRETATION FUNCTION
EXTENSION

formal models of meaning based on Fregean Compositionality is **Montague Semantics** (Montague, 1974). **RICHARD MONTAGUE** (1930–1971) aims at describing natural language interpretation with a fully compositional truth-conditional semantics (Lenci and Sandu, 2009). Natural language items are translated into expressions of an unambiguous logical language called **logical forms**, which are then interpreted on a **model** $\langle D, I \rangle$ (hence the name of **model-theoretic semantics**). A model consists of a **domain** D, the set of all individuals in the world, and the **interpretation function** I, which assigns to each logical form its **extension** (i.e., its reference in the world), formally expressed with $[\![\ ]\!]$.[3] Lexical items are mapped onto expressions of the formal language, which are then combined with function application to derive the logical forms that represent the truth conditions of sentences.

SEMANTIC TYPE

In Montague Semantics, the expressions of the logical language are assigned a **semantic type** τ that individuates the domain of their interpretation, D_τ. Types are recursively defined in the following way:

(i) e is the type of **individual entities**: $D_e = D$;
(ii) t is the type of **truth values**: $D_t = \{0, 1\}$;
(iii) if a and b are types, then the type $\langle a, b \rangle$ is the type of **functions** whose arguments are of type a and whose values are of type b: $D_{\langle a,b \rangle}$ is the set of functions $f : D_a \rightarrow D_b$.

CHARACTERISTIC FUNCTION

Individual entities and truth values are saturated basic types, while functions are unsaturated types. Each syntactic category is mapped onto a logical form of suitable semantic type. As illustrated in Table 9.1, a proper noun like *Tom* is associated with the logical expression "$\mathrm{Tom} : e$" (i.e., a constant; the column introduces the type) that denotes the individual Tom. Common nouns and intransitive verbs are translated into one-place predicates of type $\langle e, t \rangle$ that denote functions from individuals to truth values. For instance, the extension of the logical form of the verb *run*, "$\lambda \mathrm{x}[\mathrm{run}(\mathrm{x})]$," is the function that for each individual produces the value 1 if it runs and 0 otherwise. A function f of type $\langle e, t \rangle$ is equivalent to the set A of individuals for which f gives the value 1, and is called the **characteristic function** of A. Therefore, formal expressions of type $\langle e, t \rangle$ denote sets of individuals (e.g., $[\![\lambda \mathrm{x}[\mathrm{run}(\mathrm{x})]]\!]$ is the set of individuals that run). Transitive verbs are mapped into two-place predicates of type $\langle e, \langle e, t \rangle \rangle$ that denote functions from individuals to functions from individuals to truth values. These are characteristic functions of sets of pairs of individuals,

[3] Actually, Montague Semantics uses an intensional logic: I assigns to logical expressions their **intensions**, defined as functions from possible worlds to extensions (Lewis, 1970; Montague, 1974). For the sake of simplicity, here we ignore intensional aspects.

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

Table 9.1 Categories of linguistic expressions associated with their logical form, semantic type, and extensional interpretation in Montague Semantics

Syntactic category	*Example*	*Type*	*Logical form*	*Interpretation*
proper noun	*Tom*	e	Tom	individual
sentence	*Tom runs*	t	$\mathrm{run(Tom)}$	truth value
common noun	*dog*	$\langle e, t\rangle$	$\lambda\mathrm{x}[\mathrm{dog(x)}]$	set
intransitive verb	*run*	$\langle e, t\rangle$	$\lambda\mathrm{x}[\mathrm{run(x)}]$	set
transitive verb	*chase*	$\langle e, \langle e, t\rangle\rangle$	$\lambda\mathrm{y}\lambda\mathrm{x}[\mathrm{chase(x, y)}]$	binary relation

which are equivalent to binary relations (e.g., $[\![\lambda\mathrm{y}\lambda\mathrm{x}[\mathrm{chase(x, y)}]]\!]$ is the relation defined by the set of pairs of individuals a and b such that a chases b). More in general, predicates with n arguments denote n-place relations.

The operation of meaning composition is **typed function application**, which combines functions and arguments of suitable type:

TYPED FUNCTION APPLICATION

If α is of type a and β is of type $\langle a, b\rangle$, then $\beta(\alpha)$ is of type b.

Typed function application and syntactic rules work *in parallel* to derive the logical form of sentences compositionally. This strict syntax-semantics correspondence is called **rule-by-rule interpretation**:

RULE-BY-RULE INTERPRETATION

Let $[_{\mathrm{X}}\mathrm{Y}\ \mathrm{Z}]$ be a syntactic node, α and β the logical forms of Y and Z.

$$\text{The logical form of X is} \begin{cases} \beta(\alpha) : b & \text{if } \alpha : a \text{ and } \beta : \langle a, b\rangle \\ \alpha(\beta) : b & \text{if } \alpha : \langle a, b\rangle \text{ and } \beta : a \end{cases}$$

Figure 9.1 shows the compositional derivation of the interpretation of *Tom chases Jerry*. Lexical items are mapped onto typed logical forms. The meaning of the VP is obtained by applying the function of type $\langle e, \langle e, t\rangle\rangle$ expressed by the verb to the representation of the object NP of type e. The result of function application is the logical form "$\lambda\mathrm{x}[\mathrm{chase(x, Jerry)}] : \langle e, t\rangle$" that denotes the set of individuals that chase Jerry. This function is then applied to the interpretation of the subject NP to obtain the logical form "$\mathrm{chase(Tom, Jerry)} : t$" that is true if and only if Tom chases Jerry.

Typed functions also represent the constraints that determine the appropriate arguments of a predicate. Traditionally, these constraints are called **selectional restrictions** (Katz and Fodor, 1963; Chomsky, 1965; Asher, 2011, 2015; Pustejovsky and Batiukova, 2019). The sentences in (5) are well-formed from the syntactic point of view, but (5b) is semantically anomalous, because the combinations of lexemes do not satisfy their selectional constraints:

SELECTIONAL RESTRICTIONS

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

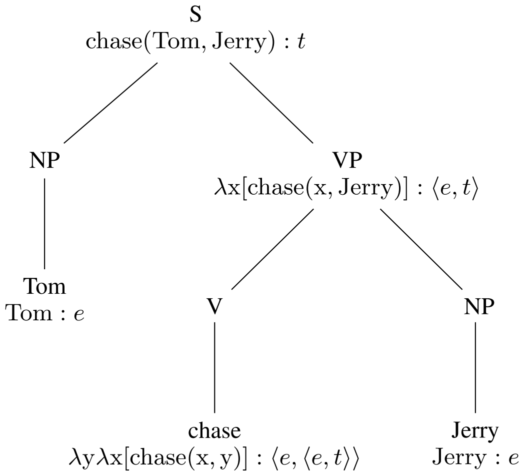

S
chase(Tom, Jerry) : t
NP
VP
λx[chase(x, Jerry)] : $\langle e, t\rangle$
Tom
Tom : e
V
NP
chase
λyλx[chase(x, y)] : $\langle e, \langle e, t\rangle\rangle$
Jerry
Jerry : e

Figure 9.1 The compositional derivation of the meaning of the sentence *Tom chases Jerry* in Montague Semantics

(5) a. The tall man drinks beer.
b. * The red idea drinks beer.

The predicate *drink* requires the subject argument to be animate, and the adjective *red* must modify concrete entities, while *idea* is an abstract noun. Another famous example of selectional restriction violation is the sentence *Colorless green ideas sleep furiously* in Chomsky (1957).

Selectional restrictions are modeled by typing the variables of predicates with symbols that identify semantic categories or features, such as ANIMATE, LOCATION, LIQUID, and so on. These can be regarded as subtypes of the Montagovian type e of individual entities (Asher, 2011). For example, the selectional restrictions of the verb *drink* are represented in the following way:

(6) λy:LIQUID λx:ANIMATE [drink(x, y)]

The verb *drink* must combine with direct objects denoting individual entities of type LIQUID and with subjects denoting individual entities of type ANIMATE.

FRAME SEMANTICS

SEMANTIC FRAMES

CONCEPTUAL SEMANTICS

The function-argument distinction is the cornerstone to build phrasal meaning also outside formal semantics. In **Frame Semantics** (Fillmore, 1982), the meaning of a sentence is not its truth conditions, but the conceptual representation of a situation or event. The **semantic frames** representing predicates (cf. Section 8.1) can be regarded as functions to be saturated by the event participants. In **Conceptual Semantics** (Jackendoff, 1990, 2002), the grammatically relevant aspects of meaning are encoded in a symbolic level of

CONCEPTUAL STRUCTURE

mental representation called **Conceptual Structure** (CS). As illustrated in Section 8.1, Conceptual Semantics adopts a decompositional approach to lexical meaning. The ingredients of CS are a small set of conceptual categories that define a universal repertoire of semantic types (e.g., EVENT, PATH, PLACE, OBJECT, etc.) and a set of primitive conceptual functions (e.g., GO, TO, IN, etc.) that are used to describe the interpretation of predicates. Lexical items are represented with fragments of CS that are combined mainly through typed function application proceeding parallel to syntactic structure:

(7) a. $[_{\text{S}}$ $[_{\text{NP}}$ The dog$]_1$ $[_{\text{VP}}$ entered$_2$ $[_{\text{NP}}$ the room$]_3]]$

b. $[_{\text{EVENT}}$ GO $([_{\text{OBJECT}}$ DOG$]_1$, $[_{\text{PATH}}$ TO $([_{\text{PLACE}}$ IN $([_{\text{OBJECT}}$ ROOM$]_3)])])]_2$

The indexes in (7) mark the links between syntactic and semantic constituents. The verb *enter* is associated with the following typed function in CS, which maps two arguments x and y of type OBJECT into an event of x going into y:

(8) $[_{\text{EVENT}}$ GO $([_{\text{OBJECT}}$ x], $[_{\text{PATH}}$ TO $([_{\text{PLACE}}$ IN $([_{\text{OBJECT}}$ y])])])]

The interpretation of the subject is the CS $[_{\text{OBJECT}}$ DOG] that saturates the first argument of the function, while the second argument is saturated by the CS $[_{\text{OBJECT}}$ ROOM] representing the meaning of the direct object.

9.1.1 The Problems of Fregean Compositionality

Fregean Compositionality claims that the meaning of complex expressions is a function of the meaning of its lexical items. Adopting this principle and its formalization with typed function-argument structures entails assuming that semantic composition in natural language obeys to the following properties:

1. lexical items are assigned their interpretation and fully disambiguated before being composed;
2. the meaning of lexemes is not affected by their composition, as the operation of application does not modify functions and arguments;
3. the semantic interpretation of linguistic expressions is strictly separated from pragmatic and contextual information;
4. if a type mismatch occurs between a predicate and its argument, their combination is semantically anomalous;
5. all elements of phrasal meaning are to be found in the meanings of their component lexical items.

Semantic research has shown that these assumptions raise several problems and are seriously challenged by empirical evidence (Pustejovsky, 1995; Jackendoff, 1997; Asher, 2011; Pustejovsky and Batiukova, 2019).

VAGUENESS

Traditionally, lexical meaning has been regarded as context-free and autonomous from the compositional process. Fodor and Pylyshyn (1988) claim that the principle of compositionality requires that a lexical item must "make approximately the same contribution to each expression in which it occurs" (p. 42). Representing a predicate as a function with unsaturated variables indeed entails assuming that its meaning is virtually independent from the arguments it can apply to (Martin and Baggio, 2019). However, this alleged context-independence of predicates from their arguments is often proved wrong. For instance, **vagueness** is a pervasive phenomenon in language (cf. Section 8.2): Lexemes have often vague meanings that are "modulated" and specified by their syntagmatic combinations. The adjective *red* corresponds to distinct chromatic varieties in the phrases *red hair*, *red wine*, *red face*, and *red brick*. Analogously, the subjects in (9) and the direct objects in (10) produce different, albeit related, events of running and opening (Kintsch, 2001):

(9) a. The man runs.
b. The ship runs before the wind.

(10) a. Mary opened the letter from her mother.
b. The rangers opened the trail for the season.
c. John opened the door for the guests.

CONTEXT-SENSITIVE MEANING

CO-COMPOSITIONALITY

These examples support the hypothesis that lexical meaning is highly **context-sensitive** (Yee and Thompson-Schill, 2016) and that predication should not be modeled just with the standard function application, since the predicate itself can be modified by the argument, a phenomenon called **co-compositionality** (Pustejovsky, 1995, 2012).

METAPHOR
METONYMY

Fregean Compositionality is also directly challenged by the widespread processes of **metaphor** and **metonymy**:

(11) a. My car drinks gasoline.
b. Plato is on the second shelf.
c. The student finished the book.
d. The student finished the cigarette.
e. fast car; fast guitarist; fast lane

As Wilks (1978) points out, the selectional restrictions that the subjects of *drink* must be animate are not satisfied in (11a), but the sentence is acceptable because the verb has a metaphoric interpretation. In (11b), *Plato* is used metonymically to refer to the philosopher's books. The sentences (11c) and (11d) exemplify a phenomenon known as **logical metonymy**, in which a type clash between an event-selecting verb and an entity-denoting object triggers the recovery of an implicit event depending on the argument noun: (11c) means that the student finished *reading* the book and (11d) that the student finished *smoking* the cigarette. Similarly, the interpretation of the phrases in (11e) includes an implicit event related to the noun head: *fast plane* means a plane that flies quickly, *fast guitarist* a guitarist that plays the guitar fast, and *fast lane* a lane in which one can drive at high speed. These examples suggest that semantic constraints should be regarded as **selectional preferences** (Wilks, 1978) rather than as hard type restrictions, since their failure does not always lead to semantic anomaly. Moreover, the process of semantic composition must include mechanisms of **coercion** to adjust the meaning of predicates or arguments and overcome selectional preference violations, and to add information that is not overtly expressed in the linguistic input and depends on contextual knowledge (Pustejovsky, 1995; Asher, 2011; Lauwers and Willems, 2011).

LOGICAL METONYMY

SELECTIONAL PREFERENCES

COERCION

It is of course possible to argue that the alleged problems of Fregean Compositionality are instead just cases of lexical ambiguity. According to this view, which Pustejovsky (1995) refers to as the **sense-enumerative** explanation, the different interpretations in (9) and (10) would correspond to distinct verb senses. Once the proper word meaning is selected in a given context, composition would proceed according to standard function application. However, such an explanation does not account for the close relationship between the word interpretations and the productive nature of the phenomenon. The examples above are rather instances of what Pustejovsky and Batiukova (2019) call **selectional polysemy**: They result from systematic and general processes that lead lexemes to acquire new meanings or modulate their interpretation in context. Jackendoff (1997, 2002) argues that Fregean Compositionality must be complemented with mechanisms of **enriched composition** that integrate lexical meaning with pragmatic information about the extralinguistic context, which is therefore an integral component of the process of constructing complex semantic representations. Similarly, the **Generative Lexicon** theory by Pustejovsky (1995) does not regard compositionality as the mere combination of lexical meanings, but rather as a generative process that allows words to acquire new senses in context. This is obtained by representing lexical items with complex structures that contain elements of contextual knowledge (cf. the lexical entry of *book* in Section 8.1). Classical function application is

SENSE-ENUMERATIVE EXPLANATION

SELECTIONAL POLYSEMY

ENRICHED COMPOSITION

GENERATIVE LEXICON

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

then extended to access the rich information inside lexical representations and model phenomena like co-compositionality and logical metonymy.

QUASI-COMPOSITIONALITY

After this brief overview of the characters and problems of compositionality, we can conclude that, in order to account for natural language creativity and productivity, we need to explain (i) *how lexical items can be combined to construct the interpretation of complex linguistic expressions*, as well as (ii) *how word meaning is affected by the compositional processes and change in context*. In fact, as Rabovsky and McClelland (2019) claim, natural language understanding should be better described as **quasi-compositional**. Symbolic semantic theories provide a general formal model of meaning composition in terms of function-argument structures and function application as the operation to carry out predication. On the other hand, the assignment of the proper interpretation to lexical items is typically presupposed rather than explained by symbolic models. These also struggle to provide a general and satisfactory explanation of the dynamic processes leading to the formation of new senses in context, which is often achieved at the cost of complicating lexical representations or introducing ad hoc compositional mechanisms. The gradient and continuous nature of distributional representations is particularly appealing as a computational model of the context sensitivity of word meaning. However, we first need to understand how to build vectors representing complex linguistic expressions and to what extent distributional semantics can offer an empirically adequate account of phrase and sentence meaning.

9.2 Vector Composition Functions

PHRASE AND SENTENCE VECTORS

VECTOR COMPOSITION

The most common approach in distributional semantics assumes the representation of phrase and sentence meaning to be a **vector**, exactly like lexical items. An influential method to build **phrase and sentence vectors** consists in applying **vector composition functions** (see Table 9.2) to the distributional vectors of lexical items (Mitchell and Lapata, 2010).

Let $[_{\mathrm{X}}\mathrm{Y}\ \mathrm{Z}]$ be a syntactic node, $\mathbf{u}$ the distributional vector of Y, $\mathbf{v}$ the distributional vector of Z, and f a **vector composition function**. The distributional representation of X is the vector $\mathbf{p} = f(\mathbf{u}, \mathbf{v})$.

We henceforth refer to $\mathbf{u}$ and $\mathbf{v}$ as the input vectors and to $\mathbf{p}$ as the output vector of the composition function.

ADDITIVE MODEL

VECTOR ADDITION

The **(simple) additive model**, which was first introduced by Landauer and Dumais (1997), uses the **vector addition** to compose the input vectors:

$$\mathbf{p} = \mathbf{u} + \mathbf{v} \tag{9.1}$$

Table 9.2 Main types of vector composition functions

Model	*Vector composition function*
simple additive	$\mathbf{u}+\mathbf{v}$
weighted additive	$\alpha\mathbf{u}+\beta\mathbf{v}$
full additive	$\mathbf{A}\mathbf{u}+\mathbf{B}\mathbf{v}$
multiplicative	$\mathbf{u}\odot\mathbf{v}$
tensor product	$\mathbf{u}\otimes\mathbf{v}$
circular convolution	$\mathbf{u}\circledast\mathbf{v}$

$$\begin{array}{c c}
\mathbf{u} & (\,2 \quad 0 \quad 3 \quad 1 \quad 0\,) \\
\mathbf{v} & (\,1 \quad 2 \quad 2 \quad 0 \quad 0\,) \\
\hline
\mathbf{u}+\mathbf{v} & (\,3 \quad 2 \quad 5 \quad 1 \quad 0\,)
\end{array}$$

Figure 9.2 Additive vector composition

Given a linguistic sequence $s = l_1, \ldots, l_n$, such as a phrase, a sentence, or a whole text, the distributional representation of s is defined as follows:

$$\mathbf{s} = \sum_{i=1}^{n} \mathbf{l_i}. \tag{9.2}$$

The vector addition is arguably the most common composition method in distributional semantics. It generates output vectors that retain *all* elements of the input vectors. As illustrated in Figure 9.2, shared elements get higher weight in the composed vector. In set theoretic terms, the meaning of the composed expression is the *union* of the distributions of the components. This implies that all the features of **red** and **car** are also features of the combined vector **red car**. In neural network models, vector addition is traditionally considered equivalent to **logical conjunction** (Smolensky, 1990). Therefore, **red** + **car** corresponds to saying that *red car* means something that is a car *and* is red (i.e., λx[red(x) $\wedge$ car(x)]).

LOGICAL CONJUNCTION

The sum vector points in an intermediate direction with respect to the input vectors (cf. Mathematical note 6 in Section 7.2.4). In fact, the additive model can be formulated as the **arithmetic average** of the lexical vectors:

VECTOR AVERAGE

$$\mathbf{s} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{l_i}. \tag{9.3}$$

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

Since averaging only rescales the sum vector by n, Equations 9.2 and 9.3 produce vectors that point in the same direction. This entails that vector addition and average are equivalent under cosine similarity, because the cosine is only sensitive to vector direction and not to vector length (cf. Section 2.6.1). Lexical vectors can also be weighted to emphasize the most informative words in the sequence s (Hill et al., 2016; Arora et al., 2017):

$$\mathbf{s} = \frac{1}{n}\sum_{i=1}^{n} w_i \mathbf{l_i}, \tag{9.4}$$

where w is typically tf-idf (cf. Section 2.3.1) or a similar kind of weight.

Despite its simplicity, the additive model is extremely competitive even with respect to more sophisticated approaches (cf. Sections 9.3 and 9.4). On the other hand, it has several shortcomings that undermine its plausibility as a general model for composing distributional representations. First of all, the additive model simply "mixes" word meanings, thereby leading to the rather counterintuitive consequence that the semantic representation of a complex expression is something in-between the interpretation of its lexical components. However, the meaning of *red car* is not the average of the meanings of *red* and *car*. Secondly, since vector addition is commutative (i.e., $\mathbf{u} + \mathbf{v} = \mathbf{v} + \mathbf{u}$), the additive model ignores syntactic structure completely. This is the reason why it is also called **continuous bag-of-words** (CBOW) model.[4] Thus, the sentences *Dogs bite men* and *Men bite dogs* are assigned identical semantic representations. Moreover, lexical vectors are not affected by the additive process: The distributional representation of *red car* and *red wine* contains the very same *red* vector. Therefore, the additive model is not able to address the meaning modulations determined by the composition process (cf. Section 9.1.1). Finally, the additive model makes linguistic expressions more ambiguous as their length increases, because the distributional vector will eventually include all the information brought by their lexical items. This contrasts with the fact that linguistic context typically reduces lexical ambiguities. For example, the verb *play* is ambiguous between a music and a game sense, and the noun *bass* can mean either a type of fish or a kind of musical instrument, but the phrase *play bass* has just the musical interpretation. In fact, vector composition should select those aspects of lexical meanings that are mutually relevant, rather than simply adding all their possible interpretations.

CBOW

Various improvements of the simple additive model have been proposed to overcome some of its problematic aspects. Mitchell and Lapata (2008) add weights to encode the order of the sequence:

[4] Not to be confused with the homonymous DSM in the `word2vec` library (cf. Section 6.3).

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

$$\begin{array}{c|ccccc}
\mathbf{u} & (\,2 & 0 & 3 & 1 & 0\,) \\
\mathbf{v} & (\,1 & 2 & 2 & 0 & 0\,) \\
\hline
\mathbf{u} \odot \mathbf{v} & (\,2 & 0 & 6 & 0 & 0\,)
\end{array}$$

Figure 9.3 Multiplicative vector composition

$$\mathbf{p} = \alpha\mathbf{u} + \beta\mathbf{v}, \qquad (9.5)$$

where α and β are weight factors. Assuming as an example $\alpha = 0.4$ and $\beta = 0.6$, we would multiply $\mathbf{u}$ with 0.4 and $\mathbf{v}$ with 0.6 before summation. This way, different distributional representations can be assigned to the sentences *Dogs bite men* and *Men bite dogs* , respectively the vector $\alpha\mathbf{dogs} + \beta\mathbf{bite} + \gamma\mathbf{men}$ and the vector $\alpha\mathbf{men} + \beta\mathbf{bite} + \gamma\mathbf{dogs}$.

The simple additive model in Equation 9.1 and its variant in Equation 9.5 are then generalized as follows (Mitchell and Lapata, 2008, 2010):

$$\mathbf{p} = \mathbf{Au} + \mathbf{Bv}, \qquad (9.6)$$

where $\mathbf{A}$ and $\mathbf{B}$ are $n \times n$ weight matrices that determine the different contributions made by $\mathbf{u}$ and $\mathbf{v}$, thereby breaking the symmetry of vector sum.[5] This is called the **full additive model** (Zanzotto et al., 2010; Dinu et al., 2013). As the word vectors are multiplied by weight matrices before being summed, the full additive model can capture the effects of word order and contextual meaning shifts occurring in the composition. Guevara (2010, 2011) and Zanzotto et al. (2010) estimate the weight matrices in Equation 9.6 with multivariate linear regression trained to predict the output vector $\mathbf{p}$ from the input vectors $\mathbf{u}$ and $\mathbf{v}$. Guevara (2010) focuses on adjective-noun combinations such as *red car*: Partial least squares regression learns to predict the holistic vectors for the training adjective-noun phrases from the vectors of the component words (cf. Section 9.3). In Zanzotto et al. (2010), the regression model is trained to map the distributional vector of an input pair (e.g., *close contact*) onto the vector of a single-word paraphrase (e.g., *interaction*).

FULL ADDITIVE MODEL

Compositional models can also be defined in terms of vector multiplication instead of addition (Mitchell and Lapata, 2008, 2010). The **multiplicative model** composes vectors by multiplying them elementwise (Figure 9.3):

MULTIPLICATIVE MODEL

$$\mathbf{p} = \mathbf{u} \odot \mathbf{v}. \qquad (9.7)$$

[5] Equation 9.6 corresponds to the original formulation of the model in Mitchell and Lapata (2010), which assumes distributional vectors to be column vectors (i.e., $n \times 1$ matrices). With row vectors (i.e., $1 \times n$ matrices), $\mathbf{p} = \mathbf{uA} + \mathbf{vB}$ (cf. Mathematical note 4 in Section 2.5.2).

The output vector retains the elements the input vectors have in common. In set theoretic terms, the meaning of a phrase is the *intersection* of the distributions of its lexemes. This implies that the features of *red car* are *only* the features shared by *red* and *car*. Interestingly, the multiplicative model also averages the lexical vectors. In fact, each p_i is the square of the **geometric average** $\sqrt{u_i v_i}$ between the corresponding components in the input vectors.

GEOMETRIC AVERAGE

Like addition, vector multiplication is commutative (i.e., $\mathbf{u} \odot \mathbf{v} = \mathbf{v} \odot \mathbf{u}$), but it avoids the problem of increasing the ambiguity of the composed expression. Since it selects only those dimensions that are mutually relevant for the component words, it performs an implicit form of disambiguation and captures meaning modulation in context. Mitchell and Lapata (2008) show that the multiplicative model outperforms the additive one in measuring how an intransitive verb's meaning is modified by its subject, and Mitchell and Lapata (2010) prove its superiority in a phrase similarity task with adjective–noun, noun–noun, and verb–noun pairs (cf. Section 9.5). On the other hand, the multiplicative model does not easily scale up to longer sequences, because it incrementally reduces the information brought by the resulting vector. For instance, if we multiply sparse vectors, the number of zero components increases with sentence length (Grefenstette and Sadrzadeh, 2015). More generally, by filtering out dimensions that are not shared by all the lexical items, the vector product determines the counterintuitive result that the longer a linguistic expression, the less information is encoded in its semantic representation. As a matter of fact, Hill et al. (2016) report a very poor performance of the multiplicative model for building sentence vectors of arbitrary length.

TENSOR PRODUCT

A variant of the multiplicative model is the **tensor product** (Smolensky, 1990; Clark and Pulman, 2007; Mitchell and Lapata, 2008; Widdows, 2008):

$$\mathbf{p} = \mathbf{u} \otimes \mathbf{v}. \tag{9.8}$$

Tensors are multidimensional arrays of numbers, and can be thought of as a generalization of vectors and matrices, where a vector is a first-order tensor and a matrix is a second-order tensor (cf. Section 4.3.1). Given two n-dimensional vectors $\mathbf{u}$ and $\mathbf{v}$, the tensor product $\mathbf{u} \otimes \mathbf{v}$ is a matrix $M_{n \times n}$ such that $m_{i,j} = u_i v_j$ (Figure 9.4a). The tensor product is asymmetric and can therefore account for word order. On the other hand, its major drawback is the increase in dimensionality of the output representation as more items are composed. For instance, $\mathbf{dogs} \otimes \mathbf{chase}$ is a second-order tensor,

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

$$\begin{array}{cc}
\mathbf{u} & (2 \quad 0 \quad 3 \quad 1 \quad 0) \\
\mathbf{v} & (1 \quad 2 \quad 2 \quad 0 \quad 0) \\
\hline
\mathbf{u} \otimes \mathbf{v} & \begin{pmatrix} 2 & 4 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 3 & 6 & 6 & 0 & 0 \\ 1 & 2 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}
\end{array}$$

(a)

$$\mathbf{u} \circledast \mathbf{v} \begin{pmatrix} \mathbf{4} \\ {\color{gray}4} \\ 7 \\ 7 \\ 8 \end{pmatrix} = \begin{pmatrix} \mathbf{m_{5,2}} + \mathbf{m_{4,3}} + \mathbf{m_{3,4}} + \mathbf{m_{2,5}} + \mathbf{m_{1,1}} \\ {\color{gray}\mathbf{m_{5,3}} + \mathbf{m_{4,4}} + \mathbf{m_{3,5}} + \mathbf{m_{1,2}} + \mathbf{m_{2,1}}} \\ m_{5,4} + m_{4,5} + m_{1,3} + m_{2,2} + m_{3,1} \\ m_{5,5} + m_{1,4} + m_{2,3} + m_{3,2} + m_{4,1} \\ m_{5,1} + m_{4,2} + m_{3,3} + m_{2,4} + m_{1,5} \end{pmatrix} \Leftarrow \begin{pmatrix} \mathbf{2} & {\color{gray}4} & 4 & 0 & 0 \\ {\color{gray}0} & 0 & 0 & 0 & \mathbf{0} \\ 3 & 6 & 6 & \mathbf{0} & {\color{gray}0} \\ 1 & 2 & \mathbf{2} & {\color{gray}0} & 0 \\ 0 & \mathbf{0} & {\color{gray}0} & 0 & 0 \end{pmatrix}$$

(b)

Figure 9.4 (a) Tensor product vector composition and (b) its compression with circular convolution

$\mathbf{dogs} \otimes \mathbf{chase} \otimes \mathbf{cats}$ is a third-order tensor, $\mathbf{dogs} \otimes \mathbf{chase} \otimes \mathbf{black} \otimes \mathbf{cats}$ is a fourth-order tensor, and so forth. Tensors can be unfolded into matrices (cf. Section 4.3.2) that can then be vectorized by concatenating their columns or rows. If $\mathbf{u}$ and $\mathbf{v}$ have n dimensions, $\mathbf{u} \otimes \mathbf{v}$ corresponds to a vector with n^2 dimensions. Since the number of dimensions of the resulting vector grows exponentially with the tensor order (e.g., given word vectors of 300 dimensions, a ten-word sentence would be represented by a vector of 5.9×10^{24} dimensions), tensor product quickly becomes computationally intractable. Moreover, linguistic expressions of different length cannot be compared because they correspond to vectors with a different number of dimensions.

One way to deal with this exploding dimensionality is to use **circular convolution** (Plate, 2003; Jones and Mewhort, 2007) to compress the tensor product into a vector of the same number of dimensions of the original input vectors: CIRCULAR CONVOLUTION

$$\mathbf{p} = \mathbf{u} \circledast \mathbf{v}, \tag{9.9}$$

where $\circledast$ denotes circular convolution, defined as follows (cf. Section 5.4):

$$\mathbf{p}_i = \sum_{j=o}^{n-1} \mathbf{u}_{j \bmod n} \cdot \mathbf{v}_{i-j \bmod n}. \tag{9.10}$$

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

The subscripts are modular over n (e.g., $j \mod n$ is the remainder of j/n), and the operation sums along the transdiagonal elements of the tensor product, which is what gives the operation its circular nature (see the black and gray boldfaced entries in the matrix in Figure 9.4b). Importantly, circular convolution is specifically designed for random encoding vectors (Grefenstette and Sadrzadeh, 2015). In fact, Mitchell and Lapata (2010) show that it greatly underperforms when applied to other kinds of distributional representations (e.g., explicit vectors and topic vectors).

FILLER-ROLLER BINDINGS

The tensor product is proposed by Smolensky (1990) as a model of the variable-argument binding operations (e.g., function saturation) typical of compositional symbolic structures. A sequence s of n elements is decomposed into the sum of n **filler-roller bindings**, each corresponding to the tensor product of a filler vector $\mathbf{f}$ representing an element in the sequence, and a vector $\mathbf{r}$ representing its role in the sequence:

$$\mathbf{s} = \sum_{i=1}^{n} \mathbf{f_i} \otimes \mathbf{r_i}. \tag{9.11}$$

Roles can correspond to positions in a sentence, syntactic relations, semantic roles, and so on. For instance, given $\mathbf{p_1}, \ldots, \mathbf{p_n}$ role vectors, such that $\mathbf{p_i}$ encodes the ith position of a lexeme in a sentence, *Dogs bite men* is represented with the vector $(\mathbf{dogs} \otimes \mathbf{p_1}) + (\mathbf{bite} \otimes \mathbf{p_2}) + (\mathbf{men} \otimes \mathbf{p_3})$. The additive nature of Equation 9.11 makes the dimensionality of the vector independent of the sequence length. On the other hand, the filler-role binding carried out by the tensor product allows sentence vectors to encode differences in word order or argument structure saturation (e.g., *Dogs bite men* and *Men bite dogs* are associated with different vectors). McCoy et al. (2019b) implement this model with a seq2seq recurrent neural network (cf. Sections 6.2.1 and 9.4.3) that learns role vectors and combines them with pretrained filler embeddings.

9.2.1 Predicting the Compositionality of Multiword Expressions

MULTIWORD EXPRESSIONS

Besides expressions whose interpretation is obtained compositionally, natural languages contain a wide array of (semi-)preconstructed phrases. These are commonly referred to as **multiword expressions** (MWEs) and include several types of constructions such as idioms, compounds, light verb constructions, and phrasal verbs (cf. Section 2.1.3). An interesting application of vector composition functions is to predict the compositionality of MWEs.

Sag et al. (2002) define MWEs as "idiosyncratic interpretations that cross word boundaries" (p. 2), since their meaning is not simply a function of the content of their constituent words. MWEs display a continuum of compositionality. For instance, the light verb construction *give confidence* is more

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

compositional than the idiomatic expressions *give a whirl*, whose interpretation of giving a try cannot be derived from the meaning of its parts (Fazly and Stevenson, 2008). In fact, different shades of (non)compositionality exist among idiomatic expressions too: Nunberg et al. (1994) distinguish between **idiomatically combining expressions** like *pop the question*, where each idiom component directly contributes its literal meaning to the figurative meaning of the whole string, and **idiomatic phrases** like *kick the bucket*, whose figurative meaning is distributed over the phrase as a whole and cannot be traced back to any of its parts. Therefore, the compositionality of linguistic expressions is more a gradient property than a dichotomous phenomenon.

IDIOMATICALLY COMBINING EXPRESSIONS

IDIOMATIC PHRASES

A simple but effective approach to **compositionality prediction** is to measure the similarity between the vector of the MWE and the vector obtained by composing the vector of its lexical parts:

COMPOSITIONALITY PREDICTION

Given a MWE $m = l_1, l_2$, the **compositionality** of m, $\mathrm{comp}(m)$, is:

$$\mathrm{comp}(m) = \mathrm{sim}(\mathbf{m}, \alpha\mathbf{l_1} + \beta\mathbf{l_2}), \tag{9.12}$$

where $\mathbf{m}$ is the **holistic vector** of m (i.e., the vector of the MWE treated as a single target lexeme), and sim is a vector similarity measure (e.g, the cosine). For example, the compositionality of *give confidence* is estimated with the similarity between the holistic vector $\mathbf{give\_confidence}$ and the sum of $\mathbf{give}$ and $\mathbf{confidence}$. Though various vector composition functions can be used, experimental results show that addition is the best-performing one (Reddy et al., 2011a; Salehi et al., 2015). The α and β weights in Equation 9.12 determine the asymmetric contribution of each of the components to the meaning of the whole MWE. In *give confidence*, it is the noun rather than the verb that transparently occurs in the overall meaning of the construction, while in *break the ice* the reverse situation applies. Some methods compare the holistic vector of the MWE only with the vector of either of its lexical parts (Fazly and Stevenson, 2008; Salehi et al., 2015). This can be considered as a special case of Equation 9.12 with α or β equal to zero.

HOLISTIC VECTOR

Most research has focused on noun compounds, but other types of MWEs, such as light verb constructions and particle verbs, have been targeted too. DSMs are tested on predicting compositionality ratings. Reddy et al. (2011a) introduce a dataset of judgments for 90 English noun–noun and adjective–noun compounds, in terms of three numerical scores corresponding to the compositionality of the compound and the literal contribution of each of its parts. Similar ratings have been collected for German by Schulte im Walde et al. (2013) and by Cordeiro et al. (2019) for several languages. The latter study shows that DSMs are able to achieve a strong correlation with human ratings, supporting the potentialities of distributional semantics for compositionality prediction.

9.3 The Distributional Functional Model (DFM)

The compositional DSMs we have seen so far share the assumption that every lexical item is represented with the *same type* of entity, namely a distributional vector. The interpretation of a complex linguistic expression is a new vector resulting from averaging (with addition or multiplication) the features of the input lexical vectors, hence the name of "vector-mixture models" suggested by Baroni et al. (2014b). This is a major difference with respect to symbolic models of Fregean Compositionality, which assume that semantic representations result from the application of functions to arguments, as distinct types of formal entities (cf. Section 9.1).

Distributional Functional Model

We use the term **Distributional Functional Model** (DFM) to refer to a group of related proposals (Clark et al., 2008; Baroni and Zamparelli, 2010; Coecke et al., 2010; Grefenstette et al., 2011; Baroni et al., 2014b; Clark, 2015; Grefenstette and Sadrzadeh, 2015; Maillard et al., 2015) that are directly inspired by Montague's type-driven theory and aim at providing a distributional interpretation of function application as the core operation to carry out predication. A key aspect of DFM is that the meaning of predicates is represented with higher-order linear-algebraic objects (e.g., matrices and tensors). Like Montague Semantics, DFM specifies a mapping between syntactic categories and semantic types. While Montagovian types define denotation domains, DFM includes **distributional types** that specify the kinds of linear-algebraic objects representing categories of linguistic expressions.

distributional types

DFM takes Montague's distinction between basic saturated types, which are assigned to nouns and sentences, and unsaturated functional types. The basic distributional types are vectors:[6]

> The distributional representation of **nouns** and **sentences** are **vectors**.

distributional functions

one-place predicates as matrices

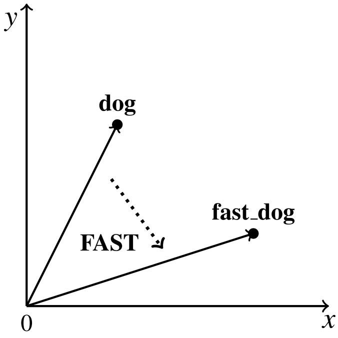
Predicates are represented with **distributional functions** that transform linear-algebraic objects into other objects. Intransitive verbs and adjectives express one-place distributional functions and are represented as **linear transformations** (or **linear maps**), in turn corresponding to **matrices** (cf. Mathematical

[6] An important difference from Montague Semantics is that DFM treats proper and common nouns alike (Baroni et al., 2014a).

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

$$\underbrace{\begin{pmatrix}0.5 & 1\\ 0.8 & 0\end{pmatrix}}_{\textbf{FAST}}\underbrace{\begin{pmatrix}3\\ 6\end{pmatrix}}_{\textbf{dog}} = \begin{pmatrix}(0.5*3)+(1*6)\\ (0.8*3)+(0*6)\end{pmatrix} = \underbrace{\begin{pmatrix}7.5\\ 2.4\end{pmatrix}}_{\textbf{fast\_dog}}$$

(a)



(b)

Figure 9.5 (a) In DFM, the compositional interpretation of *fast dog* is obtained with the matrix-vector product; (b) the distributional representation of the adjective *fast* is a linear map from the vector **dog** onto the vector representing the noun phrase *fast dog*

note 4 in Section 2.5.2). Their application to an argument noun is modeled with matrix-vector multiplication:[7]

> The distributional representation of a one-place predicate $\lambda\mathrm{x}[\mathrm{P(x)}]$ is a **matrix** $\mathbf{P}_{m\times n}$. Given an argument a represented by the n-dimensional vector $\mathbf{a}$, the distributional representation of $\lambda\mathrm{x}[\mathrm{P(x)}](a) = \mathrm{P}(a)$ is the m-dimensional vector $\mathbf{b} = \mathbf{Pa}$.

Figure 9.5 shows the compositional interpretation of the phrase *fast dog*, which takes from Montague Semantics the idea that attributive adjectives are functions that map the meaning of the noun head onto the meaning of the modified noun (Montague, 1974; Baroni and Zamparelli, 2010; Coecke et al., 2010). The adjective *fast* is represented with the matrix $\mathbf{FAST}$, which is multiplied by $\mathbf{dog}$ to produce the vector of the noun phrase *fast dog*.

N-PLACE PREDICATES AS TENSORS

The representation of one-place predicates as matrices is generalized to n-place predicates with **tensors**, which also correspond to linear transformations:

[7] We follow the original notation of DFM, in which distributional vectors are column vectors.

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

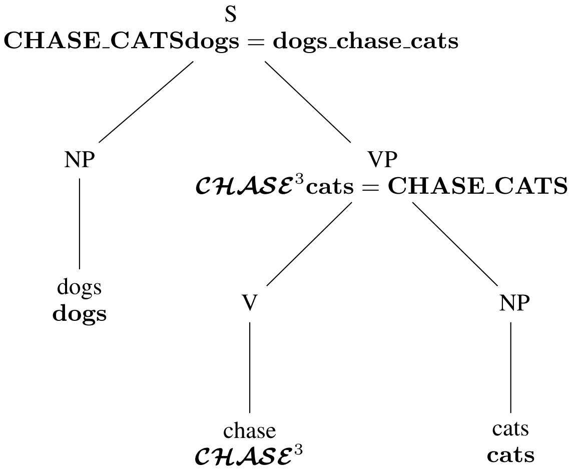


Figure 9.6 The compositional derivation of the vector representing the meaning of the sentence *Dogs chase cats* in DFM

> The distributional representation of a n-place predicate $\lambda \mathrm{x}_1, \ldots, \lambda \mathrm{x}_\mathrm{n}[\mathrm{P}(\mathrm{x}_1, \ldots, \mathrm{x}_\mathrm{n})]$ is an ***n*+1th-order tensor** $\mathcal{P}^{n+1}$.

For instance, the two-place transitive verb *chase* is represented with a third-order tensor $\mathcal{CHASE}^3$ viewed as a linear map of a noun vector onto a matrix encoding the verb phrase meaning. This explicitly resembles Montague's analysis of transitive verbs as expressions of type $\langle e, \langle e, t\rangle\rangle$ that denote functions from one argument into one-place functions. Matrix-vector product is also extended to tensors as a general linear-algebraic model of function application. Since matrices and vectors are respectively second- and first-order tensors, semantic composition in DFM can be expressed in the following generalized form, adopting the rule-by-rule interpretation typical of Montague Semantics:

> Let $[_\mathrm{X}\mathrm{Y}\ \mathrm{Z}]$ be a syntactic node. The distributional representation of X is the tensor $\mathcal{X}^{n-1}$ such that:
>
> $$\mathcal{X}^{n-1} = \begin{cases} \mathcal{Y}^n\mathbf{z} & \text{if } \mathcal{Y}^n \text{ represents Y and the vector } \mathbf{z} \text{ represents Z} \\ \mathcal{Z}^n\mathbf{y} & \text{if } \mathcal{Z}^n \text{ represents Z and the vector } \mathbf{y} \text{ represents Y} \end{cases}$$

As illustrated in Figure 9.6, the tensor $\mathcal{CHASE}^3$ is multiplied by the object NP vector **cats** to produce the matrix **CHASE_CATS** that represents the VP. The sentence vector is then obtained by multiplying the VP matrix by the subject NP vector **dogs**.

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

LINEAR REGRESSION

While noun vectors are generated with standard DSMs, DFM learns the tensors that represent the distributional functions with **linear regression**, adapting the method introduced by Guevara (2010). For instance, Baroni and Zamparelli (2010) learn the matrix of the adjective *fast* by training a regression model to predict output holistic vectors of *fast*-noun pairs (e.g., *fast car*) from the input vector of the noun head. The regression weights form the entries of the adjective matrix **FAST**. Differently from Guevara (2010) and Zanzotto et al. (2010), Baroni and Zamparelli (2010) train a distinct regression model for each adjective. Similarly, the matrix representing the intransitive verb *run* consists of the regression weights learned by predicting the holistic vector of subject-*run* pairs (e.g., *horse runs*) from the subject vector. This method is generalized to train higher-order tensors by Grefenstette et al. (2013) with multi-step regression. To learn the tensor $\mathcal{CHASE}^3$, matrices for VPs (e.g., *chase car*, *chase ball*, etc.) are estimated with linear regression on input subject and output holistic subject–verb–object vector pairs. The final tensor is then obtained by applying linear regression to predict the matrices estimated in the previous step from the vectors for the corresponding objects. The weights in the tensor entries thus capture possible dependencies between subject and object arguments (e.g., *cats* is the typical subject of the VP *chase mice*, while *cops* is the preferred subject of the VP *chase thieves*).

DFM is an elegant attempt to provide a distributional model of Fregean Compositionality and function application, and it has been applied to address various semantic phenomena, with a particular focus on adjectival modification (Baroni and Zamparelli, 2010; Asher et al., 2016; Vecchi et al., 2016). For instance, treating adjectives as matrices that map noun vectors onto adjective-noun vectors captures phenomena of co-composition and contextual meaning shift (e.g., the interpretation of *heavy* is different in *heavy rain* and *heavy book*), and Boleda et al. (2012a, 2013) adopt this approach to study the difference between intersective, subsective, and intensional adjectives.[8] Gutiérrez et al. (2016) apply the DFM to model metaphorical uses of adjectives (e.g., *cold* has a figurative interpretation in *cold reception*). Marelli and Baroni (2015) use DFM to provide a compositional representation of morphologically complex words: Affixes are represented with matrices and the vector of a derived word like *nameless* is built by multiplying the matrix of the affix

[8] An adjective is **intersective** if the denotation of the adjective–noun pair AN is the intersection of the denotations of A and N, and therefore AN entails both A and N: a *red car* is both *red* and a *car*. An adjective is **subsective** if the denotation of AN is a subset of the denotation of N, and AN entails N: a *skillful violinist* is a *violinist*, but not necessarily *skillful* in general. An adjective is **intensional** if AN does not entail N: an *alleged terrorist* is not necessarily a *terrorist* (Pustejovsky and Batiukova, 2019).

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

-less by the vector of the base *name* (e.g., $\mathbf{nameless} = \mathbf{LESSname}$). This solution is extended by Marelli et al. (2017) to compounds such as *boathouse*.

THE LIMITS OF DFM

Despite these interesting applications, the definition of distributional types as a strict association between n-place functions and $n + 1^{th}$-order tensors is problematic for several reasons. First of all, the complexity of the model grows exponentially with the number of predicate arguments. Given vectors of 300 dimensions, two-place predicates are represented with tensors formed by $300^3 = 27$ million entries. A three-place predicate like *give* is instead represented with a fourth-order tensor of $300^4 = 81$ billion entries. This means that estimating the distributional functions becomes unwieldy and runs against huge data sparseness problems, especially for less frequent lexical items. Secondly, it is not clear how DFM can account for unexpressed arguments. For instance, the verb *eat* is transitive, but it can also be used intransitively in a sentence like *Dogs eat*. If *eat* is represented with a third-order tensor, than it is not possible to derive the representation of *Dogs eat*, unless we postulate two distinct representations for transitive and intransitive *eat*. Moreover, since in DFM nouns and verbs correspond to tensors of different orders, they cannot be directly compared. Thus, the model cannot account for the strong similarity between deverbal nouns like *destruction* and their base verb (*destroy*). In general, DFM inherits from Montague Semantics most of the limits stemming from a one-to-one mapping between semantic types and syntactic categories. In fact, phenomena like coercion and logical metonymy can hardly be addressed in DFM, unless by introducing type-shifting mechanisms similar to those often adopted in formal semantics (Pustejovsky and Batiukova, 2019). Therefore, while DFM offers a promising approach to explain local compositional phenomena, such as adjectival modification, it has great difficulties to tackle more complex linguistic constructions.

PLF

In order to overcome some of these limits, Paperno et al. (2014) propose a simplification of the DFM called **Practical Lexical Function** model (PLF), which represents every lexical item l in the following way:

$$\langle \mathbf{l}, \mathbf{L_1}, \ldots, \mathbf{L_n} \rangle, \tag{9.13}$$

where $\mathbf{l}$ is a standard distributional vector and $\mathbf{L_i}$ is a matrix corresponding to the ith argument slot of l. The number of matrices associated with a lexical item depends on its arity. A word without arguments like *dog* is represented just with its vector (i.e., $\langle \mathbf{dog} \rangle$). The one-place predicate *run* is associated with a vector and a matrix for the subject slot (i.e., $\langle \mathbf{run}, \mathbf{RUN_S} \rangle$), and the two-place predicate *chase* with a vector and two matrices for the subject and the object

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

slots (i.e., $\langle \mathbf{chase}, \mathbf{CHASE_S}, \mathbf{CHASE_O} \rangle$). Matrices are estimated with linear regression from corpus-extracted holistic vectors of predicate-argument pairs and predication is modeled with matrix-vector product: Each matrix slot is multiplied by the vector argument, and the result is summed to the predicate vector. The interpretation of the sentence *Dogs chase cats* is constructed as follows, like in the filler-role bindings model (cf. Section 9.2):

$$\mathbf{chase} + \mathbf{CHASE_S dogs} + \mathbf{CHASE_O cats} \tag{9.14}$$

Gupta et al. (2015) slightly amend PLF by not adding the verb vector, in order to solve a problem in the original formulation of the model. By representing every lexical item only with vectors and matrices, PLF avoids the problems deriving from higher-order tensors, though this way it gives up capturing dependencies between different arguments. Moreover, since predication is carried out with matrix-vector product, PLF can assign different vectors to *Dogs chase cats* and *Cats chase dogs*, thereby bypassing the problem of simple additive model (cf. Section 9.2). On the other hand, Rimell et al. (2016) show that even the much more versatile PLF is not able to outperform the simple additive model when tested on the RELPRON dataset (cf. Section 9.5).

9.3.1 Matrix-Vector Recursive Neural Networks (MV-RNN)

The **Matrix-Vector Recursive Neural Network** (MV-RNN) by Socher et al. (2012) shares with DFM the idea of representing the meaning of lexical items with higher-order linear-algebraic objects. While DFM follows Montague Semantics in assigning functional semantic types only to certain categories of lexemes (e.g., adjectives, verbs, etc.), MV-RNN assumes that any word can both have a content and act as an operator (i.e., a function) that modifies the meaning of the expressions it combines with. Therefore, every lexical item l is represented with the pair $\langle \mathbf{l}, \mathbf{L} \rangle$, such that $\mathbf{l}$ is a n-dimensional vector and $\mathbf{L}$ is a $n \times n$ item-specific matrix. The vector is expected to encode the meaning of l, and the matrix to capture its operator semantics. Words may differ for the relative salience of either component: If a word has mainly a functional nature (e.g., the negation *not*), its vector will be close to zero, while if a word lacks operator semantics (e.g., the noun *dog*), its matrix will be the identity matrix $\mathbf{I}$ (cf. Section 8.4.1), corresponding to the identity function (i.e., $\mathbf{uI} = \mathbf{u}$). MV-RNN

Like PLF, MV-RNN dispenses with higher-order tensors, but instead of combining lexemes with matrix-vector product the composition function is now a feed-forward neural network (cf. Section 6.1) that recursively projects the vector-matrix pairs of two child nodes in the syntactic tree onto the vector-matrix pair representing their parent node:

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

Let $[_{\text{X}}\text{Y Z}]$ be a syntactic node, $\langle \mathbf{y}, \mathbf{Y} \rangle$ and $\langle \mathbf{z}, \mathbf{Z} \rangle$ the representations of Y and Z. The representation of X is the pair $\langle \mathbf{x}, \mathbf{X} \rangle$ such that:

$$\mathbf{x} = g([\mathbf{yZ}; \mathbf{zY}]\mathbf{W}) \quad \mathbf{X} = g([\mathbf{Y}; \mathbf{Z}]\mathbf{W_M}), \tag{9.15}$$

where $g = \tanh$, ";" denotes concatenation, and $\mathbf{W}$ and $\mathbf{W_M}$ are $2n \times n$ network weight matrices. First, the vector of either child node is multiplied by the matrix of the other, to capture co-composition effects. The resulting vectors are then concatenated and mapped onto the vector of the parent node by the weight matrix $\mathbf{W}$, followed by the nonlinear function g. The concatenation of the component matrices is mapped onto the parent node matrix in a similar way, but using the weight matrix $\mathbf{W_M}$. The composition network is **recursive** (hence the model name), because it is applied to its own previous output until it reaches the tree top node. The lexical vector-matrix pairs and the weights of the composition function are model parameters that are learned with backpropagation by adding a softmax classifier to each parent node and using a supervised task (e.g., sentiment classification). Variations of MV-RNN are proposed by Socher et al. (2011, 2013a).

RECURSIVE COMPOSITION

DFM and its closest relatives like MV-RNN provide a linguistically motivated treatment of phrasal meaning and a theoretically well-grounded distributional interpretation of Fregean Compositionality (Potts, 2019). On the other hand, these compositional DSMs have a high number of parameters to be estimated. Such increased complexity often hampers the model scalability to a large range of linguistic constructions and, at the same time, does not produce substantial improvements in the quality of semantic representations. Though Dinu et al. (2013) report positive empirical results of DFM in various phrase similarity tasks, these are not always significantly better than those achieved by vector addition. Similarly, Blacoe and Lapata (2012) show that a simple additive model is able to match the neural network in Socher et al. (2011).

9.4 Sentence Embeddings

The advent of deep learning has also stormed compositional distributional semantics, introducing a new generation of DSMs that build **sentence embeddings** with artificial neural networks.

SENTENCE EMBEDDINGS

Given a sentence $s = l_1, \ldots, l_n$, the **sentence embedding** $\mathbf{s}$ is a vector such that $\mathbf{s} = N(\mathbf{l_1}, \ldots, \mathbf{l_n})$, where N is a neural network and $\mathbf{l_1}, \ldots, \mathbf{l_n}$ are the embeddings of the words in s.

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

The network N is trained on some self-supervised or supervised task to build a vector $\mathbf{s}$ that encodes relevant aspects of the sentence meaning. The model can learn both word and sentence embeddings from the same training data, but it is also customary to represent words with pretrained vectors built with DSMs such as word2vec or FastText. This way, the network can leverage the semantic information encoded in the pretrained word embeddings.

9.4.1 Paragraph Vector (doc2vec)

Le and Mikolov (2014) propose **Paragraph Vector** (aka **doc2vec**), a generalization of word2vec that learns embeddings for paragraphs, defined as any variable-length piece of texts, ranging from phrases up to sentences and whole documents. In word2vec, words are represented with vectors in an embedding matrix whose parameters are optimized by predicting context words. Similarly, in Paragraph Vector, each paragraph in the training corpus is treated as a kind of new "token" and represented with a distributional vector in a separate embedding matrix whose weights are learned by predicting the words in the paragraph itself. Therefore, Paragraph Vector rests on the assumption that the distributional representation of a linguistic sequence (e.g., a sentence) contains the information relevant to predict its component lexemes. DOC2VEC

Paragraph Vector comes in two variants. Given a paragraph corresponding to a sentence $s = l_1, \ldots, l_n$, the **Distributed Memory Model of Paragraph Vectors** (PV-DM) combines (by average or concatenation) the sentence embedding $\mathbf{s}$ with the embeddings of k consecutive words $l_i, \ldots, l_{i+k}$ sampled from the sentence. This combined vector is then used to predict l_{i+k+1}. For instance, if $s =$ *The dog chases the cat* and $k = 2$, the vector $\mathbf{s}$ is combined with the embeddings of $\mathbf{the}$ and $\mathbf{dog}$ to predict *chases*. While PV-DM is a close relative of CBOW, the **Distributed Bag of Words Paragraph Vector** (PV-DBOW) is similar to SG in word2vec. In this case, the sentence embedding $\mathbf{s}$ is directly used to predict lexemes randomly sampled from s, without including the embeddings of the context words. PV-DM PV-DBOW

Le and Mikolov (2014) finally represent each paragraph with the concatenation of the embeddings produced by PV-DM and PV-DBOW, which are then tested on sentiment analysis and information retrieval tasks. During training, Paragraph Vector stores representations for every sentence (and word) in the training corpus. One limit of this approach is that the model needs to be partially retrained to obtain embeddings for unobserved sentences. For instance, to build the embedding for a new sentence s', a new vector is added to the sentence embedding matrix, which is again trained with backpropagation, after freezing the weights of the word embeddings.

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

SENT2VEC

Another model based on word2vec is **Sent2Vec** (Pagliardini et al., 2018). The CBOW algorithm is extended to learn word and n-gram embeddings. Sentences are then represented as the average of their unigram and n-gram vectors.

9.4.2 Convolutional Neural Networks (CNNs)

CONVOLUTIONAL NEURAL NETWORKS

FILTERS

After being first designed for object recognition in computer vision by LeCun et al. (1998), **Convolutional Neural Networks** (CNNs) have also been applied to NLP tasks (Collobert and Weston, 2008; Collobert et al., 2011) and to build sentence embeddings (Kalchbrenner et al., 2014; Kim, 2014). CNNs consist of a set of **filters** trained to identify features of the input, which are then combined to form a single embedding encoding key aspects of its structure. In computer vision, the filters learn to identify important features of an object like edges, shape, and colors (cf. Section 8.7.1), while in computational linguistics the filters learn features of the most informative n-grams of a sentence.

RECEPTIVE FIELD

Given an input sentence s, each lexeme l_i in s is represented with the embedding $\mathbf{l_i}$. A sliding window of size n (often called **receptive field**) is then applied to s to identify k n-grams $c_1, \ldots, c_k$. Each n-gram c_i is represented with a vector $\mathbf{c_i} = [\mathbf{l_i}; \ldots; \mathbf{l_n}]$ obtained by concatenating the embeddings of its lexemes. The CNN includes q filters corresponding to weight vectors $\mathbf{w_1} \ldots \mathbf{w_q}$ arranged into a matrix $\mathbf{W}$, which are trainable parameters. The filters extract features from the n-grams with a sequence of convolution and pooling layers:

CONVOLUTION

convolution – given the n-gram c_i, the convolution layer produces a new vector $\mathbf{f_i}$ in the following way:

$$\mathbf{f_i} = g(\mathbf{c_i}\mathbf{W} + \mathbf{b}), \qquad (9.16)$$

where g is a nonlinear activation function and $\mathbf{b}$ a bias term. The convolution layers output $\mathbf{f_1}, \ldots, \mathbf{f_k}$ q-dimensional vectors representing the n-grams of the input sentences. The jth component of $\mathbf{f_i}$ encodes the feature extracted by the filter $\mathbf{w_j}$ from the n-gram c_i;

POOLING

pooling – this layer combines the vectors produced by convolution into a single q-dimensional embedding $\mathbf{s}$. The most common pooling operation is the **max pooling**, such that the component s_i is the maximum value for that dimension across the various $\mathbf{f_1}, \ldots, \mathbf{f_k}$ vectors. Alternatively, the **average pooling** is applied to take the average value.

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

Therefore, the CNN represents a sentence with an embedding whose dimensions correspond to features identified by the filters from its various n-grams.

CNN can have more complex architectures with multiple stacks of convolution and pooling layers, or with several convolutions working in parallel on different types of n-grams. An important aspect of the embeddings produced by CNNs is that they are trained with supervised tasks, rather than the self-supervised prediction task used by Paragraph Vector. The output of the pooling layer is passed to a fully connected layer with a softmax classifier. This is used to optimize the network parameters, with backpropagation using annotated sentences (e.g., with sentiment or other semantic labels).

9.4.3 Encoder–Decoder Models (seq2seq)

The most common approach to build neural sentence embeddings is based on the **encoder-decoder** architecture, also known as **sequence-to-sequence** (seq2seq) model (Cho et al., 2014; Sutskever et al., 2014). SEQ2SEQ

> In the **seq2seq** architecture, a network (the **encoder**) is used to produce a vector representation of a sentence s_i, which is then fed into another network (the **decoder**) that uses it to generate a sentence s_j.

From a probabilistic point of view, seq2seq models learn the conditional probability $p(s_j|s_i)$, where s_i and s_j may have different length. These models have been first introduced for machine translation: in this setting, s_j is the translation of s_i in a target language. However, the same approach can be applied to any type of sequence pairs (e.g., s_i can be a question and s_j its answer). The encoder-decoder architecture is successfully used to develop end-to-end NLP applications. However, since the vector produced by the encoder captures several aspects of the input sentence that are useful for the decoder to carry out its task, seq2seq models can also be regarded as general purpose mechanisms to learn sentence vectors (Adi et al., 2017). The encoder and the decoder networks are trained together with backpropagation. Then, the decoder is discarded and the encoder is used to produce embeddings for new sentences.

Encoders and decoders are typically **Long Short-Term Memory** (LSTM) networks (Hochreiter and Schmidhuber, 1997). LSTMs are a type of recurrent network designed to solve some limitations of the Simple Recurrent Networks (SRNs) we have presented in Section 6.2.1, in particular the so-called **vanishing gradient** problem. The gradient is the derivative of the error signal used by backpropagation to update the network weights (cf. Section 6.1). In SRNs, gradients tend to become exponentially smaller during training, making LSTM VANISHING GRADIENT

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

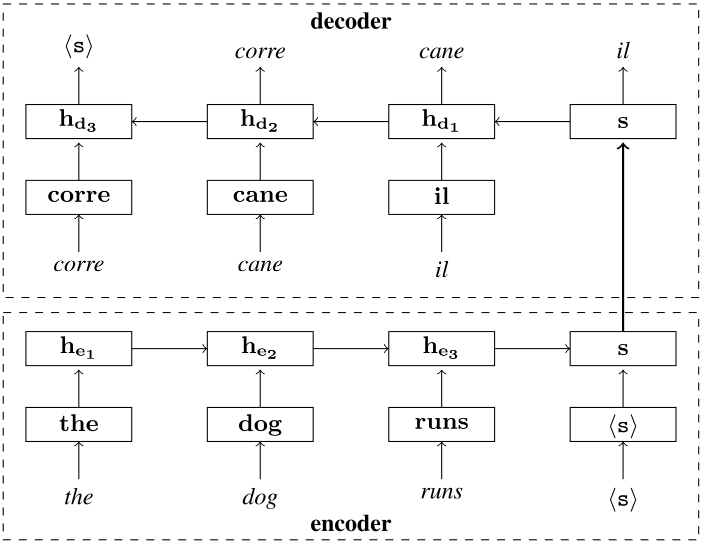


Figure 9.7 Seq-2-seq model that reads the sentence *The dog runs* and generates its Italian translation *Il cane corre*. The encoder and decoder are LSTMs

it difficult for the network to keep track of long-distance dependencies (Goodfellow et al., 2016). Each SRN hidden state $\mathbf{h_i}$ is a function of the current input $\mathbf{l_i}$ and the previous hidden state $\mathbf{h_{i-1}}$. LSTMs have a similar recurrent structure, but the hidden states consist of **memory cells** with a set of **gates** to decide the amount of the input to be written to the memory cell, and the amount of the previous hidden state to be forgotten. A gate is a vector $\mathbf{g}$ of real-valued weights that filter the content of another vector $\mathbf{x}$ by elementwise multiplication $\mathbf{g} \odot \mathbf{x}$ (e.g., the components of $\mathbf{g}$ that are close to zero tend to block the corresponding components of $\mathbf{x}$). The gates are parameters trained to regulate the information flow across the recurrent steps, thereby reducing the vanishing of gradients and improving the ability of the network to encode long-distance dependencies. A similar solution is adopted by **Gated Recurrent Unit** (GRU) networks, introduced for seq2seq modeling by Cho et al. (2014). Other types of encoder-decoder architectures are based on CNNs enriched with attention mechanisms (Gehring et al., 2017) and on Transformers (cf. Section 9.6.3).

MEMORY CELLS
GATES
GRU

Figure 9.7 shows a seq2seq model in which the encoder and the decoder are LSTMs. Each input word is represented with an embedding, which is either randomly initialized or a pretrained distributional vector. Word embeddings may be left fixed during the network training, or may be further fine-tuned (i.e., treated as model parameters) and therefore affected by sentence-level

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

information. The embedding $\mathbf{s}$ of the input sentence *The dog runs* corresponds to the last hidden state of the encoder network ($\langle \mathrm{s} \rangle$ is a special symbol that marks sentence boundaries). The sentence embedding is then used by the decoder to generate the Italian translation of the input, *Il cane corre*. In this example, the sentence embedding initializes the first hidden state of the decoder, like in Sutskever et al. (2014). Alternatively, the sentence embedding can be concatenated to the word embeddings used by the decoder in each step of the output sentence generation (Goldberg, 2017).

AUTOENCODER

SKIPTHOUGHT

FASTSENT

DISSENT

The major difference among seq2seq models of sentence meaning concerns the training task. Dai and Le (2015) and Li et al. (2015) train the model as an **autoencoder** in which the decoder learns to reconstruct the input sentence. **SkipThought** (Kiros et al., 2015) extends the Distributional Hypothesis and assumes that the meaning of a sentence depends on the sentences it co-occurs with. Given a tuple $\langle s_{i-1}, s_i, s_{i+1} \rangle$ of contiguous sentences, the encoder (a recurrent network with GRUs) represents s_i with an embedding that is then used by the decoder to reconstruct the previous and the next sentences. As recurrent networks are slow to train, Hill et al. (2016) propose **FastSent**, a simplification and much faster version of SkipThought in which the encoder and decoder networks are replaced by an additive model. Given a representation of a sentence as the sum of its word embeddings, the network is trained to predict adjacent sentences that are also represented with vector addition. In **DisSent** (Nie et al., 2019), the sentence embeddings are learnt with a discourse prediction task. Given a pair of sentences in the training corpus, an LSTM encodes them with vectors that are used to predict the word (e.g., *but*, *because*, etc.) expressing the discourse relation between them.

PARAGRAM

INFERSENT

UNIVERSAL SENTENCE ENCODER

The models above are trained with variations of a self-supervised language modeling task (cf. Section 6.2). Other seq2seq networks use supervised tasks. Wieting et al. (2016b) extend the **Paragram** model by Wieting et al. (2015) to learn word embeddings and create sentence vectors with an encoder trained on paraphrase pairs from the annotated Paraphrase Database (Ganitkevitch et al., 2013). In **InferSent** (Conneau et al., 2017), the model is trained on a natural language inference task, and consists in classifying pairs of sentences from the SNLI annotated dataset (cf. Section 9.5) into *entailment*, *contradiction*, or *neutral*. The encoder produces a vector representation of the premise and the hypothesis sentences of each inference pairs. The vectors are combined with different methods and fed into a feed-forward layer with a softmax classifier. The **Universal Sentence Encoder** (Cer et al., 2018) learns sentence embeddings with Transformers trained using multi-task learning, which includes the SkipThought prediction objective and the InferSent supervised task.

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

Neural models like seq2seq architectures can encode any type of sequence with vectors. Therefore, they clearly improve on other types of compositional DSMs, like for instance the DFM (cf. Section 9.3), which instead focus on specific phrasal types, but strive to scale up to more complex linguistic constructions. This explains the popularity enjoyed by neural sentence embeddings in NLP and AI, which need general methods to represent text structures with vectors to be used in downstream deep learning applications, such as dialogue understanding, question-answering, etc. On the other hand, neural networks often require a lot of time and text to be trained. The question is therefore whether this complexity is worth the effort, and whether neural architectures do provide better representations of sentence meaning.

THE LIMITS OF SENTENCE EMBEDDINGS

Research has shown that this is not always the case and that the much simpler – but theoretically unsatisfactory – additive model (cf. Section 9.2) is highly competitive with respect to neural models in intrinsic and extrinsic tasks (cf. Section 9.5). For instance, Wieting et al. (2016a) represent sentences with the average of embeddings encoding character n-grams, and obtain better results than LSTMs and CNNs. More generally, additive models, possibly improved with more sophisticated vector weighting methods, are able to outperform neural sentence embeddings (Hill et al., 2016; Wieting et al., 2016b; Arora et al., 2017; Shen et al., 2018). This means that the contribution of the neural network combining the word embeddings is actually quite limited. Such a conclusion is consistent with the findings by Conneau et al. (2017, 2018a) that untrained LSTMs encoders using pretrained embeddings behave as well as trained encoders. This fact has been systematically investigated by Wieting and Kiela (2019), who compare the performance of trained seq2seq models with LSTMs with random weights. Surprisingly, the trained models do not significantly outperform the random ones. Wieting and Kiela (2019) conclude that the added value brought by neural sentence embeddings to NLP systems depends more on the quality of the component word vectors rather than on the composition process carried out by the encoder network.

9.5 Evaluation of Compositional DSMs

A crucial question for compositional distributional semantics is how to evaluate its representation of phrase and sentence meaning. In symbolic models, the semantic properties of complex linguistic expressions can be directly “read off” from the formal structures representing them. In fact, the very purpose of semantic analysis is to associate sentences with structures of symbols that make explicit their logical form. Compositional distributional representations are instead radically different. The models we have seen in the previous

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

sections encode complex expressions with vectors whose content is not immediately interpretable. Therefore, evaluation should not only aim at comparing the performances of compositional DSMs, but also understanding the aspects of meaning captured by distributional representations of phrases and sentences.

Like with lexical models, we can distinguish intrinsic and extrinsic evaluation methods (cf. Chapter 7). **Intrisic evaluation** uses tasks and datasets that are specifically designed to test the model's ability to perform semantic composition. One of the most common approaches consists in testing compositional DSMs to predict human ratings of **phrase and sentence similarity**, as the compositional analogue of the task used to evaluate lexical DSMs (cf. Section 7.2.2). The evaluation measure is the **Spearman rank correlation** (ρ) between distributional similarity scores and ratings.

INTRISIC EVALUATION

PHRASE AND SENTENCE SIMILARITY

SPEARMAN CORRELATION

The first generation of benchmarks mainly focuses on fixed and small linguistic constructions. The dataset in Mitchell and Lapata (2008), **ML2008**, is formed by 120 pairs of subject–verb sentences. A reference subject–verb sentence (e.g., *horse ran*) is paired with two "landmark" intransitive verbs (cf. Section 9.6.1), whose meaning is compatible (e.g., *horse galloped*) or incompatible (e.g., *horse dissolved*) to the reference one. The resulting sentence pairs are then rated for similarity. Mitchell and Lapata (2010) introduce a dataset (**ML2010**) consisting of similarity judgments for 324 adjective–noun, noun–noun, and verb–object combinations rated on a $1 - 7$ scale. For example, *vast amount – large quantity*, *marketing director – assistant manager*, and *produce effect – achieve result* have high similarity, while *general level – federal assembly*, *bedroom window – education office*, and *encourage child – leave house* have low similarity. The **GS2011** (Grefenstette and Sadrzadeh, 2011) and **KS2014** (Kartsaklis and Sadrzadeh, 2014) datasets extend the same approach to transitive sentences. The former is inspired by ML2008 and contains similarity ratings for 200 pairs of subject–verb–object sentences differing only for their verb (e.g., *government provide cash – government supply cash*), while KS2014 includes 108 sentence pairs without lexical overlap (e.g., *project present problem – programme face difficulty*).

ML2008

ML2010

GS2011

KS2014

Other benchmarks evaluate compositional DSMs on **cross-level semantic similarity** (Jurgens et al., 2014), which involves comparing linguistic expressions of different lengths, like a single word with a synonym phrase, whose vector is generated by the model. Turney (2012) introduces a dataset (**T2012**) of $2,180$ noun–noun compounds and adjective–noun phrases and frames the task as a multiple-choice synonym test (cf. Section 7.2.1): Given a target phrase called the *stem* (e.g., *dog house*), the task is to select the *solution* synonym word (e.g., *kennel*) out of seven alternatives. **RELPRON** (Rimell et al., 2016) consists of $1,087$ term-property pairs, where the *term* is a noun (e.g., *army*) and the *property* is a noun phrase providing the definition of the term

CROSS-LEVEL SEMANTIC SIMILARITY

T2012

RELPRON

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

Table 9.3 An example of sentences from the SICK dataset

Sentence A	*Sentence B*	*Relatedness score*
A man is talking	A man is speaking	4.9
A baby is crying	A man is exercising	1
Sentence A	*Sentence B*	*Entailment label*
A sad man is crying	A man is crying	ENTAILMENT
A jet is not flying	A jet is flying	CONTRADICTION
A man is sitting in a field	A man is running in a field	NEUTRAL

(e.g., *organization that general commands*). Each term has between four and ten different properties. All properties have the same structure: a noun head, which is a hypernym of the term (e.g., *organization*), and a transitive restrictive relative clause, whose subject or direct object is the term. RELPRON is primarily conceived for a property ranking task. Composed vectors are produced for all properties in the dataset. Given a term, a model must rank properties by their similarity to the term. The ideal system will rank all properties corresponding to that term above the other properties.

SICK

ENTAILMENT RELATIONS

NATURAL LANGUAGE INFERENCE

The **SICK** (Sentences Involving Compositional Knowledge) dataset (Marelli et al., 2014; Bentivogli et al., 2016) has two important elements of novelty with respect to the first-generation benchmarks. Firstly, while those contain small “skeleton” sentences with fixed structure and no function words, SICK consists of real sentences of various complexity and inclusive of all their components (e.g., determiners, prepositions, conjunctions, etc.). Secondly, besides predicting sentence similarity, SICK tests compositional DSMs to recognize **entailment relations** between sentences. In fact, knowing the meaning of a sentence entails understanding which other sentences follow from its truth (e.g., the truth of *John met a dog* entails the truth of *An animal was met by John*). Therefore, assessing a model ability to identify entailments is an important vantage point to evaluate the semantic representations generated by compositional DSMs. SICK contains $9,840$ sentence pairs including different kinds of linguistic phenomena relevant to determine its meaning and inferential potential, such as negation, active/passive alternations, quantifiers, lexical entailments, and so on. As illustrated in Table 9.3, each sentence pair is rated for semantic relatedness on a $1-5$ scale (SICK-R) and labeled with an entailment relation (SICK-E). The labels in SICK-E are those used in the **recognizing textual entailment** task, also known as **natural language inference** (NLI). Given a pair of sentences A (the *premise*) and B (the *hypothesis*), the pair is classified as ENTAILMENT if the hypothesis is true given the premise, CONTRADICTION if the hypothesis is false given the premise, and NEUTRAL if the hypothesis is underdetermined by the premise (cf. Section 7.4.2).

NLI is also one of the most important tasks for the extrinsic evaluation of compositional DSMs. **Extrinsic evaluation** assesses models by using their sentence vectors in downstream applications. Besides NLI, common tasks include paraphrase identification (i.e., given a pair of sentences, classify them as paraphrases or not), question-answering, and sentiment analysis. Since carrying out these tasks presupposes the ability to represent sentence meaning, the improvement achieved by a certain model is then taken as a measure of the quality of its representations. Extrinsic evaluation tasks are also available in general platforms to benchmark sentence vectors, such as **SentEval** (Conneau and Kiela, 2018) and **GLUE** (Wang et al., 2019a).

EXTRINSIC EVALUATION

SENTEVAL

GLUE

Extrinsic methods have become the leading evaluation paradigm for sentence embeddings, as they measure their contribution to NLP systems in application scenarios. On the other hand, the datasets are not designed to investigate compositionality, but rather to test the system's ability to solve natural language understanding problems. The model's mastery of meaning composition is inferred indirectly from its performance in downstream tasks. Unfortunately, sheer extrinsic performance is often unreliable evidence. In fact, existing evaluation datasets contain biases that enable models to achieve high scores based on superficial cues rather than on truly compositional construction of sentence meaning (Ettinger et al., 2018). An interesting example is provided by NLI itself: Gururangan et al. (2018) and Poliak et al. (2018) show that some of the most widely used datasets, such as **SNLI** (Stanford NLI; Bowman et al., 2015) and **MultiNLI** (Multi-Genre NLI Corpus; Williams et al., 2018), contain annotation artifacts that allow models to perform surprisingly well using only the hypothesis, without actually learning the entailment relation. In SICK, 86.4% of CONTRADICTION sentence pairs can be identified only by the presence of negation (Bentivogli et al., 2016). This has prompted the development of challenging benchmarks that include more complex tasks, like **SuperGLUE** (Wang et al., 2019b), or are specifically designed to reduce artifacts in the data, like **SWAG**, a dataset of commonsense inferences (Zellers et al., 2018), and **WinoGrande** (Sakaguchi et al., 2020). The latter extends the Winograd Schema Challenge (Levesque, 2011) and consists of pronoun resolution problems requiring advanced compositional and commonsense reasoning abilities.

SNLI

MULTINLI

SUPERGLUE

SWAG

WINOGRANDE

An alternative approach to sentence vector evaluation is provided by **probing tasks** (Ettinger et al., 2016; Belinkov and Glass, 2019), which have become extremely popular to interpret the information encoded by deep neural networks (cf. Section 8.8.2).

PROBING TASKS

> Given a representation generated by a model (e.g., a sentence embedding), a **probing task** is a classifier trained to predict a specific property based on this representation.

The successful performance of the classifier is then used to deduce that the representations of the model encode that property. Compared with extrinsic evaluation in downstream tasks, probing tasks aim at providing more nuanced analyses with respect to specific linguistic dimensions.

SemRole

edge probing

Several tasks are designed to target syntactic phenomena, such as word order and constituency structure (Adi et al., 2017; Conneau et al., 2018a; Gulordava et al., 2018). Others instead specifically address compositional semantics. The **SemRole** task in Ettinger et al. (2016, 2018) aims at identifying whether sentence embeddings encode information about semantic roles. The classifier receives in input the embedding $\mathbf{n}$ of a probe noun n, the embedding $\mathbf{v}$ of a probe verb v, and the embedding $\mathbf{s}$ of sentence s containing n and v, and must decide whether n has the AGENT role in s or not (e.g., in the sentence *The dog chases the cat*, the classifier should identify the probe *dog* as a positive instance, and the probe *cat* as a negative instance). The classifier accuracy then measures whether the sentence embedding discriminates the semantic roles. Zhu et al. (2018) propose a battery of probing tasks to test argument structure and negation, while Shwartz and Dagan (2019) target phenomena related to figurative meaning and multiword expressions. The **edge probing** suite of tasks introduced by Tenney et al. (2019b) test various syntactic and semantic aspects of sentence structure (e.g., semantic roles, anaphora, etc.).

Probing tasks are a form of "black box" testing, as they provide indirect evidence about the embedding content. Moreover, they crucially depend on the careful and balanced design of training and test sets, which must single out the phenomenon of interest and avoid any confounding factor (Belinkov, 2022). However, probing classifiers represent important tools to explore the semantic properties captured by compositional DSMs.

9.6 Context-Sensitive Distributional Representations

In the previous sections, we have presented the main methods to create and evaluate distributional representations of phrases and sentences. Now, we move to the related question of how DSMs capture the context-sensitive behaviour of lexical meaning. As illustrated in Section 9.1.1, Fregean Compositionality presupposes that word meaning is essentially context-independent, but this

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

Table 9.4 Top: Two items from the WiC dataset. Bottom: lexical substitutes (in parenthesis the number of annotators that selected them) for two occurrences of the adjective *bright* from the SemEval 2007 dataset

Sentence 1	*Sentence 2*	*Same meaning*
The expanded **window** will give us time to catch the thieves.	You have a two-hour **window** of clear weather to finish working on the lawn.	TRUE
There's a lot of trash on the **bed** of the river.	I keep a glass of water next to my **bed** when I sleep	FALSE

Sentence	*Lexical substitutes*
He was **bright** and independent and proud.	intelligent (3); clever (3)
She turns eyes **bright** with excitement.	shining (4); alight (1)

assumption is challenged by the wealth of cases in which lexemes undergo meaning modulation in context, as the verb *run* in the following examples:

(12) a. The horse runs.
b. The water runs.
c. The virus runs.

Symbolic semantic models strive to represent contextual meaning shifts and co-compositonality phenomena, which are reduced to cases of lexical ambiguity (Fodor and Pylyshyn, 1988; Pustejovsky, 1995). The continuous nature of distributional vectors instead promises to provide more satisfactory and explanatory ways to address this phenomenon. In this section, we illustrate current approaches to build **context-sensitive distributional representations** that capture contextualization effects. Meaning modulation is a source of lexical polysemy, which is discussed in Section 8.2, but from a different, albeit related, perspective. There we target the problem of how DSMs represent word senses, while here we focus on the compositional mechanisms that change the meaning of a lexeme depending on its contexts.

CONTEXT-SENSITIVE DISTRIBUTIONAL REPRESENTATIONS

One of the most common tasks to evaluate context-sensitive representations is **word similarity in context**, which consists in modeling human similarity ratings for words presented *in linguistic contexts*, rather than in isolation, as in the standard similarity task (cf. Section 7.2.2). It is thus possible to estimate the effect of context in modulating semantic similarity judgments. The **Stanford Contextual Word Similarity** (SCWS) dataset (Huang et al., 2012) and the **USim** dataset (Erk et al., 2013) respectively contain $2,003$ and $1,142$ pairs of

WORD SIMILARITY IN CONTEXT

SWCS

USIM

WIC

COSIMLEX

words in distinct sentences annotated with graded similarity judgments. The **Words-in-Context** (WiC) dataset (Pilehvar and Camacho-Collados, 2019) includes $7,466$ words, each presented in two different contexts and labeled as to whether it has the same meaning in the two examples or not (see Table 9.4). **CoSimLex** (Armendariz et al., 2020) consists of similarity ratings of 333 word pairs derived from SimLex 999, appearing together in two shared contexts. The dataset is multilingual, as it also contains Croatian, Finnish, and Slovene data.

LEXICAL SUBSTITUTION

SEMEVAL 2007

TWSI

COINCO

Another setting to evaluate contextualized vectors is **lexical substitution**, which is the task of finding a suitable meaning-preserving substitute for a target word in the context of a sentence. The **SemEval 2007** dataset (McCarthy and Navigli, 2009) contains 200 target nouns, verbs, adjectives, and adverbs, each occurring in ten distinct sentential contexts for a total of $2,000$ test items. Five human annotators were asked to provide substitutes for these target words (see Table 9.4). The evaluation of vector contextualization models is framed as a ranking task: The list of substitute words must be ranked so that for each item the suitable paraphrases are ranked higher than the inappropriate ones. For instance, given the context *She turns eyes bright with excitement*, the model should rank *shining* and *alight* higher than *clever* and *intelligent*, as replacements for *bright*. Much larger lexical substitution benchmarks for English are the **Turk Bootstrap Word Sense Inventory** (TWSI) by Biemann (2013), and **CoInCo** (Concepts in Context) by Kremer et al. (2014), which is also proposed as an evaluation dataset for semantic composition in Buljan et al. (2018).

9.6.1 Vector Contextualization

VECTOR CONTEXTUALIZATION

WORD-IN-CONTEXT VECTORS

Standard DSMs represent the meaning of lexical items with type vectors (cf. Section 2.4). These vectors are context-independent, because they "summarize" the whole distributional history of lexical items. A first strategy to obtain context-sensitive semantic representations consists in **contextualizing** the type vector of a target lexeme to create **word-in-context vectors**.

> Given a lexeme l, its type distributional vector $\mathbf{l}$, and a context c, the **word-in-context vector** $\mathbf{l}^{\mathbf{c}}$ represents the meaning of l in c and is obtained by modifying $\mathbf{l}$ with information about c.

In order to capture the meaning shifts resulting from predication, Kintsch (2001) modifies the vector sum of a one-place predicate p and its argument a by adding their mutual **nearest neighbors**:

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

$$\mathbf{p}(\mathbf{a}) = \mathbf{p} + \mathbf{a} + \sum_i \mathbf{n_i}, \tag{9.17}$$

where n is a lexeme selected among the top nearest neighbors of the predicate that are also closest to the argument. For instance, the meaning of *The horse runs* is represented by summing the vector of *run*, the vector of *horse*, and the vectors of some of the nearest neighbors of *run* that are also closest to *horse* (e.g., *jump*, *race*, etc.). The result is to strengthen the features of the predicate that are more appropriate for the argument (e.g., the horse-like aspects of running). Kintsch (2001) evaluates vector contextualization with the **landmark method**: Given a target lexeme l_1 (e.g., *run*) and a context lexeme l_2 (e.g., *horse*), the model must predict the similarity of the word-in-context vector $\mathbf{l_1^{l_2}}$ to the vector of another lexeme l_3 (e.g., *gallop*) called *landmark*.

LANDMARK METHOD

Instead of using nearest neighbors, Erk and Padó (2008) contextualize type vectors by combining them with vectors representing **selectional preferences** (cf. Section 9.7), which they assume to correspond to the "expectations" that words trigger about associated events and their participants (McRae et al., 2005b). The selectional preferences of predicates are determined by their expected arguments (e.g., *horse* for *gallop*). In turn, arguments have **inverse preferences** about their expected predicates (e.g., *gallop* for *horse*). When words are composed together, the mutual selectional preferences affect their meaning. Given a tuple $\langle l_1, r, l_2\rangle$ formed by a lexeme l_1 occurring in the context of l_2 with the syntactic relation r (e.g., $\langle run$, nsubj, $horse\rangle$ derived from *The horse runs*), Erk and Padó (2008) assign to l_1 a type vector $\mathbf{l_1}$ and a word-in-context vector $\mathbf{l_1^{\langle r,l_2\rangle}}$ obtained by combining $\mathbf{l_1}$ via componentwise multiplication with the vector $\mathbf{sp^{\langle r,l_2\rangle}}$ representing the (inverse) selectional preferences of l_2 with respect to the syntactic relation r:[9]

SELECTIONAL PREFERENCES

INVERSE PREFERENCES

$$\mathbf{l_1^{\langle r,l_2\rangle}} = \mathbf{l_1} \odot \mathbf{sp^{\langle r,l_2\rangle}}. \tag{9.18}$$

The vector $\mathbf{sp^{\langle r,l_2\rangle}}$ is the weighted centroid vector of the words l_j co-occurring with l_2 with the relation r, whose co-occurrence frequency $F(\langle l_j, r, l_2\rangle)$ is above the empirically determined threshold θ:

$$\mathbf{sp^{\langle r,l_2\rangle}} = \sum_{j:F(\langle l_j,r,l_2\rangle)>\theta} F(\langle l_j, r, l_2\rangle)\mathbf{l_j}. \tag{9.19}$$

[9] The symmetric procedure is used to assign to l_2 the word-in-context vector $\mathbf{l_2^{\langle l_1,r\rangle}}$, in which the type vector of l_2 is modified according to the (inverse) selectional preferences of l_1.

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

For instance, *run* in the tuple $\langle run$, nsubj, $horse\rangle$ is represented with the word-in-context vector $\mathbf{run}^{\langle \mathbf{nsubj}, \mathbf{horse}\rangle}$, which is obtained by multiplying the type vector $\mathbf{run}$ with $\mathbf{sp}^{\langle \mathbf{nsubj}, \mathbf{horse}\rangle}$, in turn computed as the weighted sum of the vectors of the most frequent verbs having *horse* as subject (e.g., *gallop*, *trot*, etc.). Erk and Padó (2008) evaluate their model with the landmark method applied to the ML2008 dataset (cf. Section 9.5). For each test pair, they create the word-in-context vector of the verb in the reference sentence (e.g., *shoulder slumped*) and compare it with the vector of the verb (e.g., *slouched*) in the other sentence, treated as landmark. The model predictions are then correlated with the human similarity ratings in the dataset.

COMPONENT RE-WEIGHTING

A similar approach is used by Thater et al. (2010), while Thater et al. (2011) contextualize distributional vectors by **re-weighting** their components, based on the similarity with the context. Lexemes are represented with explicit type vectors produced by a syntax-based matrix DSM. The jth component of a target vector $\mathbf{t}$ contains the weight w_{t,c_j} corresponding to the PPMI association score between t and a dependency-typed collocate $c_j = \langle r, l_j\rangle$, such that the lexeme l_j is linked to t by the syntactic relation r (cf. Section 2.2.1). A lexeme l_1 in the tuple $\langle l_1, r', l_2\rangle$ is represented with the word-in-context vector $\mathbf{l_1}^{\langle \mathbf{r}', \mathbf{l_2}\rangle}$ whose components are computed as follows:

$$l_{1_i}^{\langle r', l_2\rangle} = \begin{cases} w_{l_1,c_i}\cos_{\text{sim}}(\mathbf{l_j}, \mathbf{l_2}) & \text{if } c_i = \langle r, l_j\rangle \text{ and } r = r' \\ 0 & \text{otherwise} \end{cases} . \quad (9.20)$$

Contextualization reinforces the components of the vector $\mathbf{l_1}$ that have the same syntactic relation as r' and whose collocate l_j has a high cosine similarity with l_2. For example, in the case of $\langle run$, nsubj, $horse\rangle$, the result of contextualizing $\mathbf{run}$ is to increase the salience of those features of running that are more relevant for animals like horses (e.g., promoting the component $\langle$nsubj, $stallion\rangle$ while demoting $\langle$nsubj, $car\rangle$). These methods are all close variations of the same approach (Dinu et al., 2012), and comparative analyses demonstrate fairly similar performances.

TOPIC MODELS

Other approaches to vector contextualization are based on **Topic Models** (cf. Section 4.4). Given a set $Z = \{z_1, \ldots, z_k\}$ of topics corresponding to latent word senses, Dinu and Lapata (2010) describe each lexeme l_1 with a type vector $\mathbf{l_1} = (p(z_1|l_1), \ldots, p(z_k|l_1))$ containing the probability distribution of l_1 over these senses. Similarly, they represent a lexeme l_1 in the context of a lexeme l_2 with the following word-in-context topic vector:

$$\mathbf{l_1^{l_2}} = (p(z_1|l_1, l_2), \ldots, p(z_k|l_1, l_2)). \quad (9.21)$$

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

The effect of contextualization is thus to modulate the sense distribution of l_1 with respect to the context l_2. Adopting the simplifying assumption that l_1 and l_2 are conditionally independent given the sense z_i, each component $p(z_i|l_1, l_2)$ is computed as follows:

$$p(z_i|l_1, l_2) \approx \frac{p(z_i|l_1)p(l_2|z_i)}{\sum_{j=1}^{k} p(z_j|l_1)p(l_2|z_j)}. \tag{9.22}$$

The parameters of the probabilistic model (i.e., the topics and the distributions $p(z|l_1)$ and $p(l_2|z)$) are learned with LDA from a word-by-word co-occurrence matrix with window-based contexts. Ó Séaghdha and Korhonen (2011, 2014) generalize this model to account for the contextualization effects produced by complex syntactic contexts formed by multiple dependency relations (e.g., in *The dog chases the cat*, the meaning of *chase* is affected both by the subject *dog* and the object *cat*). While Dinu and Lapata (2010) contextualize the latent senses themselves, Van de Cruys et al. (2011) represent lexemes with type vectors containing the probability distribution with respect to syntactic collocates, and use the topics to promote the features in the original vector that are most salient given a particular context. Latent topics are learnt with NMF, which has a probabilistic interpretation similar to LDA (cf. Section 4.4.1).

MULTI-PROTOTYPE DSMs

Multi-prototype DSMs, which represent lexemes with sense vectors (cf. Section 8.2.1), predict word similarity in a context c (Reisinger and Mooney, 2010b; Huang et al., 2012) with the following generalization of Equation 8.1.

$$\text{AvgSimC}(l_1, l_2, c) = \frac{1}{N^2} \sum_{i=1}^{n} \sum_{j=1}^{n} p(\mathbf{c}, \mathbf{s_i}(\mathbf{l_1}))p(\mathbf{c}, \mathbf{s_j}(\mathbf{l_2}))\text{sim}(\mathbf{s_i}(\mathbf{l_1}), \mathbf{s_j}(\mathbf{l_2})), \tag{9.23}$$

where $p(\mathbf{c}, \mathbf{s}(\mathbf{l_1}))$ and $p(\mathbf{c}, \mathbf{s}(\mathbf{l_2}))$ are the probabilities of the context vector $\mathbf{c}$ to belong to the sense cluster $\mathbf{s}(\mathbf{l_1})$ and $\mathbf{s}(\mathbf{l_2})$ (Equation 8.2 can be similarly modified). As word senses are inherently represented as clusters of contexts, contextual meaning modulation is modeled by making the similarity measure context-sensitive, without modifying the representation of the lexemes. The similarity between *run* and *gallop* in the context of *horse* depends on the likelihood of *horse* being one of the contexts defining the senses of *run* and *gallop*.

9.6.2 Exemplar DSMs

EXEMPLAR DSMs

Instead of modifying a context-independent type vector to create word-in-context representations, an alternative strategy to account for contextual meaning shifts is to use **exemplar DSMs**.

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

Exemplar DSMs represent each instance of a word with a distinct **token vector** that encodes aspects of its context.

Although Westera and Boleda (2019) limit the scope of distributional semantics to DSMs that generate type vectors representing context-invariant aspects of meaning, exemplar models are DSMs in every respect: They learn inherently context-sensitive representations that encode the distributional properties of each word token. Their name refers to **exemplar models of concepts** (Murphy, 2002; Bybee, 2010), according to which a concept is a collection of instances of a category (e.g., the different exemplars of dogs we have experienced) stored in the semantic memory, and categorization is similarity computation between a new stimulus and each remembered exemplar. Standard DSMs instead learn from the various occurrences of a lexical item in the training corpus one or more vectors that encode its most prototypical distributional features. Therefore, they share characteristics with **prototype models of concepts**, which assume that humans abstract features across the exemplars of a category and represent a concept with a prototypical representation of the category that is the central tendency of its instances. Categorisation of a new exemplar depends on its similarity to category prototypes.

EXEMPLAR MODELS OF CONCEPTS

PROTOTYPE MODELS OF CONCEPTS

In the exemplar DSM by Erk and Padó (2010), a type lexeme l is represented with the set E_l of its **token vectors**, each built from the content words appearing in the sentence in which the token occurs. For example, the instance of *run* in *The horse runs in the field* is associated with a token vector $\mathbf{run^i}$ containing its co-occurrences with the collocates *horse* and *field*, while the instance of *run* in *The car runs in the freeway* is represented with the token vector $\mathbf{run^j}$ containing its co-occurrences with *car* and *freeway*. The meaning shift of a lexeme l produced by the context c is modeled by assuming that c activates the subset of E_l containing the most similar token vectors to c. The **activation** of the **exemplar set** $\text{act}(E_l, c)$ is defined as follows:

TOKEN VECTORS

ESEMPLAR SET ACTIVATION

$$\text{act}(E_l, c) = \{\mathbf{l^i} \in E_l \mid \cos_{\text{sim}}(\mathbf{l^i}, \mathbf{c}) > \theta\}, \qquad (9.24)$$

where $\mathbf{l^i}$ is the vector of the ith token of l, $\mathbf{c}$ is the token vector of c, and θ is an empirically determined similarity threshold. The similarity in context between the exemplar sets of two lexemes l_1 and l_2 is measured with the cosine similarity between the centroid vectors of $\text{act}(E_{l_1}, c)$ and $\text{act}(E_{l_2}, c)$. Erk and Padó (2010) test their model on a lexical substitution task.

SUBSTITUTE VECTORS

A similar approach is proposed by Melamud et al. (2015), who represent lexemes with sets of **substitute vectors**, each representing a specific context of

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

a lexeme. Substitute vectors are second-order token vectors whose components correspond to a weight estimating how fit a lexical item is to substitute a target word in a certain context. For instance, the substitute vector of *run* in *The horse runs in the field* contains the conditional probability of each word in the model vocabulary given the context *The horse* __ *in the field*. Melamud et al. (2015) define the **out-of-context type vector** $\mathbf{l}$ of a lexeme l as the average of the substitute vectors of its contexts:

OUT-OF-CONTEXT TYPE VECTOR

$$\mathbf{l} = \frac{1}{|C_l|}\sum_{i \in C_l} \mathbf{c_i}, \tag{9.25}$$

where C_l is a collection of the observed contexts for l in the training corpus, and $\mathbf{c}$ are their substitute vectors. The meaning shift of l produced by a context c is then modeled by selecting only the substitute vectors of l that are most similar to the substitute vector of c.

Johns and Jones (2015), Jamieson et al. (2018), and Jones (2019) propose an exemplar DSM based on the **BEAGLE** random encoding model (cf. Section 5.4) and inspired by the **MINERVA 2** multiple-trace model of episodic memory (Hintzman, 1986). Each context c of a lexeme l is represented as the sum of the random vectors of its n words and stored as a distinct **memory trace**:

BEAGLE EXEMPLAR DSM

MINERVA 2

MEMORY TRACE

$$\mathbf{c_l} = \sum_{j=1}^{n} \mathbf{r}(\mathbf{l_j}). \tag{9.26}$$

When the random vector of l is presented to the memory as a probe, it activates a set of traces with a strength proportional to their similarity with the probe vector, like in Erk and Padó (2010). The sum of the activated memory traces, weighted by their activation strength, is called an **echo vector** and corresponds to the out-of-context meaning of l. If l is presented together with another contextual probe c, the echo vector will be formed by the memory traces similar to both l and c, and will correspond to the word-in-context representation of l.

ECHO VECTOR

9.6.3 Contextual DSMs

The most recent and popular kind of context-sensitive distributional representation are **contextual embeddings** learned with deep neural networks (Liu et al., 2020). By contrast, standard type vectors are now referred to as **static embeddings**. **Contextual DSMs** represent each word token in a linguistic sequence with a contextual embedding that is a function of the whole input sequence.

CONTEXTUAL EMBEDDINGS

STATIC EMBEDDINGS

CONTEXTUAL DSMS

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

Given a sequence of lexemes $s = l_1, \dots, l_n$, a **contextual DSM** assigns to each token l_i the **contextual embedding** $\mathbf{h}_{\mathbf{l_i}}$, such that $\mathbf{h}_{\mathbf{l_i}} = f(\mathbf{l_n}, \dots, \mathbf{l_1})$, where $\mathbf{l_i}$ is a non-contextual (static) embedding of l_i.

NEURAL LANGUAGE MODEL

The function f is typically a deep **neural language model** trained with self-supervised learning (cf. Section 6.2), and the contextual embeddings $\mathbf{h_l}$ correspond to its **hidden vectors**. The idea that the hidden states of networks trained to predict linguistic sequences encode context-sensitive lexical representations is already present in Elman (2011). He shows that the hidden vector generated by an SRN for *cut* in *The butcher cuts the meat* is different from the vector generated for the same verb in *The lumberjack cuts the meat*. In fact, the internal states of recurrent networks depend on previous states and therefore capture relevant aspects of a word context. Current contextual DSMs are based on the same assumption, but learn context-sensitive embeddings with much more complex neural architectures.

Recurrent Models: ELMo

CoVE

A first group of contextual DSMs is based on recurrent networks. The **Contextual Vector** (CoVe) model (McCann et al., 2017) is an encoder-decoder architecture (cf. Section 9.4.3) with two layers of bidirectional LSTMs (biLSTMs).

BiLSTM

A **biLSTM** (Graves and Schmidhuber, 2005) is formed by two networks: (i) a forward LSTM that reads the input sequence from left to right and for each lexeme l_i in the input sequence $s = l_1, \dots, l_n$ produces the hidden representation $\overrightarrow{\mathbf{h_i}}$; (ii) a backward LSTM that reads s from right to left and produces the hidden vector $\overleftarrow{\mathbf{h_i}}$. The hidden state of the biLSTM $\mathbf{h_i}$ for the lexeme l_i is the concatenated vector $[\overrightarrow{\mathbf{h_i}}; \overleftarrow{\mathbf{h_i}}]$. In CoVe, the hidden vector generated by the first LSTM layer is then fed into the second LSTM, whose hidden layer is the contextual embedding of l_i. The biLSTM is trained with a machine translation task and the input embeddings are GloVe vectors. McCann et al. (2017) apply the CoVe embeddings to various NLP downstream tasks. Generally, CoVe outperforms context-independent pretrained vectors, thereby showing the positive effects of contextualization.

ELMo

A more effective model is **ELMo** (Embeddings from Language Models; Peters et al., 2018), which learns contextual embeddings with a two-layer biLSTM architecture very similar to CoVe, except for three major differences: (i) the encoder-decoder model is trained with the self-supervised language modeling task (Peters et al., 2017); (ii) the LSTMs are fed with word embeddings generated by a CNN input layer operating on character embeddings (Kim et al., 2016), to encode sub-word information and to represent out-of-vocabulary

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

words unseen during training; (iii) the contextual embedding of l_i is the weighted average of its representations in the network layers:

$$\mathbf{ELMo_i} = \gamma \sum_{j=0}^{k} w_j \mathbf{h_i^j}, \tag{9.27}$$

where k is the number of ELMo layers, $\mathbf{h_i^j}$ is the vector representation of l_i produced by the jth layer, and γ and w_j are parameters optimized in the tasks in which the ELMo vectors are employed as pretrained representations.

Since $j = 0$ is the input layer, $\mathbf{h_i^0}$ is a static embedding for l_i. Each $\mathbf{h_i^j}$, with $j > 0$, is a contextual embedding, because it encodes information coming from the word context. Peters et al. (2018) show that the different layers of the biLSTM encode distinct aspects of the context. Lower layers tend to capture morphosyntactic information, while semantic information is better represented at higher layers. Pooling the various hidden vectors with Equation 9.27 is reported to produce better representations, when used as pretrained features in downstream tasks. In general, ELMo is able to achieve significant improvements over CoVe, a result that supports the beneficial effect of language modeling as training task.

Transformer Models: BERT and GPT

Further advancements in learning contextual embeddings have been obtained by using Transformers instead of recurrent networks. The **Transformer** is an encoder-decoder architecture proposed by Vaswani et al. (2017) for machine translation and based on the mechanism of **attention** (Bahdanau et al., 2015).

TRANSFORMER

ATTENTION

In the standard seq2seq model, the input sequence is represented by a single sentence embedding corresponding to the last hidden state of the encoder, which is then employed by the decoder to generate the output sequence (cf. Section 9.4.3). In attention-based architectures, the input is encoded as a sequence of hidden vectors $\mathbf{h_1}, \ldots, \mathbf{h_n}$, and the decoder has an attention mechanism that allows it to focus on specific parts of the input sequence to generate the output. Attention consists of weight vectors associated with each decoding step. At step j, the input embeddings are multiplied by the **attention vector** $\mathbf{a_j}$ (whose components are all positive and sum to one) and added to obtain a context vector $\mathbf{c_j}$, which is then fed into the decoder:

ATTENTION VECTOR

$$\mathbf{c_j} = \sum_{i=1}^{n} a_{j_i} \mathbf{h_i}. \tag{9.28}$$

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

CONTEXT VECTOR

The **context vector** $\mathbf{c_j}$ thus modulates the contribution of the input embeddings to produce the jth output word. The attention vectors vary from step to step, making the network shift its focus as well. The attention weights are trained with the rest of the network parameters, so that the decoder learns to “attend” to those aspects of the input that are most important for each generation step.

The main novelty of the Transformer consists in dispensing with recurrent networks and using attention as the only mechanism to carry out the seq2seq mapping. While recurrent networks process the input sequentially (i.e., each word internal representation depends on the hidden states of the previous words), the key property of the Transformer is that the input words are processed *in parallel*. This allows the network to achieve a better encoding of long-distance dependencies. As the Transformer does not have any recurrent structure, it needs to have information about the position of each word in the input sequence. This is achieved by using **positional embeddings** that encode the absolute position of input elements (Gehring et al., 2017). Let $s_i = l_1, \ldots, l_n$ be the input sequence and $\mathbf{l_i}$ the m-dimensional embedding of the ith lexeme ($m = 512$ in Vaswani et al., 2017) contained in an input embedding matrix (cf. Section 6.2.2). The Transformer input vector of l_i is the embedding $\mathbf{e_i}$, such that:

POSITIONAL EMBEDDINGS

$$\mathbf{e_i} = \mathbf{l_i} + \mathbf{p_i}, \tag{9.29}$$

where $\mathbf{p_i}$ is the m-dimensional embedding for the ith position. Thus, the two tokens of *dog* in *The brown dog chases the white dog* are represented by different input vectors that give to the network information about their positions. Position embeddings can be either learned (Gehring et al., 2017) or created with a function that maps positions to distinct vectors (Vaswani et al., 2017).

TRANSFORMER BLOCKS

MULTI-HEAD SELF-ATTENTION

The Transformer encoder and decoder are formed by a stack of n layers L called **blocks**, in turn composed of sublayers. The input of the block L_i is the output of the block L_{i-1}. The structure of an encoder block is illustrated in Figure 9.8. Its heart is represented by the **multi-head self-attention** layer, which allows every word to differentially “attend” to all the words in the same sentence, including itself. Self-attention is the method the Transformer uses to encode information about other relevant words into the one that it is currently processing, which is then represented as a weighted sum of its context word vectors. Let $\mathbf{x_1}, \ldots, \mathbf{x_n}$ be the sequence of m-dimensional vectors received in input by the block L_j, and corresponding to the sentence $s_i = l_1, \ldots, l_n$. The multi-head self-attention layer of L_j produces a new sequence $\mathbf{y_1}, \ldots, \mathbf{y_n}$ of m-dimensional vectors. Each $\mathbf{y_i}$ is generated in the following way:[10]

[10] Like in Chapter 6, we assume that $\mathbf{x_1}, \ldots, \mathbf{x_n}$ are row vectors.

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

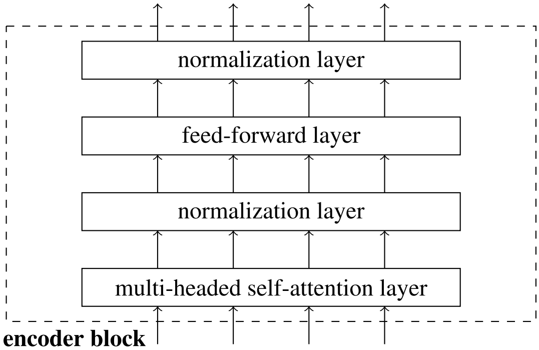


Figure 9.8 Transformer encoder block

1. every input vector $\mathbf{x_j}$, with $1 \leq j \leq n$, is turned into three vectors: a k-dimensional **query** vector ($\mathbf{q_j}$), a k-dimensional **key** vector ($\mathbf{k_j}$), and a v-dimensional **value** vector ($\mathbf{v_j}$), possibly with $k = v$. This is obtained by multiplying $\mathbf{x_j}$ by three distinct weight matrices, $\mathbf{W_Q}$, $\mathbf{W_K}$, and $\mathbf{W_V}$: QUERY, KEY, AND VALUE VECTORS

$$\mathbf{q_j} = \mathbf{x_j}\mathbf{W_Q} \quad \mathbf{k_j} = \mathbf{x_j}\mathbf{W_K} \quad \mathbf{v_j} = \mathbf{x_j}\mathbf{W_V} \tag{9.30}$$

2. the weight $w_{i,j}$ is computed for each word pair $\langle l_i, l_j \rangle$:

$$w_{i,j} = \text{softmax}\left(\frac{\mathbf{q_i} \cdot \mathbf{k_j}}{\sqrt{k}}\right), \tag{9.31}$$

where $\sqrt{k}$ is a scaling value, and the softmax function normalizes the scores. The weight $w_{i,j}$ measures the attention level of the word l_i for l_j as a function of the dot product of their query and key vectors;

3. the **self-attention vector** $\mathbf{a_i}$ is computed as follows: SELF-ATTENTION VECTOR

$$\mathbf{a_i} = \sum_{j=1}^{n} w_{i,j}\mathbf{v_j} \tag{9.32}$$

The self-attention vector is the weighted sum of the value vectors of the other words in the sequence s. This is very similar to the attention mechanism in Equation 9.28;

4. self-attention is multi-headed and consists of h **heads** each corresponding to a function that carries out the three steps above with separate parameter matrices. The result is the sequence $\mathbf{a_i^1}, \ldots, \mathbf{a_i}^{\mathbf{h}}$ of self-attention vectors. ATTENTION HEADS

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

The multi-headed mechanism allows each lexeme to focus simultaneously to more context words and therefore expands the network attention ability;

5. the vectors $\mathbf{a_i^1}, \ldots, \mathbf{a_i}^{\mathbf{h}}$ are concatenated and then multiplied by the weight matrix $\mathbf{W_O}$ to produce the final vector $\mathbf{y_i}$:

$$\mathbf{y_i} = [\mathbf{a_i^1}; \ldots; \mathbf{a_i}^{\mathbf{h}}]\mathbf{W_O}. \tag{9.33}$$

The vectors $\mathbf{y_1}, \ldots, \mathbf{y_n}$ are then fed into two regularization layers (to improve the network accuracy) and to a fully connected feed-forward layer that produces a new vector sequence representing the output of the block L_j. The decoder blocks that generate the output sequence $s_j = l_{n+1}, \ldots, l_z$ have a similar structure, except for two aspects: (i) they have an extra layer that operates multi-head attention over the vectors produced by the encoder blocks; (ii) the self-attention sub-layer is unidirectional and operates left-to-right. This is achieved by "masking" the positions following any output lexeme that is being processed, so that only the positions to its left can be attended to.

The cornerstone of the Transformer architecture is the self-attention mechanism whose weights are computed with the dot product that expresses how related two vectors in the input sequence are. The training task allows the network to learn which words are most related to the one that is being processed and to encode in the output vectors relevant aspects of the context structure. The output vectors generated by each layer are weighted sums over the whole input sequence, and therefore are inherently context-sensitive.

TRANSFORMER CONTEXTUAL EMBEDDINGS

> Given a Transformer with n layers, $L_1, \ldots, L_n$, the vectors $\mathbf{h_1^i}, \ldots, \mathbf{h_n^i}$ generated by each layer L_i are **contextual embeddings** of the input words.

In fact, the Transformer has become the "engine" of several contextual DSMs.

GPT

UNIDIRECTIONAL LANGUAGE MODEL

AUTOREGRESSIVE LANGUAGE MODEL

The various versions of **GPT** (Generative Pre-trained Transformer; Radford et al., 2018, 2019; Brown et al., 2020) use the decoder stack of the Transformer trained on a standard **unidirectional language modeling** task, whose input units are subwords generated through Byte Pair Encoding (Sennrich et al., 2016). Unidirectional language models are also called **autoregressive** or **causal**.

> Given a sequence $s = l_1, \ldots, l_n$, an **autoregressive (unidirectional, causal) language model** is trained to predict the probability of a target word $t = l_i$ by computing its probability either given the preceding context, $p(t|c = l_1, \ldots, l_{i-1})$, or given the following context, $p(t|c = l_{i+1}, \ldots, l_n)$.

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

GPT is an autoregressive model with left-to-right self-attention. Therefore, its embeddings are contextualized only with respect to the preceding context.

BERT

BIDIRECTIONAL LANGUAGE MODEL

MASKED LANGUAGE MODEL

DENOISING AUTOENCODERS

Differently from GPT, **BERT** (Bidirectional Encoder Representations from Transformers; Devlin et al., 2019) employs the Transformer encoder stack and is fully **bidirectional**. The autoregressive language modeling task cannot be used to train a bidirectional self-attention model, since each word would be allowed to indirectly "see itself," and the model could eventually predict the target word trivially. The solution adopted with BERT consists in training it with a **masked language modeling** task, inspired to the Cloze test in psychology: Some of the tokens from the input are randomly masked, and the training objective is to predict the original masked words based on their context. This training task is an example of **denoising autoencoding**.[11]

> **Denoising autoencoders** are neural networks trained to reconstruct an input text where a random subset of the words has been masked out.

NEXT-SENTENCE PREDICTION

BERT is also trained with an additional task of **next-sentence prediction**: Given a sentence pair, the network must predict whether the latter is the next sentence that actually follows the former. This allows BERT to learn relations between sentences, which are useful in applications like question-answering and natural language inference.

The BERT input is a sequence formed by a single sentence or by a pair of sentences A and B, separated by the special token `[SEP]` (see Figure 9.9). The first token of the input sequence is the special `[CLS]` token. The BERT input units are a combination of word, subword and character embeddings, to deal with out-of-vocabulary lexemes. If a whole word is not in the vocabulary, BERT tries to tokenize it into the largest possible subwords contained in the vocabulary, and as a last resort will decompose the word into individual characters. The subword units are learned from the training corpus with WordPiece (Wu et al., 2016). Each ith input token is represented with the embedding $\mathbf{e_i}$, obtained by summing a token embedding, a positional embedding (like in Equation 9.29), and a segment embedding specifying whether the unit belongs to sentence A or B. These embeddings are learned together with the other Transformer parameters. During training, 15% of all the tokens are chosen for the random masking procedure: 80% of the selected units are replaced by the special `[MASK]` token, 10% are replaced with a random word, and 10% are left unchanged. The vectors corresponding to these tokens produced by the final layer are converted into probabilities by first applying a non-linear

[11] Denoising autoencoders do not explicitly estimate the probability distribution of a target given its context, so they are not language models in the strictest sense of this term.

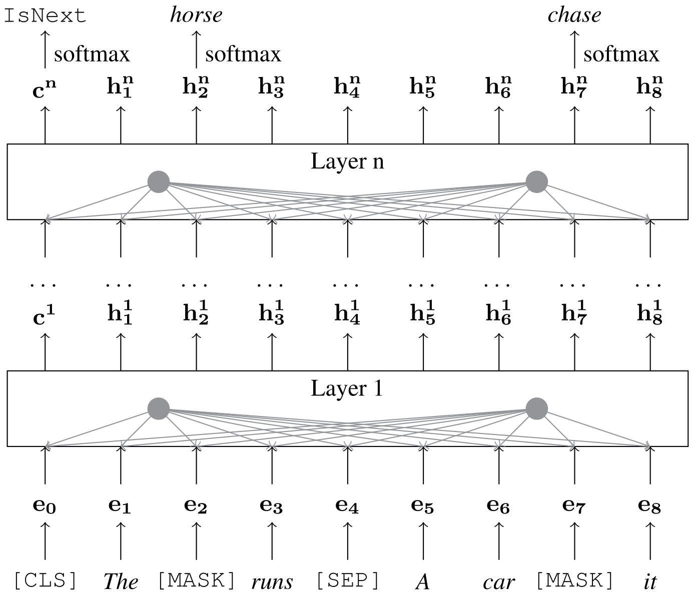

IsNext
horse
chase
softmax
softmax
softmax
$\mathbf{c^n}$ $\mathbf{h_1^n}$ $\mathbf{h_2^n}$ $\mathbf{h_3^n}$ $\mathbf{h_4^n}$ $\mathbf{h_5^n}$ $\mathbf{h_6^n}$ $\mathbf{h_7^n}$ $\mathbf{h_8^n}$
Layer n
...
$\mathbf{c^1}$ $\mathbf{h_1^1}$ $\mathbf{h_2^1}$ $\mathbf{h_3^1}$ $\mathbf{h_4^1}$ $\mathbf{h_5^1}$ $\mathbf{h_6^1}$ $\mathbf{h_7^1}$ $\mathbf{h_8^1}$
Layer 1
$\mathbf{e_0}$ $\mathbf{e_1}$ $\mathbf{e_2}$ $\mathbf{e_3}$ $\mathbf{e_4}$ $\mathbf{e_5}$ $\mathbf{e_6}$ $\mathbf{e_7}$ $\mathbf{e_8}$
[CLS] The [MASK] runs [SEP] A car [MASK] it

Figure 9.9 The structure of BERT. During training, some random items in the input sequence are masked, and the objective is to predict the corresponding words. The gray dots represent the bidirectional self-attention heads

transformation, then a linear one whose weight matrix is the transpose of the input embedding matrix of the Transformer. The result is fed into a softmax function over the vocabulary, to predict the correct lexeme. Like in Vaswani et al. (2017), the weights of the BERT input and output matrices are shared, a solution called **weight-tying trick** (Press and Wolf, 2017), which improves the performance of neural language models and reduces the number of their parameters. The hidden state $\mathbf{c^i}$ produced by the layer L_i for the token [CLS] is the aggregate embedding of the whole input sequence and is used for the next-sentence prediction. This is a binary classification task, in which the vector $\mathbf{c^n}$ of the last layer is assigned the label IsNext, if the sentence B actually follows A, and the label NotNext, otherwise.

WEIGHT-TYING TRICK

Thanks to the self-attention mechanism of the Transformer encoder, BERT is "deeply bidirectional" (Devlin et al., 2019). This is a major difference with respect to ELMo too, which instead just concatenates two unidirectional representations. The original BERT model comes into two variants: (i) **BERT$_{\textbf{base}}$**, with 12 layers and vectors of 768 dimensions; (ii) **BERT$_{\textbf{large}}$**, with 24 layers and vectors of $1,024$ dimensions. Table 9.5 shows how the vectors generated by the last layer of BERT$_{\text{base}}$ capture co-compositionality effects

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

Table 9.5 Cosine similarity between the contextual embeddings of the target lexemes in boldface, generated by the layer 12 of BERT$_{\text{base}}$

	The professor **opened** the conference.
The professor **began** the conference.	**0.81**
The professor **unlocked** the door.	0.57
	The professor **opened** the door.
The professor **began** the conference.	0.53
The professor **unlocked** the door.	**0.77**
	The horse **runs** fast.
The horse **gallops** fast.	**0.73**
The water **flows** fast.	0.70
	The water **runs** fast.
The horse **gallops** fast.	0.54
The water **flows** fast.	**0.85**

(cf. Section 9.1.1). For instance, the vector **run** in *The horse runs* is closer to **gallops** than to **flows**, and vice versa for **run** in *The water runs*.

The Success of Contextual Embeddings

Contextual DSMs represent a very different approach to generate distributional representations with respect to static ones.

STATIC AND CONTEXTUAL DSMs

> **Static DSMs** produce a single distributional vector for each word type in the model vocabulary. **Contextual DSMs** take in input a whole sentence and generate embeddings for each of its word tokens.

Since they generate inherently context-sensitive token vectors through a language modeling training objective, contextual DSMs are **exemplar predict models**. However, generating contextual embeddings is not the end goal of systems like GPT or BERT, which are designed chiefly as general, multi-task architectures to develop NLP applications based on **fine-tuning** (Dai and Le, 2015; Radford et al., 2018; Devlin et al., 2019). This is a method of **transfer learning** that represents a different way of using word embeddings in downstream tasks. The classical approach based on **pretrained features** (cf. Section 7.3) consists in having two separate models: the first one (e.g., word2vec) learns the embeddings that are then fed as features into a distinct task-specific model. In fine-tuning, it is instead the same architecture that deals with the whole process in two steps:

EXEMPLAR PREDICT DSMs

FINE-TUNING

TRANSFER LEARNING

PRETRAINED FEATURES

pretraining – the model is first trained with a general unsupervised or self-supervised task (i.e., language modeling) on large amounts of texts;

fine-tuning – the pretrained model is then re-trained on a specific supervised task (e.g., question-answering), by replacing the softmax layer employed in the former phase with another task-specific softmax classifier.

During fine-tuning, the model leverages the linguistic knowledge encoded in the pretrained distributional representations, and requires a smaller amount of annotated data to learn the final task. Devlin et al. (2019) show that BERT can achieve state-of-art results in several NLP tasks using either the feature-based or the fine-tuning method. The success of BERT has sparked the development of several improved models, such as **RoBERTa** (Liu et al., 2019), **XLNet** (Yang et al., 2019), **ELECTRA** (Clark et al., 2020), **DeBERTa** (He et al., 2021), among many others (Han et al., 2021).

CROSS-LINGUAL CONTEXTUAL EMBEDDINGS

MULTILINGUAL MASKED LANGUAGE MODELING

BERT is trained on monolingual textual data, but the same approach has been extended to the multilingual and multimodal settings as well. **mBERT** (Devlin et al., 2019) and **XLM** (Conneau and Lample, 2019) are multilingual versions of BERT that learn **cross-lingual contextual embeddings** (cf. Section 8.4) using concatenated monolingual data from 100 languages. The models have a shared vocabulary across languages and are trained with **multilingual masked language modeling**: The model must reconstruct masked words that belong to multiple languages. Both mBERT and XLM obtain strong performances on zero-shot cross-lingual transfer in downstream NLP tasks, proving that they can learn regularities among languages even without parallel data. This is also confirmed by Conneau et al. (2020), who show that similar embedding spaces emerge from monolingual BERT models, whose contextual representations in different languages can be aligned in a joint space by using a linear mapping (cf. Section 8.4.1).

MULTIMODAL CONTEXTUAL EMBEDDINGS

MASKED MULTIMODAL LEARNING

MULTIMODAL ALIGNMENT PREDICTION

CLIP

Other language models learn **multimodal contextual embeddings** from texts and images (cf. Section 8.7.1). **ViLBERT** (Lu et al., 2019) consists of two parallel Transformer streams for visual and linguistic processing that are then fused through co-attentional layers attending in each stream to information in the other one. ViLBERT is trained on a corpus of image-caption pairs with two tasks. In **masked multimodal learning**, the model must reconstruct masked regions (i.e., whose features are zeroed) from images and masked words from their caption. In **multimodal alignment prediction**, the model must predict whether the text is a correct description of the image. ViLBERT learns multimodal representations that achieve state-of-the art performances in tasks such as visual question answering or grounding referring expressions. **CLIP**

(Contrastive Language-Image Pre-training; Radford et al., 2021) is a multimodal extension of GPT which jointly trains an image and a text encoder to predict the correct pairings of a set of image-text training examples.

The improvements achieved by deep neural language models in several tasks have granted huge popularity to contextual embeddings, which have fast replaced static ones, especially in downstream applications. In fact, contextual embeddings are often preferred to sentence embeddings too (cf. Section 9.4), and it has become customary to represent sentences with the sequence or the average of the contextualized vectors of their component words.[12] The reason for this success is ascribed to the ability of deep neural language models to generate lexical representations that capture context-dependent aspects of word meaning, thereby overcoming the meaning conflation deficiency of static embeddings (cf. Section 8.2). Several studies aiming at exploring the "geometry" of contextual semantic spaces show that they encode a fine-grained representation of polysemy and word senses (Coenen et al., 2019; Wiedemann et al., 2019; Garí Soler and Apidianaki, 2021). Ethayarajh (2019) introduces the following **self-similarity** measure to estimate the degree of contextualization of the n token embeddings $\mathbf{l_1^L}, \ldots, \mathbf{l_n^L}$ of a lexeme type l produced by the layer L:

SELF-SIMILARITY

$$\text{SelfSim}_L(l) = \frac{1}{n^2 - n} \sum_{i=1}^{n} \sum_{j \neq i}^{n} \cos_{\text{sim}}(\mathbf{l_i^L}, \mathbf{l_j^L}). \qquad (9.34)$$

The self-similarity of l is the average cosine similarity between its contextualized token vectors, and measures the uniformity of the semantic space of l in the different contexts: The higher the self-similarity of l, the less contextualized are its token embeddings. Ethayarajh (2019) reports that in ELMo, BERT, and GPT the higher layers produce more context-specific embeddings than lower ones, though with important differences depending on the model. Garí Soler and Apidianaki (2021) use the same measure to show that BERT discriminates between monosemous and polysemous lexemes.

Contextual DSMs only generate embeddings of word tokens, as a function of the sequences in which they occur. This is consistent with exemplar semantic models, which assume there is no such thing as a context-independent lexical meaning (Bybee, 2010; Jones, 2019). The latter is simply viewed as a

[12] In BERT, the final hidden state for the `[CLS]` is a sentence embedding that is close to the weighted average of its word embeddings. Reimers and Gurevych (2019) introduce **Sentence-BERT**, which learns sentence embeddings by fine-tuning BERT on natural language inference data.

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

TYPE-LEVEL LEXICAL MEANING

TYPE CONTEXTUAL EMBEDDING

second-order entity corresponding to a set of exemplar meanings (cf. Section 9.6.2). On the other hand, several semantic tasks concern out-of-context words and it is therefore important to evaluate to what extent contextual DSMs are also able to capture **type-level** aspects of lexical knowledge. Although context modulates word meaning, several properties also remain constant across contexts: The fact that *dog* is more similar to *cat* than to *strawberry* is a type-level semantic property which is essentially context-independent. Contextual DSMs should be able to learn embeddings that preserve these type-level properties, beyond the contextual modulation encoded in each token vector. Mickus et al. (2020) show that BERT token embeddings indeed form coherent clusters when grouped by lexical types. Other studies compare the type embeddings produced by static DSMs with those obtained by pooling token contextual vectors. The simplest approach consists in defining the **type contextual embedding** $\mathbf{l}$ as the average of the token embeddings obtained from a set S of n randomly selected sentences in which l occurs (Bommasani et al., 2020):

$$\mathbf{l^L} = \frac{1}{|S|}\sum_{i \in S} \mathbf{l_i^L}, \tag{9.35}$$

where $\mathbf{l_i^L}$ is the embedding of the ith token of the lexeme l generated by the layer L of the model (it is also customary to average embeddings across multiple layers, as this can improve the representations). When tested on some of the tasks and datasets used for the intrinsic evaluation of standard DSMs (cf. Section 7.2), Bommasani et al. (2020) and Vulić et al. (2020) report that type contextual embeddings are competitive with static ones, even if they are generated with a small number of contexts (e.g., $|S| = 10$). On the other hand, Lenci et al. (2022) compare BERT type embeddings with those produced by the DSMs in Section 7.4.1, and show that static embeddings significantly outperform the type vectors derived from BERT in almost all tasks. Therefore, they argue that, when properly optimized, static DSMs still have an edge in out-of-context semantic tasks, though the performance of contextual DSMs might be improved by adopting more complex methods to build type embeddings. For instance, Chronis and Erk (2020) propose a multi-prototype model to generate sense vectors by clustering BERT token vectors (cf. Section 8.2.1).

(AN)ISOTROPY

The ability of contextual DSMs to capture semantic knowledge is also affected by the strong **anisotropy** of their embedding space (Ethayarajh, 2019).

> A vector space is **isotropic** if the vectors are distributed uniformly.

Perfect isotropy entails that the average cosine similarity between randomly sampled words would be zero (Arora et al., 2016). The more anisotropic the space is, the more its vectors are instead distributed in a narrow cone, in which even unrelated words have high cosine similarities. A high degree of isotropy is a desirable property, because it means a better ability of word embeddings to capture similarity relations. Mimno and Thompson (2017) and Mu and Viswanath (2018) notice the anisotropy of the word2vec and GloVe semantic spaces, but Ethayarajh (2019) shows that this phenomenon is particularly strong in BERT and GPT, thereby producing unwanted anomalies in the semantic space. For instance, Rajaee and Pilehvar (2021) report that the representations for different senses of a verb tend to be closer than the embeddings for the same sense that correspond to different verb tenses. Gao et al. (2019) relate the high degree of anisotropy of contextual DSMs – which they call **representation degeneration problem** – to the fact that they are neural language models trained to optimize the likelihood of the words to be predicted using the weight-tying trick. This would make some dimensions highly dominant in the embeddings, thereby grouping them in a narrow area. Both Cai et al. (2021) and Rajaee and Pilehvar (2021) show that the phenomenon of anisotropy is less pronounced in local clusters of lexical items, which would explain the good performance of contextual embeddings anyway. Some common solutions to smooth anisotropy is to standardize the embeddings by turning their components into z-scores, or to remove the top principal components of the embeddings identified with PCA (Mu and Viswanath, 2018; Timkey and Van Schijndel, 2021). This latter solution is similar to the case of inverse truncated SVD in count models (cf. Section 2.5.1), in which discarding the top singular values results into better distributional representations.

REPRESENTATION DEGENERATION PROBLEM

A major drawback of Transformer contextual DSMs resides in their huge **complexity**. BERT$_{\text{large}}$ has 340 million parameters, some of the later models have hundreds of billions of parameters and are trained on huge amounts of text. Such complexity causes excessive costs to learn the embeddings and greatly limits their sustainability (Strubell et al., 2019). This has prompted several attempts to develop lighter models that aim at preserving the original quality of the representations while reducing the parameters or the training data, such as **DistilBERT** (Sanh et al., 2019) and **BabyBERTa** (Huebner et al., 2021). The latter is trained with just five million words of child-directed speech, to match the quantity and quality of the input children are exposed to. Zhang et al. (2021) instead explore the relationship between the amount of training data and the type of linguistic knowledge that BERT is able to learn, and show that the model performance in syntactic and semantic tasks peaks after only ten to one hundred million words.

COMPLEXITY

DistilBERT

BabyBERTa

The complexity of contextual DSMs also decreases the **interpretability** of their representations, thereby exacerbating a problem that affects all neural

INTERPRETABILITY

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

BERTology

architectures. Some works attempt to explore the attention mechanism to identify the contextual elements the model attends to (Clark et al., 2019; Kovaleva et al., 2019). However, the overall picture is still fairly unclear, proving the difficulty to get at a full understanding of the actual model behavior. While static DSMs aim at representing type-level aspects of lexical meaning (e.g., similarity relations), the goal of contextual models like BERT is to produce representations that encode large amounts of linguistic information, including but not limited to semantic ones, which can be leveraged in downstream tasks. A new research trend, which Rogers et al. (2020) call **BERTology**, aims at investigating the nature and extent of this linguistic knowledge, mostly using probing tasks. For example, Tenney et al. (2019a) show that BERT lower layers tend to capture morphosyntactic information, while higher layers focus on semantic dimensions, similarly to what happens in ELMo. At the same time, Tenney et al. (2019b) report that, though contextual embeddings improve over static ones, this mainly concerns syntactic tasks (e.g., constituent labeling) rather than semantic ones (e.g., coreference), suggesting that these embeddings encode more syntactic than semantic information.

prompting

oLMpics

Another method to explore the knowledge encoded by contextual DSMs is **prompting** (Liu et al., 2021). Instead of fine-tuning the models and using auxiliary supervised classifiers, propting is based on an unsupervised zero-shot setting. A *prompt* is fed to the language model, which either predicts the continuation of the prompt (with autoregressive models like GPT) or the identity of one or more masked words (with bidirectional ones like BERT). The output of the language model is then used as the response to the task at hand. Ettinger (2020) introduces a suite of tasks derived from psycholinguistic research to test BERT semantic knowledge by looking at its word predictions. Prompts are formed by a context plus a `[MASK]` token (e.g., *A robin is a* `[MASK]`.) in the position of interest, and evaluation measures the prediction accuracy for this `[MASK]` token. The results of her experiments show that BERT is sensible to certain aspects of semantic roles (e.g., *The waiter served the customer* vs. *The customer served the waiter*), but in general it has limited knowledge of event inference and negation, as also confirmed by Staliūnaitė and Iacobacci (2020). The **oLMpics** test suite by Talmor et al. (2020) is structured in the same fashion and reveals the models have in many cases poor reasoning abilities. Ravichander et al. (2020) and Hanna and Mareček (2021) show with prompting the difficulty of contextual DSMs to capture hypernymy (cf. Section 8.3.1).

In summary, the huge popularity gained by contextual DSMs is surely justified by the fact that they provide distributional representations of lexical items in their sentential contexts, thereby capturing context-sensitive dimensions of meaning, alongside other linguistic information that is crucial to improve the

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

performances of NLP and AI systems. On the other hand, such benefits should not make us underestimate the limits of these models, whose real semantic competence is still an open research question.

9.7 Distributional Models of Selectional Preferences

SELECTIONAL PREFERENCES

Selectional preferences are the constraints that govern semantic composition (cf. Section 9.1). In symbolic models, they are represented with symbols drawn from a stipulated ontology of semantic types that should characterize the necessary and sufficient conditions for a lexeme to be the argument of a predicate. However, predicates greatly differ for the granularity of argument types they admit, and in several cases it is impossible to find an independently motivated ontological class that capture their semantic constraints (Zeldes, 2013). For instance, the verb *achieve* admits as direct objects nouns like *result*, *aim*, *goal*, *success*, and so on. In this case, the only possible way to group the selected entities into a categorical type is by referring to the predicate itself (e.g., *achieve* selects for ENTITIES_THAT_CAN_BE_ACHIEVED), thereby making the identification of semantic types dangerously circular.

Asher (2011) correctly points out that semantic types are "mind-dependent representations" that are part of the "conceptual apparatus necessary for linguistic understanding" (p. 37). Like other kinds of conceptual representations, semantic constraints show typicality effects (Murphy, 2002): The possible arguments of a predicate are not all equally favored. For instance, the verb *arrest* prefers animate direct objects. However, though *arrest a policeman* and *arrest a thief* are both well-formed, *thief* is a more prototypical patient of *arrest* than *policeman*. In psycholinguistics, the typicality of an argument with respect to a predicate is called **thematic fit** (McRae et al., 1997b, 1998).

THEMATIC FIT

> The **thematic fit** is a word plausibility as the argument of a predicate.

GENERALIZED EVENT KNOWLEDGE

Thematic fit derives from our linguistic and first-hand experience of events and their typical participants. McRae and Matsuki (2009) call this information **generalized event knowledge**. Typicality effects in argument selection pose an important challenge to symbolic models of selectional preferences and call for a different kind of representation to address their gradient nature.

Distributional models of selectional preferences share this assumption:

> The semantic constraints of predicates derive from the **co-occurrences with their arguments**.

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

However, selectional constraints do not simply consist in the set of observed arguments in a corpus, because they are by definition general knowledge about the semantic classes of the *possible* arguments of a predicate. For instance, if we know that *ponche* is a kind of liquor, we can understand the sentence *I drank a ponche*, even if we have never heard it before. Therefore, the process of inducing selectional preferences from corpus data must include a **generalization step** to infer the degree of preference for new, unseen lexemes as fillers of a given predicate slot. This is consistent with a **usage-based** view of selectional preferences, according to which they are abstractions from the lexemes that are observed as arguments of a particular predicate.

GENERALIZATION STEP

USAGE-BASED VIEW

Though the notion of selectional constraint applies to every category of predicate, research has mostly focused on verbs. Early computational models of selectional preferences, such as Resnik (1996), rely on existing **lexical resources** like WordNet to identify the semantic types selected by predicates. In Resnik's model, the generalization from observed predicate–argument pairs is performed with an information-theoretic similarity measure defined over the class hierarchy of WordNet. This has been an influential model, which has spawned a number of similar approaches, including Clark and Weir (2002), Brockmann and Lapata (2003) for German, Schulte im Walde et al. (2008), Ó Séaghdha and Korhonen (2012), which combine WordNet with Topic Models, and Judea et al. (2012), who use Wikipedia as lexical resource. DSMs are attractive alternatives to lexical resources for generalization in selectional preference models, since by design they are not limited to a particular data set, domain, or language. Moreover, they do not depend on the a priori stipulated structure of semantic classes that we find in manually constructed resources.

LEXICAL RESOURCES

CLUSTERING

A first group of DSMs generalize observed co-occurrences by **clustering** the distributional vectors of arguments extracted from the training corpus. The resulting clusters represent the semantic classes selected by the predicates and are used to estimate the plausibility of unseen arguments. Pereira et al. (1993) model the probability of a noun n as the argument of a predicate v in the following way, with C a set of distributionally induced clusters:

$$p(n, v) = \sum_{c \in C} p(c)p(c|n)p(c|v). \tag{9.36}$$

Pereira et al. (1993) group nouns that occur as heads of direct objects of verbs with hierarchical clustering. Rooth et al. (1999) and Prescher et al. (2000) formulate the distributional clustering method as an Expectation-Maximization algorithm. Other approaches include Pantel et al. (2007), Basili et al. (2007), who employ a variant of k-means clustering of LSA vectors, and the discriminative model of Bergsma et al. (2008) with CBC clustering (cf. Section 8.2.2).

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

The most influential family of distributional approaches to selectional preferences dispenses with clustering and is based on the following assumption:

> The thematic fit of a new lexeme as argument of a predicate depends on its distributional similarity with previously observed arguments.

Erk (2007), Padó et al. (2007), and Erk et al. (2010) propose **EPP**, an **exemplar model** of selectional preferences in which the thematic fit of a noun n as the argument of a verb v in a syntactic role r is measured with the similarity between the distributional vector of n and the vectors of a set of noun exemplars occurring in the same argument role of the predicate. Erk et al. (2010) compute the following selectional preference score:

EPP

$$\mathrm{SelPref_{EPP}}(n, r, v) = \sum_{n_i \in \mathrm{SeenArg(r,v)}} \frac{w_{r,v}(n_i)}{Z_{r,v}} \mathrm{sim}(\mathbf{n}, \mathbf{n_i}), \qquad (9.37)$$

where sim is a vector similarity measure, $\mathrm{SeenArg}(r, v)$ is a set of nouns occurring as arguments of v in the role r in the corpus, $w_{r,v}(n)$ is a weight (e.g., the co-occurrence frequency) estimating the salience of the observed argument n for v, and $Z_{r,v}$ is a normalization factor to make the score independent of the number of selected exemplars ($Z_{r,v} = \sum_{n_i \in \mathrm{SeenArg(r,v)}} w_{r,v}(n_i)$). $\mathrm{Selpref_{EPP}}$ is essentially the weighted average over similarities between the noun n and the previously observed arguments of v. Erk et al. (2010) experiment with different similarity measures and weights, showing that EPP is generally able to outperform WordNet-based and EM-based models.

Selectional preferences are typically defined as semantic constraints that predicates place on their arguments. Erk et al. (2010) also apply the EPP model to **inverse preferences**, namely the preferences that arguments have for their predicates. For instance, the noun *book* prefers to appear as direct object of verbs like *read* and *write*, while *violinist* prefers to occur as subjects of verbs like *play* and *perform*. In fact, psycholinguistic studies have shown with different experimental paradigms (e.g., semantic priming, self-paced reading, etc.) that nouns trigger expectations about the most typical verbs they co-occur with (McRae et al., 2005a). Inverse preferences are related to the Generative Lexicon model of lexical entries (Pustejovsky, 1995), in which a noun like *book* is represented as containing event information that refers to its typical function (e.g., *read*) and its typical mode of creation (e.g., *write*) (cf. Section 8.1).

INVERSE PREFERENCES

Baroni and Lenci (2010) propose a variation of EPP inspired by prototype representations of concepts (Murphy, 2002). The thematic fit of a noun n as

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

PROTOTYPE MODEL

an argument of a verb v in a syntactic role r is measured with the similarity between the distributional vector of n and a **prototype vector $\mathbf{r_v}$**:

$$\mathrm{SelPref}_{\mathrm{Prot}}(n, r, v) = \mathrm{sim}(\mathbf{n}, \mathbf{r_v}). \qquad (9.38)$$

The prototype vector is the centroid vector (cf. Section 8.2.1) of the most typical arguments of r observed in the training corpus:

$$\mathbf{r_v} = \frac{1}{|\mathrm{SeenArg(r, v)}|} \sum_{n_i \in \mathrm{SeenArg(r,v)}} \mathbf{n_i}. \qquad (9.39)$$

Baroni and Lenci (2010) define the set $\mathrm{SeenArg}(r, v)$ as the k arguments in the training corpus ($k = 20$ in their original setting) with the highest association score (e.g., PLMI; cf. Section 2.3.1) with v in the role r. Other variations of the exemplar and prototype models of selectional preferences and thematic fit have been explored by Schulte im Walde (2010), Greenberg et al. (2015), Sayeed et al. (2015), and Santus et al. (2017). Peirsman and Padó (2010) introduce a cross-lingual version of EPP that employs a bilingual DSM (cf. Section 8.4).

Selectional preferences are usually modeled as two-way relations between a predicate and an argument. However, this representation does not take into account the fact that the semantic constraints of a particular syntactic role depend on how other roles in the same sentence are filled (Elman, 2009, 2011, 2014). For instance, the expected patient of the verb *cut* is likely to be *meat* if the agent is *butcher*, and to be *hair* if the agent is *coiffeur*. Lenci (2011) extends the prototype model to account for **multi-way selectional preferences** with a dynamic process of compositional update of semantic constraints. The distributional information coming from the agent and the predicate are composed with either an additive or a multiplicative model (cf. Section 9.2), to generate a prototype vector that represents the expectations on the likely fillers of the patient role, given the agent filler. This makes it possible to model the dynamic aspect of thematic fit, since the expectations on an argument are progressively updated as the other roles in the sentence are filled (cf. Chersoni et al., 2017 for a variant of the same approach). Van de Cruys (2010a) uses Non-negative Tensor Factorization (NTF), which is a generalization of Non-negative Matrix Factorization (cf. Section 2.5.3), to capture the multi-way selectional preferences of transitive predicates with a third-order tensor representing subject-verb-object co-occurrence data.

MULTI-WAY SELECTIONAL PREFERENCES

TOPIC MODELS

Other distributional approaches to selectional preferences use **Topic Models** (cf. Section 4.4). In particular, the EM-based approach of Rooth et al. (1999) has inspired more recent works that use LDA. The idea is to model selectional preferences with a probability distribution over topics representing latent semantic classes selected by predicates. Ó Séaghdha (2010) proposes a probabilistic model of thematic fit, by computing the probability of a noun n as the argument of v in the role r as follows:

$$p(n|v,r) = \sum_{z \in Z} p(n|z)p(z|v,r), \qquad (9.40)$$

where Z is a set of latent topics induced with LDA. Ritter et al. (2010) use a very similar solution to model multi-way selectional preferences of binary predicates (e.g., companies and organizations are likely to fill the first argument of *x is headquartered in y*, and locations the second argument). LDA approaches to selectional preferences are competitive with the EM-based model of Rooth et al. (1999), the similarity-based approach of Erk (2007), and the clustering-based approach of Pantel et al. (2007).

NEURAL NETWORKS

NEURAL LANGUAGE MODELS

Selectional preferences are also modeled with **neural networks**. Van de Cruys (2014) use a feed-forward network that takes in input the concatenation of the embeddings of the verb and one or two arguments (the latter case for multi-way preferences) and computes their plausibility score. Similarly to Collobert and Weston (2008), the network is trained to discriminate attested verb-argument tuples from noisy ones obtained by replacing the verb with a random one. Tilk et al. (2016) model two-way and multi-way selectional preferences with a recurrent neural network trained to predict the filler of the semantic role of a given verb. However, their network needs to be trained on a corpus previously annotated with semantic roles. Other approaches model the preferred arguments of a predicate directly with the predictions of **neural language models**. Elman and McRae (2019) use an SRN (cf. Section 6.2.1) to represent generalized event knowledge, while Metheniti et al. (2020) study to what extent selectional preferences are reflected in the word predictions by BERT. Given an example sentence containing a target predicate with its argument masked using a `[MASK]` token (e.g., *The journalist writes the* `[MASK]`.), the probability that BERT assigns to the masked position is retrieved and correlated with the predicate-argument plausibility. Generally, the correlations are not very strong, suggesting a limited mastery of selectional preferences. Similarly, the experiments by Pedinotti et al. (2021a) reveal that the thematic fit predictions of contextual DSMs often depend on surface linguistic features, such as frequent words, collocations, and syntactic patterns, thereby showing sub-optimal generalization abilities.

Distributional models of selectional preferences are typically evaluated in two tasks: pseudo-disambiguation and the prediction of human thematic fit

Table 9.6 An example of thematic fit ratings from the McRae dataset

verb	role	noun	thematic fit
arrest	agent	cop	6.7
arrest	patient	cop	1.6
fire	agent	employer	6.1
fire	patient	employer	2.4

PSEUDO-DISAMBIGUATION

ACCURACY

ratings. In the **pseudo-disambiguation** task (Dagan et al., 1999; Rooth et al., 1999), the model decides which of two words is a better argument for a syntactic role r in a predicate p. Given an observed set of tuples $\langle p, r, l\rangle$ (e.g., $\langle$*drink*, dobj, *beer*$\rangle$), each lexeme l is paired with a randomly selected word l' (e.g., *computer*). The tuples $\langle p, r, l\rangle$ and $\langle p, r, l'\rangle$ are removed from the training corpus to make sure that we measure a model's ability to generalize from observed items and predict the plausibility of unseen arguments of a predicate. Models are then evaluated for their **accuracy** in choosing for each pair $\langle l, l'\rangle$ (e.g., $\langle$*beer*, *computer*$\rangle$) the most plausible filler for p in the role r. Pseudo-disambiguation can be viewed as a kind of WSD task in which the two lexemes l and l' together form a "pseudo-word" that models disambiguate by choosing the lexical item that fits better in the given predicate.

THEMATIC FIT

SPEARMAN CORRELATION

MCRAE – PADÓ

FERRETTI

SP-10K

Thematic fit evaluation (Padó et al., 2007; Baroni and Lenci, 2010; Sayeed et al., 2016) is instead based on datasets of human ratings about the plausibility of nouns as arguments of verbs (cf. Table 9.6). The evaluation measure is the **Spearman rank correlation** (ρ) between distributional similarity scores and thematic fit ratings. The **McRae** (McRae et al., 1998) and **Padó** datasets (Padó, 2007) respectively consist of $1,444$ and 414 typicality scores for verb–agent (e.g., *doctor-advise*) and verb–patient pairs (e.g., *hit-ball*), whereas the **Ferretti** dataset (Ferretti et al., 2001) includes ratings for 374 verb–instrument (e.g., *cut-mower*) and 248 verb–location pairs (e.g., *teach-classroom*). The scores range from 1 (atypical) to 7 (very typical). The largest dataset available to date is **SP-10K** (Zhang et al., 2019), which contains over ten thousand predicate argument pairs rated for their plausibility.

DTFIT

BICKNELL

Differently from the previous datasets, **DTFit** (Vassallo et al., 2018) contains tuples of different lengths, so that crowdsourced typicality ratings of an argument depend on its interaction with the other arguments in the tuple. The dataset consists of triples and quadruples that differ for typical (e.g., *sergeant-assign-mission*) vs. atypical (e.g., *sergeant-assign-homework*) patients, locations, or instruments. The much smaller **Bicknell** dataset (Bicknell et al., 2010) includes 100 typical and atypical agent–verb–patient triples. Rather than focusing on typicality, the **Wang2018** dataset (Wang et al., 2018) contains 3, 080 agent–verb–patient triples distinguishing an atypical but physically plausible event (e.g., *The student climbed the ship*) from an atypical and physically implausible one (e.g., *The student climbed the water*).

WANG2018

9.7.1 Modeling Coercion: The Case of Logical Metonymy

The term **(complement) coercion** refers to a wide range of phenomena in which an argument is reinterpreted to overcome the violation of the selectional preferences of its predicate (Lauwers and Willems, 2011; Pustejovsky and Batiukova, 2019). One well-known case of coercion is **logical metonymy**:

COERCION

LOGICAL METONYMY

(13) a. The author began the book.
b. The dog finished the bone.

The verb *begin* in (13a) normally selects for an event and the noun *book* denotes an object, but it is reinterpreted as the participant of an implicit event that is recovered, like *write* or *read* (cf. Section 9.1.1). This reinterpretation also produces extra processing costs during online sentence comprehension (McElree et al., 2001; Traxler et al., 2002; Zarcone et al., 2014).

Formal semantics models logical metonymy with type-shifting mechanisms (Asher, 2011, 2015) or with complex lexical entries, like in the Generative Lexicon. Pustejovsky (1995) argues that the implicit event is retrieved from the information encoded in the Qualia Structure (cf. Section 8.1). According to this hypothesis, the TELIC role of *book* in (13a) includes the event of reading as its typical purpose, and the AGENTIVE role specifies the event of writing as its mode of creation. These events are retrieved to solve the predicate-argument type mismatch, thereby producing a semantic representation equivalent to *begin to write (read) the book*. However, several cases cannot be accounted by Qualia roles equally well. The most straightforward interpretation of (13b) is that the dog finished eating the bone, but being eaten can hardly be regarded as the typical purpose of bones (cf. also Lascarides and Copestake, 1998).

An alternative explanation of logical metonymy is that it relies on our generalized event knowledge (Zarcone et al., 2014). Reading and writing are part of the interpretation of (13a), because they are likely events associated with *book* and *author*. Similarly, eating is recovered from (13b), because it is the most typical action performed by a dog on a bone. Distributional semantics has pursued this argument by representing knowledge about events and their participants with co-occurrence data extracted from corpora. Lapata and Lascarides (2003) propose a **probabilistic model** of logical metonymy in which the recovered event $\hat{e}$ is the one that maximizes the following joint probability:

PROBABILISTIC MODEL

$$\hat{e} = \underset{e}{\operatorname{argmax}}\, p(e, n_1, v, n_2), \tag{9.41}$$

where v is the metonymic verb, n_1 its subject, and n_2 its direct object. Assuming that n_2 is conditionally independent of v and n_1 and that n_1 is conditionally independent of v, they obtain:

$$p(e, n_1, v, n_2) \approx p(e)p(n_2|e)p(v|e)p(n_1|e). \tag{9.42}$$

The probabilities in Equation 9.42 are estimated with simple co-occurrence frequencies. Therefore, (13b) is interpreted as *The dog finished eating the bone* because *eat* is the most probable event given *finished*, *dog*, and *bone*.

THEMATIC FIT MODEL

Other distributional analyses of logical metonymy are based on the notion of **thematic fit** (Zarcone et al., 2012, 2013; Chersoni et al., 2017, 2021b). Differently from Lapata and Lascarides (2003), these models predict both the implicit event and the higher cognitive cost of logical metonymy. Chersoni et al. (2017, 2021b) analyze logical metonymy within a general model of processing complexity, which is based on the hypothesis that the cognitive load of a sentence is inversely proportional to the mutual typicality of its components, estimated with a distributional measure of thematic fit. Given a transitive sentence $s = \langle n_1, v, n_2 \rangle$, its **typicality score** θ_s is defined as follows:

TYPICALITY SCORE

$$\theta_s = \theta(n_1|v)\theta(n_2|v)\theta(n_2|n_1), \tag{9.43}$$

where $\theta(n_1|v)$ is the thematic fit of the subject noun with the verb, $\theta(n_2|v)$ the thematic fit of the object noun with the verb, and $\theta(n_2|n_1)$ the thematic fit of the object with the subject. For instance, the typicality score of (13a) is the product of the typicality of *author* as the subject of *began*, the typicality of *book* as the object of *began*, and the typicality of *book* as an object of events involving *author* as subject. The θ scores are computed with a prototype model of thematic fit, adapting Equation 9.38. Chersoni et al. (2017) use the typicality scores to account for the experimental results in McElree et al. (2001) and Traxler et al. (2002), which show that metonymic sentences produce higher processing costs than control sentences with no coercion (e.g., *The author wrote the book*). In fact, metonymic sentences have significantly lower values of θ_s than control sentences, supporting the hypothesis that the low thematic fit between the metonymic verb and the object-denoting noun triggers complement coercion and, at the same time, causes the extra processing load.

IMPLICIT EVENT

Chersoni et al. (2017, 2021b) adopt the same model to predict the **implicit event** $\hat{e}$ recovered in interpreting logical metonymy. Extending Equation 9.43, this is the event that maximizes the following product of thematic fit scores:

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

$$\hat{e} = \operatorname*{argmax}_{e} \theta(n_1|v)\theta(e|v)\theta(n_1|e)\theta(n_2|e)\theta(n_2|n_1). \tag{9.44}$$

Assuming that the choice of the implicit event does not depend on $\theta(n_1|v)$ and on $\theta(n_2|n_1)$, this equation can be further simplified as:

$$\hat{e} = \operatorname*{argmax}_{e} \theta(e|v)\theta(n_1|e)\theta(n_2|e). \tag{9.45}$$

For instance, *eat* is the preferred implicit event in (13b), because it is the event with the highest thematic fit with the various sentence components: the subject, the metonymic verb, and the direct object. Like in Lapata and Lascarides (2003), this model is able to account for the fact that the event recovered when interpreting logical metonymy also depends on the choice of verb subject (e.g., the preferred implicit event in *The student began the book* is *read*, while in *The author began the book* is *write*). Rambelli et al. (2020) also test the ability of Transformer language models to predict the implicit event, comparing them with probabilistic and thematic fit approaches.

9.8 Compositional Distributional Semantics: Limits and Prospects

In order to explain the **productivity** of natural language, a semantic theory must contain some general process to assign an interpretation to a potentially infinite number of complex expressions. The classical approach to address productivity is assuming that the construction of semantic representations is governed by the **Principle of Compositionality**: The meaning of a whole expression is determined by the meaning of its syntactic parts and their combination (cf. Section 9.1). This entails that there is a computational procedure to combine the interpretation of lexical items to obtain the interpretation of the expressions they are part of, according to syntactic structure. In symbolic models, this procedure is function application, which in turn relies on a basic distinction between lexemes acting as functions with variables, and lexemes acting as arguments that saturate them. In this chapter, we have reviewed several ways to tackle productivity with compositional distributional semantics. Now, it is time to make a general appraisal of the state of the art in this area, to highlight its current limits and possible future research lines.

PRODUCTIVITY

PRINCIPLE OF COMPOSITIONALITY

A first group of methods to construct the distributional representation of complex linguistic expressions substantially adheres to the Principle of Compositionality (Liang and Potts, 2015; Potts, 2019).

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

> Given the syntactic node $[_C AB]$ and the distributional representations of A and B, $\mathbf{DR}(A)$ and $\mathbf{DR}(B)$, there is a composition function f such that $\mathbf{DR}(C) = f(\mathbf{DR}(A), \mathbf{DR}(B))$.

Proposals vary for the type of distributional representation and for the choice of f. For instance, $\mathbf{DR}(A)$ and $\mathbf{DR}(B)$ can be vectors and f an operation like sum or tensor product (cf. Section 9.2). DFM assumes that $\mathbf{DR}(A)$ and $\mathbf{DR}(B)$ are tensors of different order (e.g., a matrix and a vector), and f is tensor-vector multiplication (cf. Section 9.3). The composition function can also be a feed-forward neural network, like in MV-RNN (cf. Section 9.3.1). Despite its being theoretically unsatisfactory, vector sum remains extremely competitive with approaches directly inspired by formal models of compositionality, such as DFM, which also suffer from great problems of scalability.

SENTENCE EMBEDDING

The limits of the methods above have prompted the popularity of a different paradigm based on the idea that the distributional representation of an input sentence (**sentence embedding**) is the hidden state of a neural network trained on some linguistic task, such as prediction or inference (cf. Section 9.4). The attractiveness of this approach lies in its simplicity and scalability. The networks are able to generalize beyond their training data and to assign a vector to any new input sentence, thereby showing some sort of productivity. Moreover, their sensitivity to word order allows sentence embeddings to avoid the problems of commutative vector operations, like addition or multiplication.

MEANINGFUL PART

Neural sentence embeddings depart from the main assumptions grounding the Principle of Compositionality. This is based on a neat separation between syntax and semantics: The former identifies the structure of linguistic expressions, following which the semantic operations combine the meaning of their parts. As Nefdt (2020) argues, while the Principle of Compositionality presupposes the notion of "**meaningful part**," that is some correspondence between the parts of a sentence and the parts of its meaning, no such notion is clearly identifiable in neural networks, whose embeddings are not constructed by combining the meaning of independently identified syntactic components. In fact, the sentence embeddings are just internal network states that come to encode a mixture of linguistic information, including several syntactic dimensions alongside semantic ones (Adi et al., 2017; Conneau et al., 2018a), as a by-product of learning a certain sentence-related task.

The lack of compositionality in neural networks has been a topic of intense discussion at least since when Fodor and Pylyshyn (1988) proposed it as a key argument against connectionist models of cognition. This debate has been

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

raging throughout these last decades and is still alive today, when new generations of neural models are available (Elman et al., 1996; Marcus, 2001; Calvo and Symons, 2014; Baroni, 2019; Berent and Marcus, 2019; Rabovsky and McClelland, 2019). Fodor and Pylyshyn (1988) argue that human cognition and linguistic competence in particular are characterized by the properties of **systematicity** and **compositionality**, which they define in the following way:

SYSTEMATICITY
COMPOSITIONALITY

Systematicity: The ability to produce/understand some sentences is intrinsically connected to the ability to produce/understand certain others.
Compositionality: A lexical item must make approximately the same contribution to each expression in which it occurs.

For instance, (14a) and (14b) are systematically related, because whoever understands the former must also understand the latter:

(14) a. The black cat chases the brown dog.
b. The black dog chases the brown cat.

This systematic relation depends on the fact that the sentences are instances of the same general structures licensed by English syntax. According to Fodor and Pylyshyn (1988), systematicity presupposes compositionality: The relation between (14a) and (14b) is possible only provided that the words have the same meaning in the two sentences. Therefore, while systematicity is a property about the nature of syntactic structures, compositionality is a property about the contribution of the lexicon to the interpretation of complex expressions. Taken together the two principles essentially correspond to Fregean Compositionality: The meaning of a complex expression is a structured entity, and the syntactic parts of the expression correspond to the parts of its meaning.

Fodor and Pylyshyn (1988) use systematicity and compositionality as arguments against neural models and distributed representations (cf. Section 8.1). According to them, these properties can be explained only by systems, like symbolic ones, that combine internally structured representations. Since neural networks generate unstructured representations, they cannot tackle systematicity and compositionality, and therefore they would lack explanatory value. For instance, a neural network that produces two distinct sentence embeddings for (14a) and (14b), might capture some aspects of their content, but nevertheless would not account for their systematic relations and the fact that the two sentences are instances of the same general structure that could generate other – potentially unlimited – systematically related sentences.

The same argument is adopted by Marcus (2001) and Berent and Marcus (2019) to defend their **algebraic hypothesis** of human mind.

ALGEBRAIC
HYPOTHESIS

Learning mechanisms operate **algebraically** over abstract rules and include the capacity to form abstract categories that treat all of their members alike and to operate over such classes using variables.

Berent and Marcus (2019) argue that algebraic rules with open variables explain a further property of human cognition, consisting in the ability of carrying out **across-the-board generalizations**:

ACROSS-THE-BOARD GENERALIZATIONS

> The hallmark of algebraic rules is not simply the capacity to generalize. Rules generalize across the board. They can extend generalizations to any member of a category, irrespective of its similarity to training items, and they obey systematicity and compositionality. (Berent and Marcus, 2019, p. e80)

This property corresponds to the way symbolic models construct the interpretation of complex expressions by combining functional symbols with open variables and arguments saturating them (cf. Section 9.1). For instance, representing *run* with the function $\lambda x[\text{run}(x)]$ explains the productivity of this predicate that can be applied to any argument satisfying the type constraints of the variable. According to Berent and Marcus (2019), since neural networks do not operate on symbolic rules with variables, they are not able to generalize across the board. They *do* generalize to new items, but only as long as they are **similar** to training data. Therefore, their productivity would be limited to **analogical generalizations**, while across-the-board ones are not similarity-driven, since algebraic rules can be applied to any new item that can fill their variables. This argument is supported by experiments that show that seq2seq models – similar to those used to generate sentence embeddings (cf. Section 9.4.3) and trained on simplified or artificial language data – show generalization abilities, but fail to generalize in a systematic and compositional way (Lake and Baroni, 2018; Loula et al., 2018; Goodwin et al., 2020; Hupkes et al., 2020).

SIMILARITY

ANALOGICAL GENERALIZATIONS

The debate sparked by Fodor and Pylyshyn (1988) and Marcus (2001) concerns distributional semantics too, since its representations are distributed and neural networks are the mainstream approach to learn them. Their critical arguments rest on the assumption that compositionality, systematicity, and across-the-board generalizations are the hallmarks of natural language. However, these assumptions are quite disputable. As discussed in Section 9.1.1, several linguistic phenomena are not compatible with Fregean Compositionality and suggest that natural language is actually **quasi-compositional** (Rabovsky and McClelland, 2019). The hypothesis about the contextually invariant nature of lexical meaning contrasts with the sense modulation

QUASI-COMPOSITIONALITY

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

that lexical items constantly undergo when they are composed. In fact, the productivity that a semantic theory is called to explain concerns not only the ability of generating and understanding a potentially unlimited number of complex expressions, but also the ability of generating and understanding a potentially unlimited number of new word senses in context (Pustejovsky, 1995). The systematicity assumption has also been questioned in light of the pervasiveness of nonsystematic and semiregular processes in language, which apply to categories of entities governed by overlapping and complex constraints (Johnson, 2004). Rather than being the realm of across-the-board generalizations, natural language is characterized by the **partial productivity** (Goldberg, 2019) of analogical generalizations based on the similarity to previously witnessed exemplars. Therefore, the fact that sentence embeddings learnt by neural models do not comply with the Principle of Compositionality and strive to generalize systematically does not entail that they could not have explanatory value for natural language semantics, since this also departs from those same properties (Baroni, 2019).

PARTIAL PRODUCTIVITY

However, sentence embeddings suffer from several empirical limitations concerning their ability to encode significant aspects of meaning. For instance, Zhu et al. (2018) show that current models are not able to discriminate between different syntactic realizations of semantic roles and fail to recognize that *Lilly loves Imogen* is more similar to its passive counterpart than to *Imogen loves Lilly*. Sentence embeddings apparently perform very well in inference tasks, but this is mostly due to statistical artifacts in the training and test datasets (cf. Section 9.5). When the biases are removed, the performance of the models drops significantly (McCoy et al., 2019a). Glockner et al. (2018) test various sentence embeddings on a dataset of inferences carefully designed to avoid annotation artifacts and confirm their scarce generalization abilities. Simple additive models remain a very strong baseline (cf. Section 9.4), revealing that neural networks are really not successful in constructing sentence meaning, independently of their lack of systematicity or compositionality. These problems have surely contributed to shift the focus from methods to learn sentence embeddings to **contextual DSMs**. The expectation is that the deeply contextualized embeddings of word tokens might encode several aspects of sentence meaning. Although this happens to a certain extent (Klafka and Ettinger, 2020), several analyses show that, their increasing complexity notwithstanding, contextual DSMs too have strong limits in capturing key aspects of compositional meaning such as semantic roles and negation (Ettinger, 2020; Staliūnaité and Iacobacci, 2020), as well as partially productive structures like metaphorical and multiword expressions (Shwartz and Dagan, 2019; Pedinotti et al., 2021b).

CONTEXTUAL DSMs

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

Despite the improvements of the last generation of models, we cannot but conclude that distributional semantics still lacks a theoretically and empirically satisfactory account of the processes to construct the meaning of complex expressions. Berent and Marcus (2019) depict a strong contrast between symbolic models that explain natural language productivity in terms of rules with open variables operating on structured representations, and models – like neural ones and DSMs – that use distributed, unstructured representations. The problem is that neither paradigm alone seems to be able to provide an adequate explanation of meaning construction. Formal models represent sentences with logical forms that are suitable to capture their inferential properties, but do not capture the graded aspects of meaning or its context-sensitiveness. Continuous vectors are instead particularly apt to deal with similarity-driven generalizations, meaning shifts in contexts, and other phenomena that are challenging for classical compositionality, but they strive to capture several core aspects of sentence meaning and lack sufficient generalization strength. These considerations have prompted the development of **hybrid DSMs** that integrate the complementary features of symbolic and distributional representations (Boleda and Herbelot, 2016). Thus, hybrid models can exploit their mutual elements of strength, in particular the flexibility of the latter and the structured nature of the former. This line of research is pursued by Beltagy et al. (2016), McNally and Boleda (2016), McNally (2017), and Chersoni et al. (2019). They hypothesize a "division of labor" between formal and distributional semantics and depart from the assumption shared by all the models reviewed in this chapter that the distributional representation of a sentence is a vector, exactly like lexical items. They instead represent sentences with logical forms enriched with distributional vectors of their lexemes. For instance, in the **Structured Distributional Model** (SDM) by Chersoni et al. (2019), the semantic representation of a sentence is a formal structure inspired by Discourse Representation Theory (Kamp and Reyle, 1993) and containing vectors that replace the logical constants associated with lexical items. This structure is dynamically and incrementally built by integrating knowledge about events and their typical participants, as they are activated by lexical items. Other models that combine formal and distributional semantics are proposed by Emerson and Copestake (2017) and Herbelot and Copestake (2021). The results of this line of research are promising, and Chersoni et al. (2019) and Pedinotti et al. (2021b) obtain performances that are competitive with neural language models. However, they are still applied to a limited number of phenomena, and much research is needed to improve their scalability and performance in downstream tasks.

HYBRID DSMS

SDM

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press

Berent and Marcus (2019) argue that the ability to operate on structured representations with open variables is an essential precondition to explain natural language productivity. It is indeed possible that further improvements might come from developing more explicit mechanisms in distributional semantics to represent abstract schemas with open slots that are filled to produce new instances of linguistic expressions. On the other hand, the continuous nature of distributional representations is an equally crucial aspect to capture the flexibility and quasi-regularity of natural language, whose constructions have slots with a prototype rather than categorical structure, and similarity and analogy are key mechanisms driving their composition with new arguments.

9.9 Summary

The representation of phrases and sentences has become an intense research area that has definitely pushed the boundary of distributional semantics beyond its original territory, the lexicon. **Compositional distributional semantics** typically represents complex linguistic expressions with vectors that are constructed with different types of methods. A first group of models combine the vectors or higher-order linear-algebraic objects associated with the component lexemes and are consistent with the **Fregean Principle of Compositionality**, traditionally adopted by formal semantics to explain natural language productivity. A second approach consists in learning **sentence embeddings** with deep neural networks that instead radically depart from such a principle. Distributional semantics is also actively engaged in modeling **selectional preferences** and coercion phenomena in predicate-argument composition.

An important outcome of this research has been the development of context-sensitive representations that can account for the meaning changes that words undergo when they are combined with other expressions. This has sparked a new generation of **contextual DSMs**, which are a significant novelty in the distributional semantic landscape, especially for their increased performances in AI and NLP downstream applications.

Still, how distributional representations can be effectively projected from the lexical to the sentence or even discourse level remains largely an open issue, because even the most advanced models fall short of capturing important meaning dimensions. The future challenges for compositional distributional semantics concern both the methods to construct the representation of complex expressions, and their evaluation. In fact, better methodologies and more reliable benchmarks are needed to understand the aspects of compositional meaning encoded by distributional representations.

9.10 Further Reading

- The Principle of Compositionality: Heim and Kratzer (1998); Marcus (2001); Werning et al. (2012); Baggio (2018); Martin and Baggio (2019)
- Compositional DSMs: Mitchell and Lapata (2010); Erk (2012); Baroni (2013); Baroni et al. (2014b); Liang and Potts (2015); Potts (2019)
- Neural networks and sentence embeddings: Goldberg (2017); Pavlick (2022)
- Neural language models: Han et al. (2021)
- Contextual DSMs: Liu et al. (2020); Erk and Chronis (2022); Lenci et al. (2022)
- Analysis of contextual DSMs and neural language models: Belinkov and Glass (2019); Rogers et al. (2020); Belinkov (2022)
- Selectional preferences and coercion: Light and Greiff (2002); Erk et al. (2010); Lauwers and Willems (2011); Pustejovsky and Batiukova (2019)

https://doi.org/10.1017/9780511783692.010 Published online by Cambridge University Press