Chapter 2
Word Representation

Abstract Word representation, aiming to represent a word with a vector, plays an essential role in NLP. In this chapter, we first introduce several typical word representation learning methods, including one-hot representation and distributed representation. After that, we present two widely used evaluation tasks for measuring the quality of word embeddings. Finally, we introduce the recent extensions for word representation learning models.

2.1 Introduction

Words are usually considered as the smallest meaningful units of speech or writing in human languages. High-level structures in a language, such as phrases and sentences, are further composed of words. For human beings, to understand a language, it is crucial to understand the meanings of words. Therefore, it is essential to accurately represent words, which could help models better understand, categorize, or generate text in NLP tasks.

A word can be naturally represented as a sequence of several characters. However, it is very inefficient and ineffective only to use raw character sequences to represent words. First, the variable lengths of words make it hard to be processed and used in machine learning methods. Moreover, it is very sparse, because only a tiny proportion of arrangements are meaningful. For example, English words are usually character sequences which are composed of 1–20 characters in the English alphabet, but most of these character sequences such as “aaaaa” are meaningless.

One-hot representation is another natural approach to represent words, which assigns a unique index to each word. It is also not good enough to represent words with one-hot representation. First, one-hot representation could not capture the semantic relatedness among words. Second, one-hot representation is a high-dimensional sparse representation, which is very inefficient. Third, it is very inflexible for one-hot representation to deal with new words, which requires assigning new indexes for new words and would change the dimensions of the representation. The change may lead to some problems for existing NLP systems.

Z. Liu et al., *Representation Learning for Natural Language Processing*,
https://doi.org/10.1007/978-981-15-5573-2_2

Recently, distributed word representation approaches are proposed to address the problem of one-hot word representation. The distributional hypothesis [23, 30] that linguistic objects with similar distributions have similar meanings is the basis for distributed word representation learning. Based on the distributional hypothesis, various word representation models, such as CBOW and Skip-gram, have been proposed and applied in different areas.

In the remaining part of this chapter, we start with one-hot word representation. Further, we introduce distributed word representation models, including Brown Cluster, Latent Semantic Analysis, Word2vec, and GloVe in detail. Then we introduce two typical evaluation tasks for word representation. Finally, we discuss various extensions of word representation models.

2.2 One-Hot Word Representation

In this section, we will introduce one-hot word representation in details. Given a fixed set of vocabulary $V = \{w_1, w_2, \dots, w_{|V|}\}$, one very intuitive way to represent a word w is to encode it with a $|V|$-dimensional vector $\mathbf{w}$, where each dimension of w is either 0 or 1. Only one dimension in $\mathbf{w}$ can be 1 while all the other dimensions are 0. Formally, each dimension of $\mathbf{w}$ can be represented as

$$\mathbf{w}_i = \begin{cases} 1 & \text{if } w = w_i \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

One-hot word representation, in essence, maps each word to an index of the vocabulary, which can be very efficient for storage and computation. However, it does not contain rich semantic and syntactic information of words. Therefore, one-hot representation cannot capture the relatedness among words. The difference between `cat` and `dog` is as much as the difference between `cat` and `bed` in one-hot word representation. Besides, one-hot word representation embeds each word into a $|V|$-dimensional vector, which can only work for a fixed vocabulary. Therefore, it is inflexible to deal with new words in a real-world scenario.

2.3 Distributed Word Representation

Recently, distributed word representation approaches are proposed to address the problem of one-hot word representation. The distributional hypothesis [23, 30] that linguistic objects with similar distributions have similar meanings is the basis for semantic word representation learning.

Based on the distributional hypothesis, Brown Cluster [9] groups words into hierarchical clusters where words in the same cluster have similar meanings. The cluster label can roughly represent the similarity between words, but it cannot precisely compare words in the same group. To address this issue, distributed word representation[1] aims to embed each word into a continuous real-valued vector. It is a dense representation, and "dense" means that one concept is represented by more than one dimension of the vector, and each dimension of the vector is involved in representing multiple concepts. Due to its continuous characteristic, distributed word representation can be easily applied in deep neural models for NLP tasks. Distributed word representation approaches such as Word2vec and GloVe usually learn word vectors from a large corpus based on the distributional hypothesis. In this section, we will introduce several distributed word representation approaches in detail.

2.3.1 Brown Cluster

Brown Cluster classifies words into several clusters that have similar semantic meanings. Detailedly, Brown Cluster learns a binary tree from a large-scale corpus, in which the leaves of the tree indicate the words and the internal nodes of the tree indicate word hierarchical clusters. This is a hard clustering method since each word belongs to exactly one group.

The idea of Brown Cluster to cluster the words comes from the n-gram language model. A language model evaluates the probability of a sentence. For example, the sentence `have a nice day` should have a higher probability than a random sequence of words. Using a k-gram language model, the probability of a sentence $s = \{w_1, w_2, w_3, \ldots, w_n\}$ can be represented as

$$P(s) = \prod_{i=1}^{n} P(w_i|\mathbf{w}_{i-k}^{i-1}). \tag{2.2}$$

It is easy to estimate $P(w_i|\mathbf{w}_{i-k}^{i-1})$ from a large corpus, but the model has $|V|^k - 1$ independent parameters which is a huge number for computers in the 1990s. Even if k is 2, the number of parameters is considerable. Moreover, the estimation is inaccurate for rare words. To address these problems, [9] proposes to group words into clusters and train a cluster-level n-gram language model rather than a word-level model. By assigning a cluster to each word, the probability can be written as

$$P(s) = \prod_{i=1}^{n} P(c_i|\mathbf{c}_{i-k}^{i-1})P(w_i|c_i), \tag{2.3}$$

[1]We emphasize that distributed representation and distributional representation are two completely different aspects of representations. A word representation method may belong to both categories. Distributed representation indicates that the representation is a real-valued vector, while distributional representation indicates that the meaning of a word is learned under the distributional hypothesis.

where c_i is the corresponding cluster of w_i. In cluster-level language model, there are only $|C^k| - 1 + |V| - |C|$ independent parameters, where C is the cluster set which is usually much smaller than the vocabulary $|V|$.

The quality of the cluster affects the performance of the language model. Given a training text s, for a 2-gram language model, the quality of a mapping π from words to clusters is defined as

$$Q(\pi) = \frac{1}{n} \log P(s) \tag{2.4}$$

$$= \frac{1}{n} \sum_{i=1}^{n} \big(\log P(c_i|c_{i-1}) + \log P(w_i|c_i) \big). \tag{2.5}$$

Let N_w be the number of times word w appears in corpus s, $N_{w_1 w_2}$ be the number of times bigram $w_1 w_2$ appears, and $N_{\pi(w)}$ be the number of times a cluster appears. Then the quality function $Q(\pi)$ can be rewritten in a statistical way as follows:

$$Q(\pi) = \frac{1}{n} \sum_{i=1}^{n} \big(\log P(c_i|c_{i-1}) + \log P(w_i|c_i) \big) \tag{2.6}$$

$$= \sum_{w_1, w_2} \frac{N_{w_1 w_2}}{n} \log P(\pi(w_2)|\pi(w_1)) P(w_2|\pi(w_2)) \tag{2.7}$$

$$= \sum_{w_1, w_2} \frac{N_{w_1 w_2}}{n} \log \frac{N_{\pi(w_1)\pi(w_2)}}{N_{\pi(w_1)}} \frac{N_{w_2}}{N_{\pi(w_2)}} \tag{2.8}$$

$$= \sum_{w_1, w_2} \frac{N_{w_1 w_2}}{n} \log \frac{N_{\pi(w_1)\pi(w_2)} n}{N_{\pi(w_1)} N_{\pi(w_2)}} + \sum_{w_1, w_2} \frac{N_{w_1 w_2}}{n} \log \frac{N_{w_2}}{n} \tag{2.9}$$

$$= \sum_{c_1, c_2} \frac{N_{c_1 c_2}}{n} \log \frac{N_{c_1 c_2} n}{N_{c_1} N_{c_2}} + \sum_{w_2} \frac{N_{w_2}}{n} \log \frac{N_{w_2}}{n}. \tag{2.10}$$

Since $P(w) = \frac{N_w}{n}$, $P(c) = \frac{N_c}{n}$, and $P(c_1 c_2) = \frac{N_{c_1 c_2}}{n}$, the quality function can be rewritten as

$$Q(\pi) = \sum_{c_1, c_2} P(c_1 c_2) \log \frac{P(c_1 c_2)}{P(c_1) P(c_2)} + \sum_{w} P(w) \log P(w) \tag{2.11}$$

$$= I(C) - H(V), \tag{2.12}$$

where $I(C)$ is the mutual information between clusters and $H(V)$ is the entropy of the word distribution, which is a constant value. Therefore, to optimize $Q(\pi)$ equals to optimize the mutual information.

There is no practical method to obtain optimum partitions. Nevertheless, Brown Cluster uses a greedy strategy to obtain a suboptimal result. Initially, it assigns a distinct class for each word. Then it merges two classes with the least average mutual information. After $|V| - |C|$ mergences, the partition is generated. Keeping the $|C|$

Table 2.1 Some clusters of Brown Cluster

Cluster #1	Friday	Monday	Thursday	Wednesday	Tuesday	Saturday
Cluster #2	June	March	July	April	January	December
Cluster #3	Water	Gas	Coal	Liquid	Acid	Sand
Cluster #4	Great	Big	Vast	Sudden	Mere	Sheer
Cluster #5	Man	Woman	Boy	Girl	Lawyer	Doctor
Cluster #6	American	Indian	European	Japanese	German	African

clusters, we can continuously perform $|C| - 1$ mergences to get a binary tree. With certain care in implementation, the complexity of this algorithm is $O(|V|^3)$.

We show some clusters in Table 2.1. From the table, we can find that each cluster relates to a sense in the natural language. The words in the same cluster tend to express similar meanings or could be used exchangeably.

2.3.2 *Latent Semantic Analysis*

Latent Semantic Analysis (LSA) is a family of strategies derived from vector space models, which could capture word semantics much better. LSA aims to explore latent factors for words and documents by matrix factorization to improve the estimation of word similarities. Reference [14] applies Singular Value Decomposition (SVD) on the word-document matrix and exploits uncorrelated factors for both words and documents. The SVD of word-document matrix $\mathbf{M}$ yields three matrices $\mathbf{E}$, Σ and $\mathbf{D}$ such that

$$\mathbf{M} = \mathbf{E}\Sigma\mathbf{D}^\top, \tag{2.13}$$

where Σ is the diagonal matrix of singular values of $\mathbf{M}$, each row vector $\mathbf{w}_i$ in matrix $\mathbf{E}$ corresponds to word w_i, and each row vector $\mathbf{d}_i$ in matrix $\mathbf{D}$ corresponds to document d_i. Then the similarity between two words could be

$$\text{sim}(w_i, w_j) = \mathbf{M}_{i,:}\mathbf{M}_{j,:}^\top = \mathbf{w}_i\Sigma^2\mathbf{w}_j. \tag{2.14}$$

Here, the number of singular values k included in Σ is a hyperparameter that needs to be tuned. With a reasonable amount of the largest singular values used, LSA could capture much useful information in the word-document matrix and provide a smoothing effect that prevents large variance.

With a relatively small k, once the matrices $\mathbf{E}$, Σ and $\mathbf{D}$ are computed, measuring word similarity could be very efficient because there are often fewer nonzero dimensions in word vectors. However, the computation of $\mathbf{E}$ and $\mathbf{D}$ can be costly because full SVD on a $n \times m$ matrix takes $O(\min\{m^2n, mn^2\})$ time, while the parallelization of SVD is not trivial.

Another algorithm for LSA is Random Indexing [34, 55]. It overcomes the difficulty of SVD-based LSA, by avoiding costly preprocessing of a huge *word-document* matrix. In random indexing, each document is assigned with a randomly generated high-dimensional sparse ternary vector (called *index vector*). Then for each word in the document, the *index vector* is added to the word's vector. The *index vectors* are supposed to be orthogonal or nearly orthogonal. This algorithm is simple and scalable, which is easy to parallelize and implemented incrementally. Moreover, its performance is comparable with the SVD-based LSA, according to [55].

2.3.3 Word2vec

Google's word2vec[2] toolkit was released in 2013. It can efficiently learn word vectors from a large corpus. The toolkit has two models, including Continuous Bag-Of-Words (CBOW) and Skip-gram. Based on the assumption that the meaning of a word can be learned from its context, CBOW optimizes the embeddings so that they can predict a target word given its context words. Skip-gram, on the contrary, learns the embeddings that can predict the context words given a target word. In this section, we will introduce these two models in detail.

2.3.3.1 Continuous Bag-of-Words

CBOW predicts the center word given a window of context. Figure 2.1 shows the idea of CBOW with a window of 5 words.

Formally, CBOW predicts w_i according to its contexts as

$$P(w_i | w_{j(|j-i| \le l, j \ne i)}) = \text{Softmax}\left(\mathbf{M}\left(\sum_{|j-i| \le l, j \ne i} \mathbf{w}_j\right)\right), \tag{2.15}$$

where $P(w_i | w_{j(|j-i| \le l, j \ne i)})$ is the probability of word w_i given its contexts, l is the size of training contexts, $\mathbf{M}$ is the weight matrix in $\mathbb{R}^{|V| \times m}$, V indicates the vocabulary, and m is the dimension of the word vector.

The CBOW model is optimized by minimizing the sum of negative log probabilities:

$$\mathscr{L} = -\sum_i \log P(w_i | w_{j(|j-i| \le l, j \ne i)}). \tag{2.16}$$

Here, the window size l is a hyperparameter to be tuned. A larger window size may lead to a higher accuracy as well as the more expense of the training time.

[2] https://code.google.com/archive/p/word2vec/.

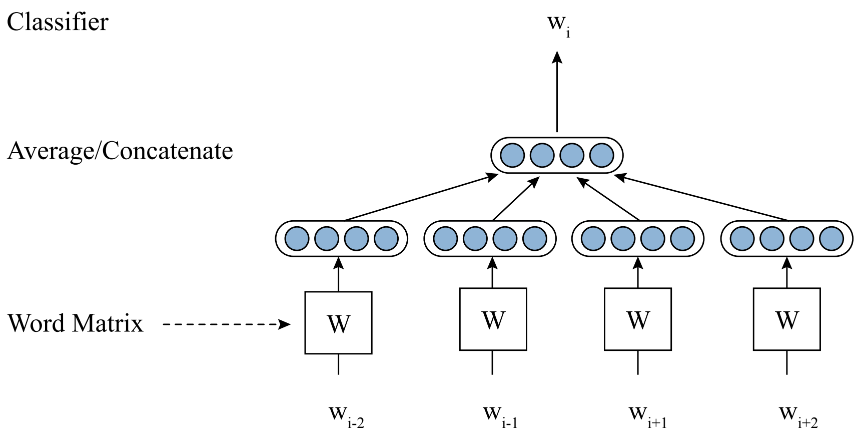

Fig. 2.1 The architecture of CBOW model

2.3.3.2 Skip-Gram

On the contrary to CBOW, Skip-gram predicts the context given the center word. Figure 2.2 shows the model.

Formally, given a word w_i, Skip-gram predicts its context as

$$P(w_j|w_i) = \text{softmax}(\mathbf{M}\mathbf{w}_i)\big(|j-i| \le l, j \ne i\big), \tag{2.17}$$

where $P(w_j|w_i)$ is the probability of context word w_j given w_i, and $\mathbf{M}$ is the weight matrix. The loss function is defined similar to CBOW as

$$\mathscr{L} = -\sum_i \sum_{j(|j-i|\le l, j\ne i)} P(w_j|w_i). \tag{2.18}$$

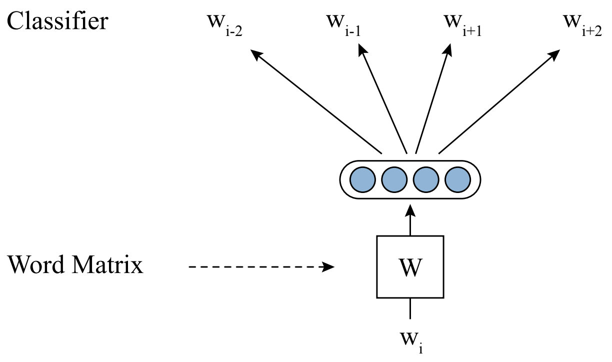

Fig. 2.2 The architecture of skip-gram model

2.3.3.3 Hierarchical Softmax and Negative Sampling

To train CBOW or Skip-gram directly is very time consuming. The most time-consuming part is the softmax layer. The conventional softmax layer needs to obtain the scores of all words even though only one word is used in computing the loss function. An intuitive idea to improve efficiency is to get a reasonable but much faster approximation of that word. Here, we will introduce two typical approximation methods which are included in the toolkit, including hierarchical softmax and negative sampling. We explain these two methods using CBOW as an example.

The idea of hierarchical softmax is to build hierarchical classes for all words and to estimate the probability of a word by estimating the conditional probability of its corresponding hierarchical class. Figure 2.3 gives an example. Each internal node of the tree indicates a hierarchical class and has a feature vector, while each leaf node of the tree indicates a word. In this example, the probability of word `the` is $p_0 \times p_{01}$ while the probability of `cat` is $p_0 \times p_{00} \times p_{001}$. The conditional probability is computed by the feature vector of each node and the context vector. For example,

$$p_0 = \frac{\exp(\mathbf{w}_0 \cdot \mathbf{w}_c)}{\exp(\mathbf{w}_0 \cdot \mathbf{w}_c) + \exp(\mathbf{w}_1 \cdot \mathbf{w}_c)}, \tag{2.19}$$

$$p_1 = 1 - p_0, \tag{2.20}$$

where $\mathbf{w}_c$ is the context vector, $\mathbf{w}_0$ and $\mathbf{w}_1$ are the feature vectors.

Hierarchical softmax generates the hierarchical classes according to the word frequency, i.e., a Huffman tree. By the approximation, it can compute the probability of each word much faster, and the complexity of calculating the probability of each word is $O(\log |V|)$.

Negative sampling is more straightforward. To calculate the probability of a word, negative sampling directly samples k words as negative samples according to the word frequency. Then, it computes a softmax over the $k + 1$ words to approximate the probability of the target word.

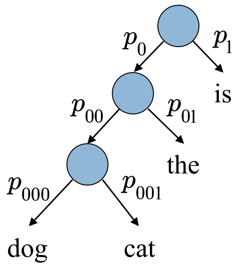

Fig. 2.3 An illustration of hierarchical softmax

Table 2.2 Co-occurrence probabilities and the ratio of probabilities for target words `ice` and `steam` with context word `solid`, `gas`, `water`, and `fashion`

Probability and ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	$1.9e-4$	$6.6e-5$	$3e-3$	$1.7e-5$
$P(k\|steam)$	$2.2e-5$	$7.8e-4$	$2.2e-3$	$1.8e-5$
$P(k\|ice)/P(k\|steam)$	8.9	$8.5e-2$	1.36	0.96

2.3.4 GloVe

Methods like Skip-gram and CBOW are shallow window-based methods. These methods scan a context window across the entire corpus, which fails to take advantage of some global information. Global Vectors for Word Representation (GloVe), on the contrary, can capture corpus statistics directly.

As shown in Table 2.2, the meaning of a word can be learned from the co-occurrence matrix. The ratio of co-occurrence probabilities can be especially useful. In the example, the meaning of `ice` and `water` can be examined by studying the ratio of their co-occurrence probabilities with various probe words. For words related to `ice` but not `steam`, for example, `solid`, the ratio $P(\texttt{solid}|\texttt{ice})/P(\texttt{solid}|\texttt{steam})$ will be large. Similarly, `gas` is related to `steam` but not `ice`, so $P(\texttt{gas}|\texttt{ice})/P(\texttt{gas}|\texttt{steam})$ will be small. For words that are relevant or irrelevant to both words, the ratio is close to 1.

Based on this idea, GloVe models

$$F(\mathbf{w}_i, \mathbf{w}_j, \tilde{\mathbf{w}}_k) = \frac{P_{ik}}{P_{jk}}, \tag{2.21}$$

where $\tilde{\mathbf{w}} \in \mathbb{R}^d$ are separate context word vectors, and P_{ij} is the probability of word j to be in the context of word i, formally

$$P_{ij} = \frac{N_{ij}}{N_i}, \tag{2.22}$$

where N_{ij} is the number of occurrences of word j in the context of word i, and $N_i = \sum_k N_{ik}$ is the number of times any word appears in the context of word j.

$F(\cdot)$ is supposed to encode the information presented in the ratio P_{ik}/P_{jk} in the word vector space. To keep the inherently linear structure, F should only depend on the difference of two target words

$$F(\mathbf{w}_i - \mathbf{w}_j, \tilde{\mathbf{w}}_k) = \frac{P_{ik}}{P_{jk}}. \tag{2.23}$$

The arguments of F are vectors while the right side of the equation is a scalar, to avoid F obfuscating the linear structure, a dot product is used:

$$F\big((\mathbf{w}_i - \mathbf{w}_j)^\top \tilde{\mathbf{w}}_k\big) = \frac{P_{ik}}{P_{jk}}. \tag{2.24}$$

The model keeps the invariance under relabeling the target word and context word. It requires F to be a homomorphism between the groups $(\mathbb{R}, +)$ and $(\mathbb{R}_{>0}, \times)$. The solution is $F = \exp$. Then

$$\mathbf{w}_i^\top \tilde{\mathbf{w}}_k = \log N_{ik} - \log N_i. \tag{2.25}$$

To keep exchange symmetry, $\log N_i$ is eliminated by adding biases b_i and $\tilde{b}_k$. The model becomes

$$\mathbf{w}_i^\top \tilde{\mathbf{w}}_k + b_i + \tilde{b}_k = \log N_{ik}, \tag{2.26}$$

which is significantly simpler than Eq. (2.21).

The loss function is defined as

$$\mathscr{L} = \sum_{i,j=1}^{|V|} f(N_{ij})(\mathbf{w}_i^\top \tilde{\mathbf{w}}_j + b_i + \tilde{b}_j - \log N_{ij}), \tag{2.27}$$

where $f(\cdot)$ is a weighting function:

$$f(x) = \begin{cases} (x/x_{max})^\alpha & \text{if } x < x_{max}, \\ 1 & \text{otherwise.} \end{cases} \tag{2.28}$$

2.4 Contextualized Word Representation

In natural language, the meaning of an individual word usually relates to its context in a sentence. For example,

- `The central bank has slashed its forecast for economic growth this year from 4.1 to 2.6%.`
- `More recently, on a blazing summer day, he took me back to one of the den sites, in a slumping bank above the South Saskatchewan River.`

In these two sentences, although the word `bank` is always the same, their meanings are different. However, most of the traditional word embeddings (CBOW, Skip-gram, GloVe, etc.) cannot well understand the different nuances of the meanings of words with the different surrounding texts. The reason is that these models only learn a unique representation for each word, and therefore it is impossible for these models to capture how the meanings of words change based on their surrounding contexts.

To address this issue, [48] proposes ELMo, which uses a deep, bidirectional LSTM model to build word representations. ELMo could represent each word depending on the entire context in which it is used. More specifically, rather than having a look-up table of word embedding matrix, ELMo converts words into low-dimensional vectors on-the-fly by feeding the word and its surrounding text into a deep neural network. ELMo utilizes a bidirectional language model to conduct word representation. Formally, given a sequence of N words, $(w_1, w_2, \ldots, w_N)$, a forward language model (LM, the details of language model are in Sect. 4) models the probability of the sequence by predicting the probability of each word t_k according to the historical context:

$$P(w_1, w_2, \ldots, w_N) = \prod_{k=1}^{N} P(w_k \mid w_1, w_2, \ldots, w_{k-1}). \tag{2.29}$$

The forward LM in ELMo is a multilayer LSTM, and the jth layer of the LSTM-based forward LM will generate the context-dependent word representation $\overrightarrow{\mathbf{h}}_{k,j}^{LM}$ for the word w_k. The backward LM is similar to the forward LM. The only difference is that it reverses the input word sequence to $(w_N, w_{N-1}, \ldots, w_1)$ and predicts each word according to the future context:

$$P(w_1, w_2, \ldots, w_N) = \prod_{k=1}^{N} P(w_k \mid w_{k+1}, w_{k+2}, \ldots, w_N). \tag{2.30}$$

As the same as the forward LM, the jth backward LM layer generates the representations $\overleftarrow{\mathbf{h}}_{k,j}^{LM}$ for the word w_k.

ELMo generates a task-specific word representation, which combines all layer representations of the bidirectional LM. Formally, it computes a task-specific weighting of all bidirectional LM layers:

$$\mathbf{w}_k = \alpha^{task} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j}^{LM}, \tag{2.31}$$

where $\mathbf{s}^{task}$ are softmax-normalized weights and α^{task} is the weight of the entire word vector for the task.

2.5 Extensions

Besides those very popular toolkits, such as word2vec and GloVe, various works are focusing on different aspects of word representation, contributing to numerous extensions. These extensions usually focus on the following directions.

2.5.1 Word Representation Theories

With the success of word representation, researchers begin to explore the theories of word representation. Some works attempt to give more theoretical analysis to prove the reasonability of existing tricks on word representation learning [39, 45], while some works try to discuss the new learning methods [26, 61].

Reasonability. Word2vec and other similar tools are empirical methods of word representation learning. Many tricks are proposed in [43] to learn the representation of words from a large corpus efficiently, for example, negative sampling. Considering the effectiveness of these methods, a more theoretical analysis should be done to prove the reasonability of these tricks. Reference [39] gives some theoretical analysis of these tricks. They formalize the Skip-gram model with negative sampling as an implicit matrix factorization process. The Skip-gram model generates a word embedding matrix $\mathbf{E}$ and a context matrix $\mathbf{C}$. The size of the word embedding matrix $\mathbf{E}$ is $|V| \times m$. Each row of context matrix $\mathbf{C}$ is a context word's m-dimensional vector. The training process of Skip-gram is an implicit factorization of $\mathbf{M} = \mathbf{E}\mathbf{C}^\top$. $\mathbf{C}$ is not explicitly considered in word2vec. This work further analyzes that the matrix $\mathbf{M}$ is

$$\mathbf{M}_{ij} = \mathbf{w}_i \cdot \mathbf{c}_j = \text{PMI}(w_i, c_j) - \log k, \tag{2.32}$$

where k is the number of negative samples, $\text{PMI}(w, c)$ is the point-wise mutual information

$$\text{PMI}(w, c) = \log \frac{P(w, c)}{P(w)P(c)}. \tag{2.33}$$

The shifted PMI matrix can directly be used to compare the similarity of words. Another intuitive idea is to factorize the shifted PMI matrix directly. Reference [39] evaluates the performance of using the SVD matrix factorization method on the implicit matrix $\mathbf{M}$. Matrix factorization achieves significantly better objective value when the embedding size is smaller than 500 dimensions and the number of negative samples is 1. With more negative samples and higher embedding dimensions, Skip-gram with negative sampling gets better objective value. This is because when the number of zeros increases in $\mathbf{M}$, and SVD prefers to factorize a matrix with minimum values. With 1,000 dimensional embeddings and different numbers of negative samples in $\{1, 5, 15\}$, SVD achieves slightly better performance on word analogy and word similarity. In contrast, Skip-gram with negative sampling achieves 2% better performance on syntactical analogy.

Interpretability. Most existing distributional word representation methods could generate a dense real-valued vector for each word. However, the word embeddings obtained by these models are hard to be interpreted. Reference [26] introduces non-negative and sparsity embeddings, where the models are interpretable and each dimension indicates a unique concept. This method factorizes the corpus statistics matrix $\mathbf{X} \in \mathbb{R}^{|V|\times|D|}$ into a word embedding matrix $\mathbf{E} \in \mathbb{R}^{|V|\times m}$ and a document statistics matrix $\mathbf{D} \in \mathbb{R}^{m\times|D|}$. Its training objective is

$$\begin{aligned}
&\arg\min_{\mathbf{E},\mathbf{D}} \frac{1}{2}\sum_{i=1}^{|V|} \|\mathbf{X}_{i,:} - \mathbf{E}_{i,:}\mathbf{D}\|_2 + \lambda\|\mathbf{E}_{i,:}\|_1,\\
&\text{s.t. } \mathbf{D}_{i,:}\mathbf{D}_{i,:}^{\top} \le 1, \forall 1 \le i \le m,\\
&\quad \mathbf{E}_{i,j} \ge 0, 1 \le i \le |V|, 1 \le j \le m.
\end{aligned} \tag{2.34}$$

By iteratively optimizing **E** and **D** via gradient descent, this model can learn nonnegative and sparse embeddings for words. Since the embeddings are sparse and nonnegative, words with the highest scores in each dimension show high similarity, which can be viewed as a concept of this dimension. To further improve the embeddings, this work also proposes phrasal-level constraints into the loss function. With new constraints, it could achieve both interpretability and compositionality.

2.5.2 Multi-prototype Word Representation

Using only one single vector to represent a word is problematic due to the ambiguity of words. A single vector cannot represent multiple meanings of a word well because it may lead to semantic confusion among the different senses of this word.

The multi-prototype vector space model [51] is proposed to better represent different meanings of a word. In multi-prototype vector space model, a mixture of von Mises-Fisher distributions (movMF) clustering method with first-order unigram contexts [5] is used to cluster different meanings of a word. Formally, it assigns a different word representation $\mathbf{w}_i(x)$ to the same word x in each different cluster i. When the multi-prototype embedding is used, the similarity between two words x, y is computed straightforwardly. If contexts of words are not available, the similarity between two words is defined as

$$\text{AvgSim}(x, y) = \frac{1}{K^2}\sum_{i=1}^{K}\sum_{j=1}^{K} s(\mathbf{w}_i(x), \mathbf{w}_j(y)), \tag{2.35}$$

$$\text{MaxSim}(x, y) = \max_{1\le i,j\le K} s(\mathbf{w}_i(x), \mathbf{w}_j(y)), \tag{2.36}$$

where K is a hyperparameter indicating the number of the clusters and $s(\cdot)$ is a similarity function of two vectors such as cosine similarity. When contexts are available, the similarity can be computed more precisely as:

$$\text{AvgSimC}(x, y) = \frac{1}{K^2}\sum_{i=1}^{K}\sum_{j=1}^{K} s_{c,x,i}s_{c,y,j}s(\mathbf{w}_i(x), \mathbf{w}_j(y)), \tag{2.37}$$

$$\text{MaxSimC}(x, y) = s(\hat{\mathbf{w}}(x), \hat{\mathbf{w}}(y)), \tag{2.38}$$

where $s_{c,x,i} = s(\mathbf{w}_i(c), \mathbf{w}_i(x))$ is the likelihood of context c belonging to cluster i, and $\hat{\mathbf{w}}(x) = \mathbf{w}_{\arg\max_{1\le i\le K} s_{c,x,i}}(x)$ is the maximum likelihood cluster for x in context c. With multi-prototype embeddings, the accuracy on the word similarity task is significantly improved, but the performance is still sensitive to the number of clusters.

Although the multi-prototype embedding method can effectively cluster different meanings of a word via its contexts, the clustering is offline, and the number of clusters is fixed and needs to be predefined. It is difficult for a model to select an appropriate amount of meanings for different words, to adapt to new senses, new words, or new data, and to align the senses with prototypes. To address these problems, [12] proposes a unified model for word sense representation and word sense disambiguation. This model uses available knowledge bases such as WordNet [46] to determine the senses of a word. Each word and each sense had a single vector and are trained jointly. This model can learn representations of both words and senses, and two simple methods are proposed to do disambiguation using the word and sense vectors.

2.5.3 *Multisource Word Representation*

There is much information about words that can be leveraged to improve the quality of word representations. We will introduce other kinds of word representation learning methods utilizing multisource information.

2.5.3.1 Word Representation with Internal Information

There is much information locating inside words, which can be utilized to improve the quality of word representations further.

Using Character Information. Many languages such as Chinese and Japanese have thousands of characters, and the words in these languages are composed of several characters. Characters in these languages have richer semantic information comparing with other languages containing only dozens of characters. Hence, the meaning of a word can not only be learned from its contexts but also the composition of characters. Driven by this intuitive idea, [13] proposes a joint learning model for Character and Word Embeddings (CWE). In CWE, a word is a composition of a word embedding and its character embeddings. Formally,

$$\mathbf{x} = \mathbf{w} + \frac{1}{|w|}\sum_i \mathbf{c}_i, \tag{2.39}$$

where $\mathbf{x}$ is the representation of a word, which is the composition of a word vector $\mathbf{w}$ and several character vectors $\mathbf{c}_i$, and $|w|$ is the number of characters in the word. Note that this model can be integrated with various models such as Skip-gram, CBOW, and GloVe.

Further, position-based and cluster-based methods are proposed to address this issue that characters are highly ambiguous. In position-based approach, each character is assigned three vectors which appear in *begin*, *middle* and *end* of a word respectively. Since the meaning of a character varies when it appears in the different positions of a word, this method can significantly resolve the ambiguity problem. However, characters that appear in the same position may also have different meanings. In the cluster-based method, a character is assigned K different vectors for its different meanings, in which a word's context is used to determine which vector to be used.

By introducing character embeddings, the representation of low-frequency words can be significantly improved. Besides, this method can deal with new words while other methods fail. Experiments show that the joint learning method can achieve better performance on both word similarity and word analogy tasks. By disambiguating characters using the position-based and cluster-based method, it can further improve the performance.

Using Morphology Information. Many languages such as English have rich morphology information and plenty of rare words. Most word representation models assign a distinct vector to each word ignoring the rich morphology information. This is a limitation because the affixes of a word can help infer the meaning of a word and the morphology information of word is essential especially when facing rare contexts.

To address this issue, [8] proposes to represent a word as a bag of morphology n-grams. This model substitutes word vectors in Skip-gram with the sum of morphology n-gram vectors. When creating the dictionary of n-grams, they select all n-grams with a length greater or equal than 3 and smaller or equal than 6. To distinguish prefixes and suffixes with other affixes, they also add special characters to indicate the beginning and the end of a word. This model is efficient and straightforward, which achieves good performance on word similarity and word analogy tasks especially when the training set is small.

Reference [41] further uses a bidirectional LSTM to generate word representation by composing morphologies. This model does not use a look-up table to assign a distinct vector to each word like what those independent word embedding methods are doing. Hence, this model not only significantly reduces the number of parameters but also addresses some disadvantages of independent word embeddings. Moreover, the embeddings of words in this model could affect each other.

2.5.3.2 Word Representation with External Knowledge

Besides internal information of words, there is much external knowledge that could help us learn the word representations.

Using Knowledge Base. Some languages have rich internal information, whereas people have also annotated lots of knowledge bases which can be used in word representation learning to constrain embeddings. Reference [62] introduces relation constraints into the CBOW model. With these constraints, the embeddings can not only predict its contexts, but also predict words with relations. The objective is to maximize the sum of log probability of all relations as

$$\mathscr{O} = \frac{1}{N} \sum_{i=1}^{N} \sum_{w \in R_{w_i}} \log P(w|w_i), \tag{2.40}$$

where R_{w_i} indicates a set of words which have relation with w_i. Then the joint objective is defined as

$$\mathscr{O} = \frac{1}{N} \sum_{i=1}^{N} \log P(w_i|w_{j(|j-i|<l, j \neq i)}) + \frac{\beta}{N} \sum_{i=1}^{N} \sum_{w \in R_{w_i}} \log p(w|w_i), \tag{2.41}$$

where β is a hyperparameter. The external information helps to train a better word representation, which shows significant improvements on word similarity benchmarks.

Moreover, Retrofitting [19] introduces a post-processing step which can introduce knowledge bases into word representation learning. It is more modular than other approaches which consider knowledge base during training. Let the word embeddings learned by existing word representation approaches be $\mathbf{E}$. Retrofitting attempts to find another embedding space $\hat{\mathbf{E}}$, which is close to $\mathbf{E}$ but considers the relations in the knowledge base. Formally,

$$\mathscr{L} = \sum_{i} \big(\alpha_i \|\mathbf{w}_i - \hat{\mathbf{w}}_i\|_2 + \sum_{(i,j) \in R} \beta_{ij} \|\mathbf{w}_i - \mathbf{w}_j\|_2\big), \tag{2.42}$$

where α and β are hyperparameters indicating the strength of the associations, and R is a set of relations in the knowledge base. The adapted embeddings $\hat{\mathbf{E}}$ can be optimized by several iterations of the following online updates:

$$\hat{\mathbf{w}}_i = \frac{\sum_{\{j|(i,j) \in R\}} \beta_{ij} \hat{\mathbf{w}}_j + \alpha_i \mathbf{w}_i}{\sum_{\{j|(i,j) \in R\}} \beta_{ij} + \alpha_i}, \tag{2.43}$$

where α is usually set to 1 and β_{ij} is $\deg(i)^{-1}$ ($\deg(\cdot)$ is a node's degree in a knowledge graph). With knowledge bases such as the paraphrase database [27], WordNet [46] and FrameNet [3], this model can achieve consistent improvement on word similarity tasks. But it also may significantly reduce the performance on the analogy of syntactic relations. Since this module is a post-processing of word embeddings, it is compatible with various distributed representation models.

In addition to the aforementioned synonym-based knowledge bases, there are also sememe-based knowledge bases, in which the sememe is defined as the minimum semantic unit of word meanings. HowNet [16] is one of such knowledge bases, which annotates each Chinese word with one or more relevant sememes. General knowledge injecting methods could not apply to HowNet. As a result, [47] proposes a specific model to introduce HowNet into word representation learning.

Bases on Skip-gram model, [47] introduces sense and sememe embeddings to represent target word w_i. More specifically, this model leverages context words, which are represented with original word embeddings, as attention over multiple senses of target word w_i to obtain its new embeddings.

$$w_i = \sum_{k=1}^{|S^{(w_i)}|} \mathrm{Att}(\mathbf{s}_k^{(w_i)})\mathbf{s}_k^{(w_i)}, \tag{2.44}$$

where $\mathbf{s}_k^{(w_i)}$ denotes the kth sense embedding of w_i and $S^{(w_i)}$ is the sense set of w_i. The attention term is as follows:

$$Att(\mathbf{s}_k^{(w_i)}) = \frac{\exp(\mathbf{w}'_c \cdot \hat{\mathbf{s}}_k^{(w_i)})}{\sum_{n=1}^{|S^{(w_i)}|} \exp(\mathbf{w}'_c \cdot \hat{\mathbf{s}}_n^{(w_i)})}, \tag{2.45}$$

where $\hat{\mathbf{s}}_k^{(w_i)}$ stands for the average of sememe embeddings $\mathbf{x}$, $\hat{\mathbf{s}}_k^{(w_i)} = \mathrm{Avg}(\mathbf{x}^{(s_k)})$ and $\mathbf{w}'_c$ is the average of context word embeddings, $\mathbf{w}'_c = \mathrm{Avg}(\mathbf{w}_j)(|j - i| \leq l, j \neq i)$.

This model shows a substantial advance in both word similarity and analogy tasks. Moreover, the introduction of sense embeddings can also be used in word sense disambiguation.

Considering Document Information. Word embedding methods like Skip-gram simply consider the context information within a window to learn word representation. However, the information in the whole document could help our word representation learning. Topical Word Embeddings (TWE) [42] introduces topic information generated by Latent Dirichlet Allocation (LDA) to help distinguish different meanings of a word. The model is defined to maximize the following average log probability:

$$\mathcal{O} = \frac{1}{N}\sum_{i=1}^{N} \sum_{-k\leq c\leq k, c\neq 0} \left(\log P(\mathbf{w}_{i+c}|\mathbf{w}_i) + \log P(\mathbf{w}_{i+c}|\mathbf{z}_i)\right), \tag{2.46}$$

where $\mathbf{w}_i$ is the word embedding and $\mathbf{z}_i$ is the topic embedding of w_i. Each word w_i is assigned a unique topic, and each topic has a topic embedding. The topical word embedding model shows advantages of contextual word similarity and document classification tasks.

However, TWE simply combines the LDA with word embeddings and lacks statistical foundations. The LDA topic model needs numerous documents to learn semantically coherent topics. Reference [40] further proposes the TopicVec model, which encodes words and topics in the same semantic space. TopicVec outperforms TWE and other word embedding methods on text classification datasets. It can learn coherent topics on only one document which is not possible for other topic models.

2.5.3.3 Word Representation with Hierarchical Structure

Human knowledge is in a hierarchical structure. Recently, many works also introduce a hierarchical structure of texts into word representation learning.

Dependency-based Word Representation. Continuous word embeddings are combinations of semantic and syntactic information. However, existing word representation models depend solely on linear contexts and show more semantic information than syntactic information. To make the embeddings show more syntactic information, the dependency-based word embedding [38] uses the dependency-based context. The dependency-based embeddings are less topical and exhibit more functional similarity than the original Skip-gram embeddings. It takes the information of dependency parsing tree into consideration when learning word representations. The contexts of a target word w are the modifiers of this word, i.e., $(m_1, r_1), \ldots, (m_k, r_k)$, where r_i is the type of the dependency relation between the head node and the modifier. When training, the model optimizes the probability of dependency-based contexts rather than neighboring contexts. This model gains some improvements on word similarity benchmarks compared with Skip-gram. Experiments also show that words with syntactic similarity are more similar in the vector space.

Semantic Hierarchies. Because of the linear substructure of the vector space, it is proven that word embeddings can make simple analogies. For example, the difference between `Japan` and `Tokyo` is similar to the difference between `China` and `Beijing`. But it has trouble identifying hypernym-hyponym relations since these relationships are complicated and do not necessarily have linear substructure.

To address this issue, [25] tries to identify hypernym-hyponym relationships using word embeddings. The basic idea is to learn a linear projection rather than simply use the embedding offset to represent the relationship. The model optimizes the projection as

$$\mathbf{M}^* = \arg\min_{\mathbf{M}} \frac{1}{N} \sum_{(i,j)} \|\mathbf{M}\mathbf{x}_i - \mathbf{y}_j\|_2, \tag{2.47}$$

where $\mathbf{x}_i$ and $\mathbf{y}_j$ are hypernym and hyponym embeddings.

To further increase the capability of the model, they propose to first cluster word pairs into several groups and learn a linear projection for each group. The linear projection can help identify various hypernym-hyponym relations.

2.5.4 Multilingual Word Representation

There are thousands of languages in the world. In word level, how to represent words from different languages in a unified vector space is an interesting problem. The bilingual word embedding model [64] uses machine translation word alignments as constraining translational evidence and embeds words of two languages into a single vector space. The basic idea is (1) to initialize each word according to its aligned words in another language and (2) to constrain the distance between two languages during the training using translation pairs.

When learning bilingual word embeddings, it firstly trains source word embeddings. Then they use aligned sentence pairs to count the co-occurrence of source and target words. The target word embeddings can be initialized as

$$\mathbf{E}_{t-init} = \sum_{s=1}^{S} \frac{N_{ts}+1}{N_t+S}\mathbf{E}_s, \tag{2.48}$$

where $\mathbf{E}_s$ and $\mathbf{E}_{t-init}$ are the trained embeddings of the source word and the initial embedding of the target word, respectively. N_{ts} is the number of target words being aligned with source word. S is all the possible alignments of word t. So $N_t + S$ normalizes the weights as a distribution. During the training, they jointly optimize the word embedding objective as well as the bilingual constraint. The constraint is defined as

$$\mathscr{L}_{cn\to en} = \|\mathbf{E}_{en} - N_{en\to cn}\mathbf{E}_{cn}\|^2, \tag{2.49}$$

where $N_{en\to cn}$ is the normalized align counts.

When given a lexicon of bilingual word pairs, [44] proposes a simple model that can learn bilingual word embeddings in a unified space. Based on the distributional geometric similarities of word vectors of two languages, this model learns a linear transformation matrix $\mathbf{T}$ that transforms the vector space of source language to that of the target language. The training loss is

$$\mathscr{L} = \|\mathbf{T}\mathbf{E}_s - \mathbf{E}_t\|^2, \tag{2.50}$$

where $\mathbf{E}_t$ is the word vector matrix of aligned words in target language.

However, this model performs badly when the seed lexicon is small. To tackle this limitation, some works introduce the idea of bootstrapping into bilingual word representation learning. Let's take [63] for example. In this work, in addition to monolingual word embedding learning and bilingual word embedding alignment based on seed lexicon, a new matching mechanism is introduced. The main idea of matching is to find the most probably matched source (target) word for each target (source) word and make their embeddings closer. Next, we explain the target-to-source matching process formally, and the source-to-target side is similar.

The target-to-source matching loss function is defined as

$$\mathscr{L}_{T2S} = -\log P\left(C^{(T)}|\mathbf{E}^{(S)}\right) = -\log \sum_{\mathbf{m}} P\left(C^{(T)}, \mathbf{m}|\mathbf{E}^{(S)}\right), \tag{2.51}$$

where $C^{(T)}$ denotes the target corpus and $\mathbf{m}$ is a latent variable specifying the matched source word for each target word. On independency assumption, it has

$$P\left(C^{(T)}, \mathbf{m}|\mathbf{E}^{(S)}\right) = \prod_{w_i^{(T)} \in C^{(T)}} P\left(w_i^{(T)}, \mathbf{m}|\mathbf{E}^{(S)}\right) = \prod_{i=1}^{|V^{(T)}|} P\left(w_i^{(T)}|w_{\mathbf{m}_i}^{(S)}\right)^{N_{w_i^{(T)}}}, \quad (2.52)$$

where $N_{w_i^{(T)}}$ is the number of $w_i^{(T)}$ occurrences in the target corpus. By training using Viterbi EM algorithm, this method can improve bilingual word embeddings on its own and address the limitation of a small seed lexicon.

2.5.5 Task-Specific Word Representation

In recent years, word representation learning has achieved great success and played a crucial role in NLP tasks. People find that word representation learning of the general field is still a limitation in a specific task and begin to explore the learning of task-specific word representation. In this section, we will take sentiment analysis as an example.

Word Representation for Sentiment Analysis. Most word representation methods capture syntactic and semantic information while ignoring sentiment of text. This is problematic because words with similar syntactic polarity but opposite sentiment polarity obtain closed word vectors. Reference [58] proposes to learn Sentiment-Specific Word Embeddings (SSWE) by integrating the sentiment information. An intuitive idea is to jointly optimize the sentiment classification model using word embeddings as its feature and SSWE minimizes the cross-entropy loss to achieve this goal. To better combine the unsupervised word embedding method and the supervised discriminative model, they further use the words in a window rather than a whole sentence to classify sentiment polarity. They propose the following ranking-based loss:

$$\mathscr{L}_r(t) = \max(0, 1 - \mathbf{1}_s(t) f_0^r(t) + \mathbf{1}_s(t) f_1^r(t)), \quad (2.53)$$

where f_0^r, f_1^r are the predicted positive and negative scores. $\mathbf{1}_s(t)$ is an indicator function:

$$\mathbf{1}_s(t) = \begin{cases} 1 & \text{if t is positive,} \\ -1 & \text{if t is negative.} \end{cases} \quad (2.54)$$

This loss function only punishes the model when the model gives an incorrect result.

To get massive training data, they use distant-supervision technology to generate sentiment labels for a document. The increase of labeled data can improve the sentiment information in word embeddings. On sentiment classification tasks, sentiment embeddings outperform other strong baselines including SVM and other word embedding methods. SSWE also shows strong polarity consistency, where the closest words of a word are more likely to have the same sentiment polarity compared with existing word representation models. This sentiment specific word embedding method provides us a general way to learn task-specific word embeddings, which is to design a joint loss function and to generate massive labeled data automatically.

2.5.6 Time-Specific Word Representation

The meaning of a word changes during the time. Analyzing the changing meaning of a word is an exciting topic in both linguistic and NLP research. With the rise of word embedding methods, some works [29, 35] use embeddings to analyze the change of words' meanings. They separate corpus into bins with respect to years to train time-specific word embeddings and compare embeddings of different time series to analyze the change of word semantics. This method is intuitive but has some problems. Dividing corpus into bins causes the data sparsity issue. The objective of word embedding methods is nonconvex so that different random initialization leads to different results, which makes comparing word embeddings difficult. Embeddings of a word in different years are in different semantic spaces and cannot be compared directly. Most work indirectly compares the meanings of a word in a different time by the changes of a word's closest words in the semantic space.

To address these issues, [4] proposes a dynamic Skip-gram model which connects several Bayesian Skip-gram models [6] using Kalman filters [33]. In this model, the embeddings of words in different periods could affect each other. For example, a word that appears in the 1990s' document can affect the embeddings of that word in the 1980s and 2000s. Moreover, it also trains the embedding in different periods by the whole corpus to reduce the sparsity issue. This model also puts all the embeddings into the same semantic space, which is a significant improvement against other methods and makes word embeddings in different periods comparable. Therefore, the change of word embeddings in this model is continuous and smooth. Experimental results show that the cosine distance between two words changes much more smoothly in this model than those models which simply divide the corpus into bins.

2.6 Evaluation

In recent years, various methods to embed words into a vector space have been proposed. Hence, it is essential to evaluate different methods. There are two general evaluations of word embeddings, including word similarity and word analogy. They both aim to check if the word distribution is reasonable. These two evaluations sometimes give different results. For example, CBOW achieves better performance on word similarity, whereas Skip-gram outperforms CBOW on word analogy. Therefore, which method to choose depends on the high-level application. Task-specific word embedding methods are usually designed for specific high-level tasks and achieve significant improvement on these tasks compared with baselines such as CBOW and Skip-gram. However, they only marginally outperform baselines on two general evaluations.

2.6.1 Word Similarity/Relatedness

The dynamics of words are very complex and subtle. There is no static, finite set of relations that can describe all interactions between two words. It is also not trivial for downstream tasks to leverage different kinds of word relations. A more practical way is to assign a score to a pair of words to represent to what extent they are related. This measurement is called *word similarity*. When talking about the term *word similarity*, the precise meaning may vary a lot in different situations. There are several kinds of *similarity* that may be referred to in various literature.

Morphological similarity. Many languages including English define morphology. The same morpheme can have multiple surface forms according to the syntactical function. For example, the word `active` is an adjective and `activeness` is its noun version. The word `activate` is a verb and `activation` is its noun version. The morphology is an important dimension when considering the meaning and usage of words. It defines some relations between words from a syntactical view. Some relations are used in the Syntactic Word Relationship test set [43], including adjectives to adverbs, past tense, and so on. However, in many higher level applications and tasks, the words are often morphologically normalized by the base form (this process is also known as lemmatization). One widely used technique is the Porter stemming algorithm [49]. This algorithm converts `active`, `activeness`, `activate`, and `activation` to the same root format `activ`. By removing morphological features, the semantic meaning of words is more emphasized.

Semantic Similarity. Two words are semantically similar if they can express the same concept, or sense, like `article` and `document`. One word may have different senses, and each of its synonyms is associated with one or more of its senses. WordNet [46] is a lexical database that organizes the words as groups according to the senses. Each group of words is called a *synset*, which contains all synonymous words sharing the same specific sense. The words within the same synset are considered semantically similar. Words from two synsets that are linked by some certain relation (such as *hyponym*) are also considered semantically similar to some degree, like `bank(river)` and `bank` (`bank(river)` is the hyponym of `bank`).

Semantic relatedness. Most modern literature that considers word similarity refers to the semantic relatedness of words. Semantic relatedness is more general than semantic similarity. Words that are not semantically similar could still be related in many ways such as meronymy (`car` and `wheel`) or antonymy (`hot` and `cold`). Semantic relatedness often yields co-occurrence, but they are not equivalent. The syntactic structure could also yield co-occurrence. Reference [10] argues that distributional similarity is not an adequate proxy for semantic relatedness.

Table 2.3 Datasets for evaluating word similarity/relatedness

Dataset	Similarity Type
RG-65 [52]	Word Similarity
WordSim-353 [22]	Mixed
WordSim-353 REL [1]	Word Relatedness
WordSim-353 SIM [1]	Word Similarity
MTurk-287 [50]	Word Relatedness
SimLex-999 [31]	Word Similarity

To evaluate the word representation system intrinsically, the most popular approach is to collect a set of word pairs and compute the correlation between human judgment and system output. So far, many datasets are collected and made public. Some datasets focus on the word similarity, such as RG-65 [52] and SimLex-999 [31]. Other datasets concern word relatedness, such as MTurk [50]. WordSim-353 [22] is a very popular dataset for word representation evaluation, but its annotation guideline does not differentiate similarity and relatedness very clearly. Reference [1] conducts another round of annotation based on WordSim-353 and generates two subsets, one for similarity and the other for relatedness. Some information about these datasets is summarized in Table 2.3.

To evaluate the similarity of two distributed word vectors, researchers usually select cosine similarity as an evaluation metric. The cosine similarity of word w and word v is defined as

$$\text{sim}(w, v) = \frac{\mathbf{w} \cdot \mathbf{v}}{\|\mathbf{w}\| \|\mathbf{v}\|}. \tag{2.55}$$

When evaluating a word representation approach, the similarity of each word pair is computed in advance using cosine similarity. After that, Spearman's correlation coefficient ρ is then used to evaluate the similarity between human annotator and word representation model as

$$\rho = 1 - \frac{6 \sum d_i^2}{n^3 - n}, \tag{2.56}$$

where a higher Spearman's correlation coefficient indicates they are more similar.

Reference [10] describes a series of methods based on WordNet to evaluate the similarity of a pair of words. After the comparison between the traditional WordNet-based methods and distributed word representations, [1] addresses that relatedness and similarity are two different concerns. They point out that WordNet-based methods perform better on similarity than on relatedness, while distributed word representation shows similar performance on both. A series of distributed word representations are compared on a wide variety of datasets in [56]. The state-of-the-art on both similarity and relatedness is achieved by distributed representation, without a doubt.

This evaluation method is simple and straightforward. However, as stated in [20], there are several problems with this evaluation. Since the datasets are small (less than 1,000 word pairs in each dataset), one system may yield many different scores on different partitions. Testing on the whole dataset makes it easier to overfit and hard to compute the statistical significance. Moreover, the performance of a system on these datasets may not be very correlated to its performance on downstream tasks.

The word similarity measurement can come in an alternative format, the TOEFL synonyms test. In this test, a cue word is given, and the test is required to choose one from four words that are the synonym of the cue word. The exciting part of this task is that the performance of a system could be compared with human beings. Reference [37] evaluates the system with the TOEFL synonyms test to address the knowledge inquiring and representing of LSA. The reported score is 64.4%, which is very close to the average rating of the human test-takers. On this test set with 80 queries, [54] reported a score of 72.0%. Reference [24] extends the original dataset with the help of WordNet and generates a new dataset[3] (named WordNet-based synonymy test) containing thousands of queries.

2.6.2 *Word Analogy*

Besides word similarity, the word analogy task is an alternative way to measure how well representations capture semantic meanings of words. This task gives three words w_1, w_2, and w_3, then it requires the system to predict a word w_4 such that the relation between w_1 and w_2 is the same as that between w_3 and w_4. This task is used since [43, 45] to exploit the structural relationships among words. Here, the word relations could be divided into two categories, including semantic relations and syntactic relations. This is a relatively novel method for word representation evaluation but quickly becomes a standard evaluation metric since the dataset is released. Unlike the TOEFL synonyms test, most words in this dataset are frequent across all kinds of the corpus, but the fourth word is chosen from the whole vocabulary instead of four options. This test favors distributed word representations because it emphasizes the structure of word space.

The comparison between different models on the word analogy task measured by accuracy could be found in [7, 56, 57, 61].

2.7 Summary

In this chapter, we first introduce word representation methods, including one-hot representation and various distributed representation methods. These classical methods are the important foundation of various NLP models, and meanwhile present the

[3]http://www.cs.cmu.edu/~dayne/wbst-nanews.tar.gz.

major concepts and mechanisms of word representation learning for the reader. Next, considering classical word representation methods often suffer from the word polysemy, we further introduce the effective contextualized word representation methods ELMo, to show the approach to capture complex word features across different linguistic contexts. As word representation methods are widely utilized in various downstream tasks, we then overview numerous extensions toward some representative directions and discuss how to adapt word representations for specific scenarios. Finally, we introduce several evaluation tasks of word representation, including word similarity and word analogy, which are the basic experimental settings for researching word representation methods.

In the past decade, learning methods and applications of word representation have been studied in depth. Here we recommend some surveys and books on word representative learning for reading:

- Erk. Vector Space Models of Word Meaning and Phrase Meaning: A Survey [18].
- Lai et al. How to Generate a Good Word Embedding [36].
- Camacho et al. From Word to Sense Embeddings: A Survey on Vector Representations of Meaning [11].
- Ruder et al. A Survey of Cross-lingual Word Embedding Models [53].
- Bakarov. A Survey of Word Embeddings Evaluation Methods [2].

In the future, toward more effective word representation learning, some directions are requiring further efforts:

(1) **Utilizing More Knowledge**. Current word representation learning models focus on representing words based on plain textual corpora. In fact, besides rich semantic information in text, there are also various kinds of word-related information hidden in heterogeneous knowledge in the real world, such as visual knowledge, factual knowledge, and commonsense knowledge. Some preliminary explorations have attempted [59, 60] to utilize heterogeneous knowledge for learning better word representations, and these explorations indicate that utilizing more knowledge is a promising direction toward enhancing word representations. There remain open problems for further explorations.

(2) **Considering More Contexts**. As shown in this chapter, those word representation learning methods considering contexts can achieve more expressive word embeddings, which can grasp richer semantic information and further benefit downstream NLP tasks than classical distributed methods. Context-aware word representations have been systematically verified for their effectiveness in existing works [32, 48], and adopting those context-aware word representations has also become a necessary and mainstream operation for various NLP tasks. After BERT [15] has been proposed, language models pretrained on large-scale corpora have entered the public vision and their fine-tuning models have also achieved the state-of-the-art performance on specific NLP tasks. These new explorations based on large-scale textual corpora and pretrained fine-tuning language representation architectures indicate a promising direction to consider more contexts with more powerful representation architectures, and we will discuss them more in the next chapter.

(3) **Orienting Finer Granularity**. Polysemy is a widespread phenomenon for words. Hence, it is essential and meaningful to consider the finer granulated semantic information than the words themselves. As some linguistic knowledge bases have been developed, such as synonym-based knowledge bases WordNet [21] and sememe-based knowledge bases HowNet [17], we thus have ways to study the atomic semantics of words. The current work on word representations learning is coarse-grained, and mainly focuses on shallow semantics of the words themselves in text, and ignores the rich semantic information inside the words, which is also an important resource for achieving better word embeddings. Reference [28] explores to inject finer granulated atomic semantics of words into word representations and performs much better language understanding. Although these explorations are still preliminary, orienting finer granularity of word representations is important. In the next chapter, we will also introduce more details in this part.

In the past decade, learning methods and applications of distributed representation have been studied in depth. Because of its efficiency and effectiveness, lots of task-specific models have been proposed for various tasks. Word representation learning has become a popular and important topic in NLP. However, word representation learning is still challenging due to its ambiguity, data sparsity, and interpretability. In recent years, word representation learning has been no longer studied in isolation, but explored together with sentence or document representation learning using pretrained language models. Readers are recommended to refer to the following chapters to further learn the integration of word representations in other scenarios.

References

1. Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of HLT-NAACL*, 2009.
2. Amir Bakarov. A survey of word embeddings evaluation methods. *arXiv preprint* arXiv:1801.09536, 2018.
3. Collin F Baker, Charles J Fillmore, and John B Lowe. The berkeley framenet project. In *Proceedings of ACL*, 1998.
4. Robert Bamler and Stephan Mandt. Dynamic word embeddings via skip-gram filtering. *arXiv preprint* arXiv:1702.08359, 2017.
5. Arindam Banerjee, Inderjit S Dhillon, Joydeep Ghosh, and Suvrit Sra. Clustering on the unit hypersphere using von mises-fisher distributions. *Journal of Machine Learning Research*, 2005.
6. Oren Barkan. Bayesian neural word embedding. In *Proceedings of AAAI*, 2017.
7. Marco Baroni, Georgiana Dinu, and Germán Kruszewski. Dont count, predict a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of ACL*, 2014.
8. Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.
9. Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479, 1992.

10. Alexander Budanitsky and Graeme Hirst. Evaluating wordnet-based measures of lexical semantic relatedness. *Computational Linguistics*, 32(1):13–47, 2006.
11. Jose Camacho-Collados and Mohammad Taher Pilehvar. From word to sense embeddings: A survey on vector representations of meaning. *Journal of Artificial Intelligence Research*, 63:743–788, 2018.
12. Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. A unified model for word sense representation and disambiguation. In *Proceedings of EMNLP*, 2014.
13. Xinxiong Chen, Lei Xu, Zhiyuan Liu, Maosong Sun, and Huanbo Luan. Joint learning of character and word embeddings. In *Proceedings of IJCAI*, 2015.
14. Scott C. Deerwester, Susan T Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *Japan Analytical & Scientific Instruments Show*, 41(6):391–407, 1990.
15. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL*, 2019.
16. Zhendong Dong and Qiang Dong. Hownet-a hybrid language and knowledge resource. In *Proceedings of NLP-KE*, 2003.
17. Zhendong Dong and Qiang Dong. *HowNet and the Computation of Meaning (With CD-Rom)*. World Scientific, 2006.
18. Katrin Erk. Vector space models of word meaning and phrase meaning: A survey. *Language and Linguistics Compass*, 6(10):635–653, 2012.
19. Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. Retrofitting word vectors to semantic lexicons. In *Proceedings of NAACL-HLT*, 2015.
20. Manaal Faruqui, Yulia Tsvetkov, Pushpendre Rastogi, and Chris Dyer. Problems with evaluation of word embeddings using word similarity tasks. In *Proceedings of RepEval*, 2016.
21. Christiane Fellbaum. Wordnet. *The encyclopedia of applied linguistics*, 2012.
22. Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. Placing search in context: The concept revisited. In *Proceedings of WWW*, 2001.
23. John R Firth. A synopsis of linguistic theory, 1930–1955. 1957.
24. Dayne Freitag, Matthias Blume, John Byrnes, Edmond Chow, Sadik Kapadia, Richard Rohwer, and Zhiqiang Wang. New experiments in distributional representations of synonymy. In *Proceedings of CoNLL*, 2005.
25. Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, and Ting Liu. Learning semantic hierarchies via word embeddings. In *Proceedings of ACL*, 2014.
26. Alona Fyshe, Leila Wehbe, Partha Pratim Talukdar, Brian Murphy, and Tom M Mitchell. A compositional and interpretable semantic space. In *Proceedings of HLT-NAACL*, 2015.
27. Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. Ppdb: The paraphrase database. In *Proceedings of HLT-NAACL*, 2013.
28. Yihong Gu, Jun Yan, Hao Zhu, Zhiyuan Liu, Ruobing Xie, Maosong Sun, Fen Lin, and Leyu Lin. Language modeling with sparse product of sememe experts. In *Proceedings of EMNLP*, pages 4642–4651, 2018.
29. William L Hamilton, Jure Leskovec, and Dan Jurafsky. Diachronic word embeddings reveal statistical laws of semantic change. In *Proceedings of ACL*, 2016.
30. Zellig S Harris. Distributional structure. *Word*, 10(2–3):146–162, 1954.
31. Felix Hill, Roi Reichart, and Anna Korhonen. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 2015.
32. Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Proceedings of ACL*, pages 328–339, 2018.
33. Rudolph Emil Kalman et al. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.
34. Pentti Kanerva, Jan Kristofersson, and Anders Holst. Random indexing of text samples for latent semantic analysis. In *Proceedings of CogSci*, 2000.
35. Yoon Kim, Yi-I Chiu, Kentaro Hanaki, Darshan Hegde, and Slav Petrov. Temporal analysis of language through neural language models. In *Proceedings of the ACL Workshop*, 2014.

36. Siwei Lai, Kang Liu, Shizhu He, and Jun Zhao. How to generate a good word embedding. *IEEE Intelligent Systems*, 31(6):5–14, 2016.
37. Thomas K Landauer and Susan T Dumais. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2):211, 1997.
38. Omer Levy and Yoav Goldberg. Dependency-based word embeddings. In *Proceedings of ACL*, 2014.
39. Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Proceedings of NeurIPS*, 2014.
40. Shaohua Li, Tat-Seng Chua, Jun Zhu, and Chunyan Miao. Generative topic embedding: a continuous representation of documents. In *Proceedings of ACL*, 2016.
41. Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramón Fermandez, Silvio Amir, Luis Marujo, and Tiago Luís. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of EMNLP*, 2015.
42. Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. Topical word embeddings. In *Proceedings of AAAI*, 2015.
43. Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Proceedings of ICLR*, 2013.
44. Tomas Mikolov, Quoc V Le, and Ilya Sutskever. Exploiting similarities among languages for machine translation. *arXiv preprint* arXiv:1309.4168, 2013.
45. Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of HLT-NAACL*, 2013.
46. George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.
47. Yilin Niu, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. Improved word representation learning with sememes. In *Proceedings of ACL*, 2017.
48. Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237, 2018.
49. Martin F Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.
50. Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. A word at a time: computing word relatedness using temporal semantic analysis. In *Proceedings of WWW*, 2011.
51. Joseph Reisinger and Raymond J Mooney. Multi-prototype vector-space models of word meaning. In *Proceedings of HLT-NAACL*, 2010.
52. Herbert Rubenstein and John B Goodenough. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633, 1965.
53. Sebastian Ruder, Ivan Vulić, and Anders Søgaard. A survey of cross-lingual word embedding models. *Journal of Artificial Intelligence Research*, 65:569–631, 2019.
54. Magnus Sahlgren. Vector-based semantic analysis: Representing word meanings based on random labels. In *Proceedings of Workshop on SKAC*, 2001.
55. Magnus Sahlgren. An introduction to random indexing. In *Proceedings of TKE*, 2005.
56. Tobias Schnabel, Igor Labutov, David Mimno, and Thorsten Joachims. Evaluation methods for unsupervised word embeddings. In *Proceedings of EMNLP*, 2015.
57. Fei Sun, Jiafeng Guo, Yanyan Lan, Jun Xu, and Xueqi Cheng. Learning word representations by jointly modeling syntagmatic and paradigmatic relations. In *Proceedings of ACL*, 2015.
58. Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of ACL*, 2014.
59. Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the EMNLP*, pages 1499–1509, 2015.
60. Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph and text jointly embedding. In *Proceedings of EMNLP*, pages 1591–1601, 2014.

61. Dani Yogatama, Manaal Faruqui, Chris Dyer, and Noah A Smith. Learning word representations with hierarchical sparse coding. In *Proceedings of ICML*, 2015.
62. Mo Yu and Mark Dredze. Improving lexical embeddings with semantic knowledge. In *Proceedings of ACL*, 2014.
63. Meng Zhang, Haoruo Peng, Yang Liu, Huan-Bo Luan, and Maosong Sun. Bilingual lexicon induction from non-parallel data with minimal supervision. In *Proceedings of AAAI*, 2017.
64. Will Y Zou, Richard Socher, Daniel M Cer, and Christopher D Manning. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of EMNLP*, 2013.