Chapter 3
Compositional Semantics

Abstract  Many important applications in NLP fields rely on understanding more complex language units such as phrases, sentences, and documents beyond words. Therefore, compositional semantics has remained a core task in NLP. In this chapter, we first introduce various models for binary semantic composition, including additive models and multiplicative models. After that, we present various typical models for N-ary semantic composition including recurrent neural network, recursive neural network, and convolutional neural network.

3.1 Introduction

From the previous chapter, following the distributed hypothesis, one could project the semantic meaning of a word into a low-dimensional real-valued vector according to its context information, which is named as word vectors. Here comes a further problem: how to compress a higher semantic unit into a vector or other kinds of mathematical representations like a matrix or a tensor. In other words, using representation learning to model a semantic composition function remains an unsolved but surging research topic recently.

Compositionality enables natural languages to construct complex semantic meanings from the combinations of simpler semantic elements. This property is often captured with the following principle: the semantic meaning of a whole is a function of the semantic meanings of its several parts. Therefore, the semantic meanings of complex structures will depend on how their semantic elements combine.

Here we express the composition of two semantic units, which are denoted as \( u \) and \( v \), respectively, and the most intuitive way to define the joint representation could be formulated as follows:

\[
\mathbf{p} = f(u, v),
\]

(3.1)

where \( \mathbf{p} \) corresponds to the representation of the joint semantic unit \((u, v)\). It should be noted that here \( u \) and \( v \) could denote words, phrases, sentences, paragraphs, or even higher level semantic units.
However, given the representations of two semantic constituents, it is not enough to derive their joint embeddings with the lack of syntactic information. For instance, although the phrase machine learning and learning machine have the same vocabulary, they contain different meanings: machine learning refers to a research field in artificial intelligence while learning machine means some specific learning algorithms. This phenomenon stresses the importance of syntactic and order information in a compositional sentence. Reference [12] takes the role of syntactic and order information into consideration and suggests a further refinement of the above principle: the meaning of a whole is a function of the meaning of its several parts and the way they are syntactically combined. Therefore, the composition function in Eq. (3.1) is redefined to combine the syntactic relationship rule $R$ between the semantic units $u$ and $v$:

$$p = f(u, v, R),$$  \hspace{1cm} (3.2)  

where $R$ denotes the syntactic relationship rule between two constituent semantic units.

Unfortunately, even this formulation may not be fully adequate. Therefore, [7] claims that the meaning of a whole is greater than the meanings of its several parts. It implies that people may suffer from the problem of constructing complex meanings rather than simply understanding the meanings of several parts and their syntactic relations. In real language composition, in different contexts, the same sentence could have different meanings, which means that some sentences are hard to understand without any background information. For example, the sentence Tom and Jerry is one of the most popular comedies in that style. needs two main backgrounds: Firstly, Tom and Jerry is a special noun phrase or knowledge entity which indicates a cartoon comedy, rather than two ordinary people. The other prior knowledge should be that style, which needs further explanation in the previous sentences. Hence, a full understanding of the compositional semantics needs to take existing knowledge into account. Here, the argument $K$ is added into the composition function, incorporating knowledge information as a prior in the compositional process:

$$p = f(u, v, R, K),$$  \hspace{1cm} (3.3)  

where $K$ represents the background knowledge.

Reference [4] claims that we should ask for the meaning of a word in isolation but only in the context of a statement. That is, the meaning of a whole is constructed from its parts, and the meanings of the parts are meanwhile derived from the whole. Moreover, compositionality is a matter of degree rather than a binary notion. Linguistic structures range from fully compositional (e.g., black hair), to partly compositional syntactically fixed expressions, (e.g., take advantage), in which the constituents can still be assigned separate meanings, and non-compositional idioms (e.g., kick the bucket) or multi-word expressions (e.g., by and large), whose meaning cannot be distributed across their constituents [11].
From the above three equations formulating composition function, it could be concluded that composition could be viewed as a specific binary operation but beyond this. The syntactic message could help to indicate a particular approach while background knowledge helps to explain some obscure words or specific context-dependent entities such as pronouns. Beyond binary compositional operations, one could build the sentence-level composition by applying binary composition operations recursively. In this chapter, we will first explain some sorts of basic binary composition functions in both the semantic vector space and matrix-vector space. After, we will climb up to more complex composition scenarios and introduce several approaches to model sentence-level composition.

3.2 Semantic Space

3.2.1 Vector Space

In general, the central task in semantic representation is projecting words from an abstract semantic space to a mathematical low-dimensional space. As introduced in the previous chapters, to make the transformation reasonable, the purpose is to maintain the word similarity in this new projected space. In other words, the more similar the words are, the closer their vectors should be. For instance, we hope the word vectors \( \textbf{w(} \text{book} \text{)} \) and \( \textbf{w(} \text{magazine} \text{)} \) are close while the word vectors \( \textbf{w(} \text{apple} \text{)} \) and \( \textbf{w(} \text{computer} \text{)} \) are far away. In this chapter, we will introduce several widely used typical semantic vector space including one-hot representation, distributed representation, and distributional representation.

3.2.2 Matrix-Vector Space

Despite the wide use of semantic vector spaces, an alternative semantic space is proposed to be a more powerful and general compositional semantic framework. Different from conventional vector spaces, matrix-vector semantic space utilizes a matrix to represent the word meaning rather than a skinny vector. The motivation behind this is when modeling the semantic meaning under a specific context, one is wondering not only what is the meaning of each word, but also the holistic meaning of the whole sentence. Thus, we concern about the semantic transformation between adjacent words inside each sentence. However, the semantic vector space could not characterize the semantic transformation of one word on the others explicitly.

Driven by the idea of modeling semantic transformation, some researchers have proposed to use a matrix to represent the transformation operation of one word on the others. Different from those vector space models, it could incorporate some structural information like the word order and syntax composition.
3.3 Binary Composition

The goal is to construct vector representations for phrases, sentences, paragraphs, and documents. Without loss of generality, we assume that each constituent of a phrase (sentence, paragraph, or document) is embedded into a vector which will be subsequently combined in some way to generate a representation vector for the phrase (sentence, paragraph, or document).^1

In this section, we focus on binary composition. We will take phrases consisting of a head and a modifier or complement as an example. If we cannot model the binary composition (or phrase representation), there is little hope that we can construct more complex compositional representations for sentences or even documents. Therefore, given a phrase such as “machine learning” and the vectors \( u \) and \( v \) representing the constituents “machine” and “learning”, respectively, we aim to produce a representation vector \( p \) of the whole phrase. Let the hypothetical vectors for machine and learning be \([0, 3, 1, 5, 2]\) and \([1, 4, 2, 2, 0]\), respectively. This simplified semantic space will serve to illustrate examples of the composition functions which we consider in this section.

The fundamental problem of semantic composition modeling in representing a two-word phrase is designing a primitive composition function as a binary operator. Based on this function, one could apply it on a word sequence recursively and derive sentence-level composition. Here a word sequence could be any level of the semantic units, such as a phrase, a sentence, a paragraph, a knowledge entity, or even a document.

From the previous section, one of the basic formulae is to formulate semantic composition \( f \) in the following equation:

\[
p = f(u, v, R, K),
\]

where \( u, v \) denote the representations of the constituent parts in this semantic unit, \( p \) denotes the joint representation, \( R \) indicates the relationship while \( K \) indicates the necessary background knowledge. The expression defines a wide class of composition functions. For easier discussion, we give some appropriate constraints to narrow the space of our considering function. First, we will ignore the background knowledge \( K \) to explore what can be achieved without any utilization of background or world knowledge. Second, for the consideration of the syntactic relation \( R \), we can proceed by investigating only one relation at a time. And then we can remove any explicit dependence on \( R \) which allows us to explore any possible distinct composition function for various syntactic relations. That is, we simplify the formula \( p = f(u, v) \) by simply ignoring the background knowledge and relationship.

---

^1Note that, the problem of combining semantic vectors of small units to make a representation for a multi-word sequence is different from the problem of incorporating information about multi-word contexts into a distributional representation for a single target word.
3.3 Binary Composition

In recent years, modeling the binary composition function is a well-studied but still challenging problem. There are mainly two perspectives toward this question, including the additive model and the multiplicative model.

### 3.3.1 Additive Model

The additive model has a constraint in which it assumes that $p$, $u$, and $v$ lie in the same semantic space. This essentially means that all syntactic types have the same dimension. One of the simplest ways is to directly use the sum to represent the joint representation:

$$ p = u + v. $$ (3.5)

According to Eq. (3.5), the sum of the two vectors representing machine and learning would be $w(machine) + w(learning) = [1, 7, 3, 7, 2]$. It assumes that the composition of different constituents is a symmetric function of them; in other words, it does not consider the order of constituents. Although having lots of drawbacks such as lack of the ability to model word orders and absence from background syntactic or knowledge information, this approach still provides a relatively strong baseline [9].

To overcome the word order issue, one easy variant is applying a weighted sum instead of uniform weights. This is to say, the composition has the following form:

$$ p = \alpha u + \beta v, $$ (3.6)

where $\alpha$ and $\beta$ correspond to different weights for two vectors. Under this setting, two sequences $(u, v)$ and $(v, u)$ have different representations, which is consistent with real language phenomena. For example, “machine learning” and “learning machine” have different meanings which requires different representations. In this setting, we could give greater emphasis to heads than other constituents. As an example, if we set $\alpha$ to 0.3 and $\beta$ to 0.7, the $0.3 \times w(machine) = [0, 0.9, 0.3, 1.5, 0.6]$ and $0.7 \times w(learning) = [0.7, 2.8, 1.4, 1.4, 0]$, and “machine learning” is represented by their addition $0.3 \times w(machine) + 0.7 \times w(learning) = [0.7, 3.6, 1.7, 2.9, 0.6]$.

However, this model could not consider prior knowledge and syntax information. To incorporate prior information into the additive model, one method combines nearest neighborhood semantics into composition, deriving

$$ p = u + v + \sum_{i=1}^{K} n_i, $$ (3.7)

where $n_1, n_2, \ldots, n_K$ denote all semantic neighbors of $v$. Therefore, this method could ensemble all synonyms of the component as a smoothing factor into composition function, which reduces the variance of language. For example, if in
the composition of “machine” and “learning”, the chosen neighbor is “optimizing”, with \( w(\text{optimizing}) = [1, 5, 3, 2, 1] \), then this leads to the situation that the representation of “machine learning” becomes \( w(\text{machine}) + w(\text{learning}) + w(\text{optimizing}) = [2, 12, 6, 9, 3] \).

Since the joint representations of one additive model still lie in the same semantic space with their original component vectors, it is natural to conduct cosine similarity to measure their semantic relationships. Thus, under a naive additive model, we have the following similarity equation:

\[
s(p, w) = \frac{p \cdot w}{\|p\| \cdot \|w\|} = \frac{(u + v)w}{\|u + v\| \cdot \|w\|} = \frac{\|u\|}{\|u + v\|} s(u, w) + \frac{\|v\|}{\|u + v\|} s(v, w),
\]

(3.9)

where \( w \) denotes any other word in the vocabulary and \( s \) indicates the similarity function. From derivation ahead, it could be concluded that this composition function composes both magnitude and directions of two component vectors. In other words, if one vector dominates the magnitude, it will also dominate the similarity. Furthermore, we have

\[
\|p\| = \|u + v\| \leq \|u\| + \|v\|.
\]

(3.10)

This lemma suggests that the semantic unit with a deeper-rooted parsing tree could determine the joint representation when combining with a shallow unit. Because the deeper the semantic unit is, the larger the magnitude it has.

Moreover, incorporating geometry insight, we can observe that the additive model builds a more solid understanding of semantic composition. Supposing that our component vectors are \( u \) and \( v \), the additive model aims to project them to \( x \) and \( y \), where \( x \) follows the direction of \( u \) while \( y \) is orthogonal to \( u \). The following figure could clearly illustrate this issue (Fig. 3.1).

\[\text{Fig. 3.1} \text{ An illustration of the additive model}\]
From the figure, the vector $x$ and the vector $y$ could be represented as

$$x = \frac{u \cdot v}{u \cdot u} \cdot u,$$

$$y = v - x = v - \frac{u \cdot v}{u \cdot u} \cdot u.$$

(3.11)

Then, using the linear combination of these two new vectors $x, y$ yields a new additive model:

$$p = \alpha x + \beta y$$

(3.12)

$$= \alpha \frac{u \cdot v}{u \cdot u} \cdot u + \beta \left( v - \frac{u \cdot v}{u \cdot u} \cdot u \right)$$

(3.13)

$$= (\alpha - \beta) \cdot \frac{u \cdot v}{u \cdot u} \cdot u + \beta v.$$

(3.14)

Furthermore, using cosine similarity measurement, the relationship could be written as follows:

$$s(p, w) = \frac{|\alpha - \beta|}{|\alpha|} s(u, w) + \frac{|\beta|}{|\alpha|} s(v, w).$$

(3.15)

From similarity measurement derivation, it is indicated that with this projection method, the composition similarity could be viewed as a linear combination of the similarities of two components, which means that combining semantic units with different semantic depths, the deeper one will not dominate the representation.

### 3.3.2 Multiplicative Model

Though the additive model achieves great success in semantic composition, the simplification it adopted may be too restrictive because it assumes all words, phrases, sentences, and documents are substantially similar enough to be represented in a unified semantic space. Different from the additive model which regards composition as a simple linear transformation, the multiplicative model aims to make higher order interaction. Among all models from this perspective, the most intuitive approach tried to apply the pair-wise product as a composition function approximation. In this method, the composition function is shown as the following:

$$p = u \odot v,$$

(3.16)

where, $p_i = u_i \cdot v_i$, which implies each dimension of the output only depends on the corresponding dimension of two input vectors. However, similar to the simplest additive model, this model is also suffering from the lack of the ability to model word order, and the absence from background syntactic or knowledge information.
In the additive model, we have $p = \alpha u + \beta v$ to alleviate the word order issue. Note that here $\alpha$ and $\beta$ are two scalars, which could be easily changed to two matrices. Therefore, the composition function could be represented as

$$p = W_\alpha \cdot u + W_\beta \cdot v,$$

(3.17)

where $W_\alpha$ and $W_\beta$ are matrices which determine the importance of $u$ and $v$ to $p$. With this expression, the composition could be more expressive and flexible although much harder to train.

Generalizing multiplicative model ahead, another approach is to utilize tensors as multiplicative descriptors and the composition function could be viewed as

$$p = \overrightarrow{W} \cdot uv,$$

(3.18)

where $\overrightarrow{W}$ denotes a 3-order tensor, i.e., the formula above could be written as $p_i = \sum_{i,j} W_{ijk} \cdot u_i \cdot v_j$. Hence, this model makes that each element of $p$ could be influenced by all elements of both $u$ and $v$, with a relationship of linear combination by assigning each $(i, j)$ a unique weight.

Starting from this simple but general baseline, some researchers proposed to make the function not symmetric to consider word order in the sequence. Paying more attention to the first element, the composition function could be

$$p = \overrightarrow{W} \cdot uuv,$$

(3.19)

where $\overrightarrow{W}$ denotes a 4-order tensor. This method could be understood as replacing linear transformation of $u$ and $v$ to a quadratic in $u$ asymmetrically. So this is a variant of the tensor multiplicative compositional model.

Different from expanding a simple multiplicative model to complex ones, other kinds of approaches are proposed to reduce the parameter space. With the reduction of parameter size, people could make compositions much more efficient rather than have an $O(n^3)$ time complexity in the tensor-based model. Thus, some compression techniques could be applied in the original tensor model. One representative instance is the circular convolution model, which could be shown as

$$p = u \odot v,$$

(3.20)

where $\odot$ represents the circular convolution operation with the following definition:

$$p_i = \sum_j u_j \cdot v_{i-j}.$$

(3.21)

If we assign each pair with unique weights, the composition function will be

$$p_i = \sum_j W_{ij} \cdot u_j \cdot v_{i-j}.$$

(3.22)
Note that the circular convolution model could be viewed as a special instance of a tensor-based composition model. If we write the circular convolution in the tensor form, we have $W_{ijk} = 0$, where $k \neq i + j$. Thus, the parameter number could be reduced from $n^3$ to $n^2$, while maintaining the interactions between each pair of dimensions in the input vectors.

Both in the additive and multiplicative models, the basic condition is all components lie in the same semantic space as the output. Nevertheless, different modeling types of words in different semantic spaces could bring us a different perspective. For instance, given $(u, v)$, the multiplicative model could be reformulated as

$$p = W \cdot (u \cdot v) = U \cdot v.$$  \hfill (3.23)

This implies that each left unit could be treated as an operation on the representation of the right one. In other words, each remaining unit could be formulated as a transformation matrix, while the right one should be represented as a semantic vector. This argument could be meaningful, especially for some kinds of phrase compositions. Reference [2] argues that for ADJ-NOUN phrases, the joint semantic information could be viewed as the conjunction of the semantic meanings of two components. Given a phrase red car, its semantic meaning is the conjunction of all red things and all different kinds of cars. Thus, red could be formulated as an operator on the vector of car, deriving the new semantic vector, which expressed the meaning of red car. These observations lead to another genre of semantic compositional modeling: semantic matrix-composition space.

3.4 N-Ary Composition

In real-world NLP tasks, the input is usually a sequence of multiple words rather than just a pair of words. Therefore, besides designing a suitable binary compositional operator, the order to apply binary operations is also important. In this section, we will introduce three mainstream strategies in N-ary composition by taking language modeling as an example.

To illustrate the language modeling task more clearly, the composition problem to model a sentence or even a document could be formulated as

Given a sentence/document consisting of a word sequence $\{w_0, w_1, w_2, \ldots, w_n\}$, we aim to design following functions to obtain the joint semantic representation of the whole sentence/document:

1. A semantic representation method like semantic vector space or compositional matrix space.
2. A binary compositional operation function $f(u, v)$ like we introduced in the previous sections. Here the input $u$ and $v$ denote the representations of two constitute semantic units, while the output is also the representation in the same space.
3. A sequential order to apply the binary function in step 2. To describe in detail, we could use a bracket to identify the order to apply the composition function. For instance, we could use \((w_1, w_2, w_3)\) to represent the sequential order from beginning to end.

In this section, we will introduce several systematic strategies to model sentence semantics by describing the solutions for the three problems above. We will classify the methods by word-level order: sequential order, recursive order (following parsing trees), and convolution order.

### 3.4.1 Recurrent Neural Network

To design orders to apply binary compositional functions, the most intuitive method is utilizing sequentiality. Namely, the sequence order should be \(s_n = (s_{n-1}, w_n)\), where \(s_{n-1}\) is the order of the first \(n - 1\) words. Motivated by this thought, the neural network model used is the Recurrent Neural Network (RNN).

An RNN applies the composition function sequentially and derives the representations of hidden semantic units. Based on these hidden semantic units, we could use them on some specific NLP tasks like sentiment analysis or text classification. Also, note that the basic RNN only utilizes the sequential information from head to tail of a sentence/document. To improve its representation ability, the RNN could be enhanced as bi-directional RNN by considering sequential and reverse-sequential information.

After deciding sequential order to model sentence-level semantics, the next question is determining the binary composition functions. In detail, supposing that \(h_t\) denotes the representation of the first \(t\) words and \(w_t\) represents the \(t\)th word, the general composition could be formulated as

\[
h_t = f(h_{t-1}, x_t),
\]

where \(f\) is a well-designed binary composition function.

From the definition of the RNN, the composition function could be formulated as follows:

\[
h_t = \tanh(W_1 h_{t-1} + W_2 w_t),
\]

where \(W_1\) and \(W_2\) are two weighted matrices.

We could see that here we use a matrix-weighted summation to represent binary semantic composition:

\[
p = W_\alpha u + W_\beta v.
\]

**LSTM.** Since the raw RNN only utilizes the simple tangent function, it is hard to obtain the long-term dependency of a long sentence/document. Reference [5] reinvents Long Short-Term Memory (LSTM) networks to strengthen the ability to
model long-term semantic dependency in RNN. In detail, the composition function of the LSTM allows information from previous layers to flow directly to their following layers. The composition function could be defined as

\[ f_t = \text{Sigmoid}(W^h_f h_{t-1} + W^x_f x_t + b_f), \]  
\[ i_t = \text{Sigmoid}(W^h_i h_{t-1} + W^x_i x_t + b_i), \]  
\[ o_t = \text{Sigmoid}(W^h_o h_{t-1} + W^x_o x_t + b_o), \]  
\[ \hat{c}_t = \tanh(W^h_c h_{t-1} + W^x_c x_t + b_c), \]  
\[ c_t = f_t \odot c_{t-1} + i_t \odot \hat{c}_t, \]  
\[ h_t = o_t \odot c_t. \]  

**Variants of LSTM.** To simplify LSTM and obtain more efficient algorithms, [3] proposes to utilize a simple but comparable RNN architecture, named Gated Recurrent Unit (GRU). Compared with LSTM, GRU has fewer parameters, which bring higher efficiency. The composition function is showed as

\[ z_t = \text{Sigmoid}(W^h_z h_{t-1} + W^x_z x_t + b_z), \]  
\[ r_t = \text{Sigmoid}(W^h_r h_{t-1} + W^x_r x_t + b_r), \]  
\[ \hat{h}_t = \tanh(W^h_z (r_t \odot h_{t-1}) + W^x_z x_t + b_h), \]  
\[ h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t. \]  

### 3.4.2 Recursive Neural Network

Besides the recurrent neural network, another strategy to apply binary compositional function follows a parsing tree instead of sequential word order. Based on this philosophy, [15] proposes a recursive neural network to model different levels of semantic units. In this subsection, we will introduce some algorithms following the recursive parsing tree with different binary compositional functions.

Since all the recursive neural networks are binary trees, the basic problem we need to consider is how to derive the representation of the father component on the tree given its two children semantic components. Reference [15] proposes a recursive matrix-vector model (MV-RNN) which captures constituent parsing tree structure information by assigning a matrix-vector representation for each constituent. The vector captures the meaning of the constituent itself, and the matrix represents how it modifies the meaning of the word it combines with. Suppose we have two children components \( a, b \) and their father component \( p \), the composition can be formulated as follows:
where \( \mathbf{a}, \mathbf{b}, \mathbf{p} \) are the embedding vectors for each component and \( \mathbf{A}, \mathbf{B}, \mathbf{P} \) are the matrices, \( \mathbf{W}_1 \) is a matrix that maps the transformed words into another semantic space, the element-wise function \( g \) is an activation function, and \( \mathbf{W}_2 \) is a matrix that maps the two matrices into one combined matrix \( \mathbf{P} \) with the same dimension. The whole process is illustrated in Fig. 3.2. And then MV-RNN selects the highest node of the path in the parse tree between the two target entities to represent the input sentence.

In fact, the composition operation used in the above recursive network is similar to an RNN unit introduced in the previous subsection. And the RNN unit here can be replaced by LSTM units or GRU units. Reference [16] proposes two types of tree-structured LSTMs including the Child-Sum Tree-LSTM and the N-ary Tree-LSTM to capture constituent or dependency parsing tree structure information. For the Child-Sum Tree-LSTM, given a tree, let \( C(t) \) denote the children set of the node \( t \). Its transition equations are defined as follows:

Fig. 3.2 The architecture of the matrix-vector recursive encoder
The N-ary Tree-LSTM has similar transition equations as the Child-Sum Tree-LSTM. The only difference is that it limits the tree structures to have at most N branches.

3.4.3 Convolutional Neural Network

Reference [6] proposes to embed an input sentence using a Convolutional Neural Network (CNN) which extracts local features by a convolution layer and combines all local features via a max-pooling operation to obtain a fixed-sized vector for the input sentence.

Formally, the convolution operation is defined as a matrix multiplication between a sequence of vectors, a convolution matrix $W$, and a bias vector $b$ with a sliding window. Let us define the vector $q_i$ as the concatenation of the subsequence of input representations in the $i$th window, we have

\[ h_j = \max_i [f(Wq_i + b)]_j, \]

where $f$ indicates a nonlinear function such as sigmoid or tangent function, and $h$ indicates the final representation of the sentence.

3.5 Summary

In this chapter, we first introduce the semantic space for compositional semantics. Afterwards, we take phrase representation as an example to introduce various models for binary semantic composition, including additive models and multiplicative models. Finally, we introduce typical models for N-ary semantic composition including recurrent neural network, recursive neural network, and convolutional neural network. Compositional semantics allows languages to construct complex meanings from the combinations of simpler elements, and its binary semantic composition
and N-ary semantic composition is the foundation of multiple NLP tasks including sentence representation, document representation, relational path representation, etc. We will give a detailed introduction to these scenarios in the following chapters.

For further understanding of compositional semantics, there are also some recommended surveys and books:

- Pelletier et al., The principle of semantic compositionality [13].
- Jeff et al., Composition in distributional models of semantics [10].

For better modeling compositional semantics, some directions require further efforts in the future:

1. **Neurobiology-inspired Compositional Semantics.** What is the neurobiology for dealing with compositional semantics in human language? Recently, [14] finds that the human combinatory system is related to rapidly peaking activity in the left anterior temporal lobe and later engagement of the medial prefrontal cortex. The analysis of how language builds meaning and lays out directions in neurobiological research may bring some instructive reference for modeling compositional semantics in representation learning. It is valuable to design novel compositional forms inspired by recent neurobiological advances.

2. **Combination of Symbolic and Distributed Representation.** Human language is inherently a discrete symbolic representation of knowledge. However, we represent the semantics of discrete symbols with distributed/distributional representations when dealing with natural language in deep learning. Recently, there are some approaches such as neural module networks [1] and neural symbolic machine [8] attempting to consider discrete symbols in neural networks. How to take advantage of these symbolic neural models to represent the composition of semantics is an open problem to be explored.

**References**

1. Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *Proceedings of CVPR*, pages 39–48, 2016.
2. Marco Baroni and Roberto Zamparelli. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of EMNLP*, 2010.
3. Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Gated feedback recurrent neural networks. In *Proceedings of ICML*, 2015.
4. Gottlob Frege. Die grundlagen derarithmetic. Eine logisch mathematische Untersuchung u’ber den Begriff der Zahl. Breslau: Koebner, 1884.
5. Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.
6. Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. In *Proceedings of ACL*, 2014.
7. George Lakoff. Linguistic gestalts. In *Proceedings of ILGISA*, 1977.
8. Chen Liang, Jonathan Berant, Quoc Le, Kenneth Forbus, and Ni Lao. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *Proceedings of ACL*, pages 23–33, 2017.
9. Jeff Mitchell and Mirella Lapata. Vector-based models of semantic composition. In *Proceedings of ACL*, 2008.
10. Jeff Mitchell and Mirella Lapata. Composition in distributional models of semantics. *Cognitive science*, 34(8):1388–1429, 2010.
11. Geoffrey Nunberg, Ivan A Sag, and Thomas Wasow. Idioms. *Language*, pages 491–538, 1994.
12. Barbara Partee. Lexical semantics and compositionality. *An Invitation to Cognitive Science: Language*, 1:311–360, 1995.
13. Francis Jeffry Pelletier. The principle of semantic compositionality. *Topoi*, 13(1):11–24, 1994.
14. Liina Pylkkänen. The neural basis of combinatory syntax and semantics. *Science*, 366(6461):62–66, 2019.
15. Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of EMNLP*, 2012.
16. Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of ACL*, 2015.

Open Access  This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter’s Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter’s Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.