Computing (2020) 102:717–740
https://doi.org/10.1007/s00607-019-00768-7

A survey of word embeddings based on deep learning

Shirui Wang[1] · Wenan Zhou[1] · Chao Jiang[1]

Received: 4 June 2019 / Accepted: 5 November 2019 / Published online: 12 November 2019

Abstract

The representational basis for downstream natural language processing tasks is word embeddings, which capture lexical semantics in numerical form to handle the abstract semantic concept of words. Recently, the word embeddings approaches, represented by deep learning, has attracted extensive attention and widely used in many tasks, such as text classification, knowledge mining, question-answering, smart Internet of Things systems and so on. These neural networks-based models are based on the distributed hypothesis while the semantic association between words can be efficiently calculated in low-dimensional space. However, the expressed semantics of most models are constrained by the context distribution of each word in the corpus while the logic and common knowledge are not better utilized. Therefore, how to use the massive multi-source data to better represent natural language and world knowledge still need to be explored. In this paper, we introduce the recent advances of neural networks-based word embeddings with their technical features, summarizing the key challenges and existing solutions, and further give a future outlook on the research and application.

Keywords Word embeddings · Neural networks · Distributed hypothesis · Multi-source data

Mathematics Subject Classification 68T50

✉ Wenan Zhou
zhouwa@bupt.edu.cn

Shirui Wang
wangshirui@bupt.edu.cn

Chao Jiang
jc0527@bupt.edu.cn

1 Department of Computer Science, Beijing University of Posts and Telecommunications, Beijing 100876, China

1 Introduction

Word representation has been one of the key areas of the focus for researchers since the direction of NLP. Many tasks conventionally take one-hot word representation method early, where each word is represented as a vocabulary-size vector with only one non-zero entry. Due to its simplicity, one-hot word representation has been widely adopted as the basis of NLP and IR [1]. But there is a critical flaw of one-hot representation that it does not take into account any semantic relatedness between words and also face the data-sparse problems [2].

The distribution hypothesis [3, 4], that contextually similar words have similar semantics, provides a theoretical basis for the idea of integrating semantics into word representations. In the early 1990s, statistical methods gradually became mainstream in natural language processing. The method of expressing semantics based on the distribution hypothesis was once again paid attention to and applied to tasks such as word sense disambiguation [5–8]. These methods were called the word-space model at that time. In the subsequent development, word representation methods based on the distribution hypothesis are mainly divided into three types: matrix-based distributed representation, also known as distributional representation [9–15], cluster-based distributed representation [16–18], and neural network-based distributed representation, also known as word embeddings [19–23], word embeddings models the context and the relationship between target word and its context words through neural networks and contain fruitful semantic information in the embeddings, which can be obtained by training language models [19, 20, 22–27], or by building specifically neural networks for generating word embeddings [28–36]. With the improvement of hardware performance and the breakthrough of optimization algorithms, the neural network models gradually exert their own advantages in various fields, and neural network-based word embeddings, which can better model the word and its context, have become mainstream methods of word distribution representations [34, 37–42], and play a powerful role in many downstream tasks such as natural language inference [43, 44],text classification [45, 46], knowledge mining [47], and named entity recognition [48–51]. As shown in Fig. 1, we present a timeline of the recent development of key network structures and models since 2013 [9, 29, 32, 52–54]. As is seen from above, the field has been moving strikingly fast since 2013 when the model Word2vec was proposed. Compare with the former methods, it removes the nonlinear hidden layer in the forward feedback neural network, connects the intermediate layer to the softmax layer directly. Besides, it also proposes two acceleration strategies include hierarchical softmax and negative sampling which effectively improves efficiency. The innovations in building better network structures and more effective models have occurred alternately and both contributed to the development of the field. It is therefore very important to design methods that can effectively capture word semantics. However, these representation methods still face some challenges, we summarize these challenges into three parts, including the following aspects:

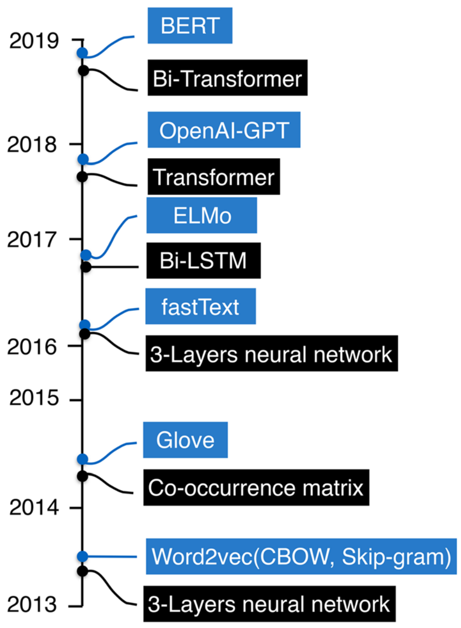


Fig. 1 A timeline of the recent development of key network structures and models of word embeddings since 2013

- Out-of-vocabulary (OOV) words. The datasets used for training cannot completely include all words and there will always be a steady stream of new words, which causes the problem of how to deal with the OOV words to fuse the task when the word embeddings is applied.
- Textual representation depends on its context. Understanding the context of word is necessary for most downstream tasks in natural language processing, which is based on the fact that a word usually have different meanings in different contexts, and correspondingly, there should be different word embeddings.
- Word embeddings of different languages. The writing system and language structure of different languages are completely different, it is therefore necessary to design the specific word embeddings model for the specific language.

In this paper, the classic word embedding models are introduced in Sect. 2, the existing solutions of OOV challenges are presented in Sect. 3, models to solve the contextual word representations problem are presented in Sect. 4, the word embedding methods for different language are introduced in Sect. 5 and we briefly introduce some neural network structures applied to natural language processing in Sect. 6. Finally, we also give a conclusion and prospect of the development of word embedding technology. We believe that to truly understand natural language, we must not only model the language itself with external knowledge and data but also model the environment and combine language learning with comprehending of the world.

2 Classic models

2.1 Language model

The language model is the method of generating text, which is the maximum likelihood of a sequence of multiple n words $\left(w_1, w_2, \dots, w_n\right)$. For the target probability $p\left(w_1, w_2, \dots, w_n\right)$ of the language model, if it is assumed that each word in the text is independent of each other, the joint probability of the whole sentence can be expressed as the product of the conditional probabilities of all the words according to:

$$p\left(w_1, w_2, \dots, w_n\right) = \prod_{i=1}^{n} p\left(w_i\right). \tag{1}$$

If we consider the finite historical hypothesis: the occurrence of the nth word is only related to the previous $n-1$ words, but not related to any other words and the joint probability is:

$$p\left(w_1, w_2, \dots, w_n\right) = \prod_{i=1}^{n} p(w_i|w_{i-n+1}, \dots, w_{i-1}). \tag{2}$$

2.2 Neural network language model

Bengio et al. [20] proposed the model named NNLM, which obtains word embeddings as the product while training language model. Similar to the traditional language model, NNLM uses the previous $n-1$ words to predict the nth word as it overall structure. The difference, however, is it based on a neural network to modeling rather than predicting the occurrence probability of nth word with counting. As shown in Fig. 2, we present the basic structure of NNLM.

NNLM uses a three-layer network structure while the input of the model is the sequential splicing of word embeddings of the sequence $w_{i-n+1}, w_{i-n+2}, \dots, w_{i-1}$ as shown in formula (3):

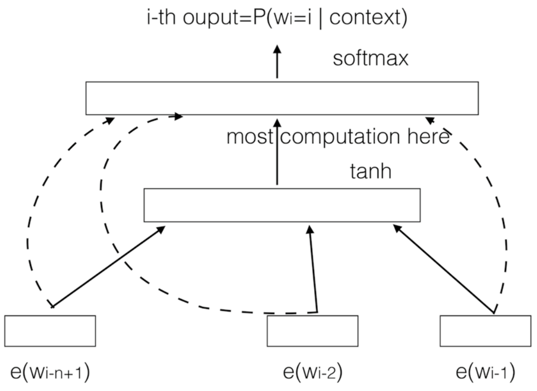


Fig. 2 The structure of neural network language model

$$x = \left[e\left(w_{i-(n-1)}\right); \cdots ; e\left(w_{i-2}\right); e\left(w_{i-1}\right)\right] \tag{3}$$

where the $e\left(w_{i-1}\right)$ is the embedding of word w_{i-1}. Through the contacts of embeddings of input words, we can obtained x as the final input, which used to predict the output through the hidden layer and output layer as shown in formulas (4) and (5):

$$h = tanh\left(b^{(1)} + Hx\right) \tag{4}$$

$$y = b^{(2)} + Wx + Uh \tag{5}$$

where both $b^{(1)}$ and $b^{(2)}$ are bias terms in the model, y is the output, H is the weight matrix of the hidden layer, U is the weight matrix of the output layer, and W represents the direct-edge weight matrix from the input layer to the output layer.

2.3 Word2Vec

Word2vec [29, 30] is an efficient and effective tool for learning word representations from corpus, which implements two models: CBOW and Skip-gram. These two models can effectively capture the semantics of words and easily transferred into other downstream tasks.

2.3.1 CBOW

As shown in Fig. 3, the model contains three parts, and the training objective of the CBOW model is to find word representations that are useful to predict the target word by the context words of it. In the Fig. 3 (left), the window size we take is 2, the words w_{t-2}, w_{t-1}, w_{t+1} and w_{t+2} are context words of target word wt. Specifically, given $w_1, w_2, \dots, w_N$, which is a sequence of corpus words, the objective of the CBOW model is to maximize the following probability:

$$\frac{1}{N} \sum_{t=1}^{N} \sum_{-c \le j \le c, j \ne 0} logp\left(w_t | w_{t+j}\right) \tag{6}$$

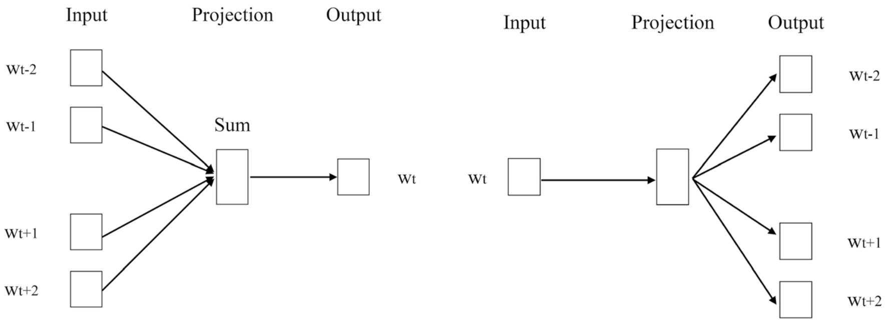


Fig. 3 The overall architecture of CBOW (left) and skip-gram (right) model

where c is the context window size, and the basic CBOW formulation defines $p(w_t|w_{t+j})$ using the softmax function as shown in formula (7):

$$p(w_t|w_{t+j}) = \frac{exp\left(sim\left(w_t, w_{t+j}\right)\right)}{\sum_{w' \in V} exp\left(sim\left(w', w_{t+j}\right)\right)} \tag{7}$$

where w' is a word in the vocabulary V, and $sim\left(w_t, w_{t+j}\right)$ is the similarity between current word w_t and one of its context words w_{t+j} as shown in formula (8):

$$sim\left(w_t, w_{t+j}\right) = \overrightarrow{w_t} \cdot \overrightarrow{w_{t+j}} \tag{8}$$

where $\overrightarrow{w_t}$ and $\overrightarrow{w_{t+j}}$ are the embeddings of w_t and w_{t+j} respectively. Since the number of V vocabulary is generally very large, the amount of calculation of the entire W matrix must be updated every time, and this is a problem of sample imbalance. The number of occurrences of different words will vary greatly, so Two different optimization methods, multilayer Softmax and negative sampling, are used in the paper.

2.3.2 Skip-gram

Similar to the principle of CBOW, the training objective of the Skip-gram model is to find word representations that are useful to predict the context words by the target word. As shown in Fig. 3, the window size we take is 2, the target word is *wt* and w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2} are context words of it while the input of model is the target word. First, the target word is mapped to a hidden layer representation vector, and we predict the context words according to this vector. Similar to the CBOW model, the objective of the Skip-gram model is to maximize the average log probability as follows:

$$\frac{1}{N}\sum_{t=1}^{N}\sum_{-c\le j\le c, j\ne 0} logp(w_{t+j}|w_t) \tag{9}$$

where c is the context window size, and the model defines $p(w_{t+j}|w_t)$ using the softmax function as shown in formula (10), where w' is a word in the vocabulary V.

$$p(w_{t+j}|w_t) = \frac{\exp\left(sim\left(w_{t+j}, w_t\right)\right)}{\sum_{w' \in V} exp\left(sim\left(w', w_t\right)\right)}. \tag{10}$$

2.4 Glove

The word embeddings trained from Word2vec can better capture the semantics of words and exploit the relatedness of words. However, Pennington et al. [9] consider that the Word2vec only focus on the information obtained from local context window while the global statistic information is not used well. So the Glove is proposed, which is a popular model based on the global co-occurrence matrix, each element

X_{ij} in the matrix represents the frequency of the word w_i and the word w_j co-occur in a particular context window.

To construct an approximate relationship between a word embedding and a co-ocurrence Matrix, Pennington et al. propose the following formula to approximate the relationship between the two words:

$$w_i^T \tilde{w}_j + b_i + \tilde{b}_j = log\left(X_{ij}\right) \tag{11}$$

where $\overrightarrow{w_i}$ and $\overrightarrow{w_j}$ are the corresponding embeddins of w_i and w_j, b_i and b_j are their offset parameters. After that we can use the following loss function to trian our word embeddings:

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - log(X_{ij})\right)^2 \tag{12}$$

where $f(x)$ is a weight function used to control the weight does not increase excessively when the word frequency is too high:

$$f(x) = \begin{cases} (x/xmax)^{0.75}, & if \quad x < xmax \\ 1 & if \quad x \geq xmax \end{cases} \tag{13}$$

As shown in Table 1, we also give a comparison of the context representation of classic word embedding models we introduced earlier in a more intuitive way.

3 Embeddings for out-of-vocabulary words

Models such as Word2vec are simple, efficient, and can learn semantic representations of words on large data sets, but they cannot learn embeddings from out-of-vocabulary(OOV) words. Such words can be interpreted in two ways: words that are not included in the existing vocabulary and words that do not appear in the existing training corpus. OOV words can be roughly divided into the 4 types including: (1) Emerging common vocabulary (such as online terms, etc.). (2) Proper names (such as names of people, names of places and organizations, time and digital expressions, etc.). (3) Professional nouns and research field names. (4) Other terminology (such

Table 1 Comparison of the context representation of classic models

Model	Representation of the context
NNLM	$\tanh\left(d + H\left[e\left(w_{i-\mathrm{n}}\right); \cdots ; e\left(w_{i-2}\right); e\left(w_{i-1}\right)\right]\right)$
CBOW	$\frac{1}{2n}\left[e\left(w_{i-n}\right) + \cdots + e\left(w_{i-1}\right) + e\left(w_{i+1}\right) + \cdots + e\left(w_{i+n}\right)\right]$
Skip-gram	$e\left(w_j\right) i - n \leq j \leq i - 1 \quad or \quad i + 1 \leq j \leq i + n$
Glove	$e\left(w_j\right) i - n \leq j \leq i - 1 \quad or \quad i + 1 \leq j \leq i + n$

Assuming the target work is w_i, the window size is n and the context words of w_i are $w_{i-\mathrm{n}}, \ldots, w_{i-1}, w_{i+1}, w_{i+\mathrm{n}}$

as the name of a new product, the name of a literary work such as a movie or a book, etc.).

In most cases, expanding the vocabulary is the preferred solution. After processing, the vocabulary is enlarged, and we can assume the remaining out-of-vocabulary words are the long tails of the unpopular parts. Therefore, these words can be directly thrown away and delete, or be replaced with a *UNK* tag. Obviously, this is not sensible, so how should we solve the problem of OOV words? Recently, a large number of extended models based on neural networks have attempted to solve this challenge. We briefly introduce some state-of-the-art models as follows.

3.1 fastText

fastText is a text categorizer which is based on CBOW and open sourced by Facebook AI Research [55]. The word embeddings is the product of fastText classification. Compared with other text classification models, fastText greatly shortens the training time while maintaining the classification effect.

Most of the existing word representation approaches assigning a distinct vector to each word while words are regarded as atomic tokens. This is a limitation, especially for languages which consist with sub-word level information. fastText uses sub-word n-gram information, which can obtain the order relationship between characters and better capture the internal semantics of words, to solve this problem. As shown in Fig. 4, the input of the model is the n-gram features $w_1, w_2, \ldots w_n$ of word w in the sentence. For example. For a word "subword", they first add two characters "<" and ">" to express the boundaries of it: <subword> and then they extract the n-grams information. If we assume to extract tri-gram information of word "subword", we can get the following set: G ≤ su, sub, ubw, bwo, wor, ord, rd>.

In practice, the author tends to extract multiple n-grams of words at the same time, such as 2/3/4/5-gram. Thus, the original word google is represented by a character-level n-gram collection. During the training process, each n-gram will train a vector, and the original word vector will be summed by the vector of all its corresponding n-grams. All word vectors and character-level n-gram vectors are simultaneously summed and averaged as input to the training model (Fig. 4).

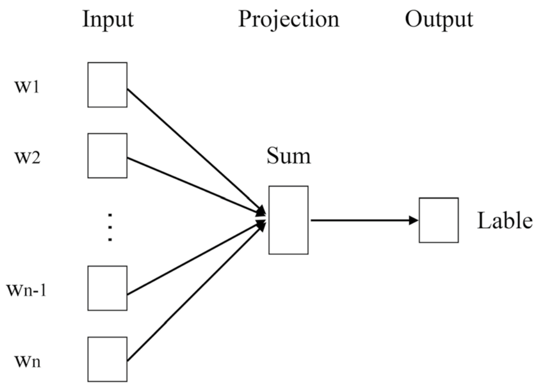


Fig. 4 The structure of fastText model

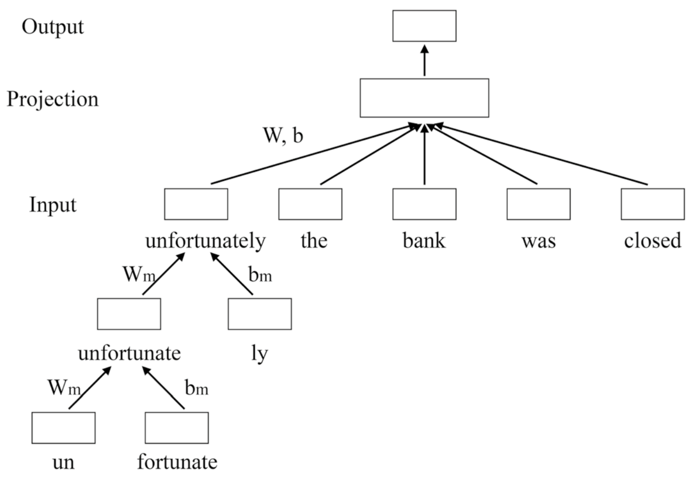


Fig. 5 The overall architecture of MorphoRNN model

3.2 MorphoRNN

The n-grams of words can better exploit the rich internal semantics, however, English is made up of various meaningful affixes such as prefix, root and suffix. Luong et al. [56] introduce morphology concept for learning word embeddings. Subword representation is obtained by training the affix with Recurrent Neural Network (RNN) and the embedding of the parent word is obtained by all morphemes. As shown in Fig. 5, different with [57], the morphoRNN models words through morpheme level rather than word level. Specifically, they consider the morpheme as the minimum unit of natural language, assigning a distinct vector to each morpheme. The embeddings of morphologically words are established from their corresponding embedding of each morpheme. A new parent word

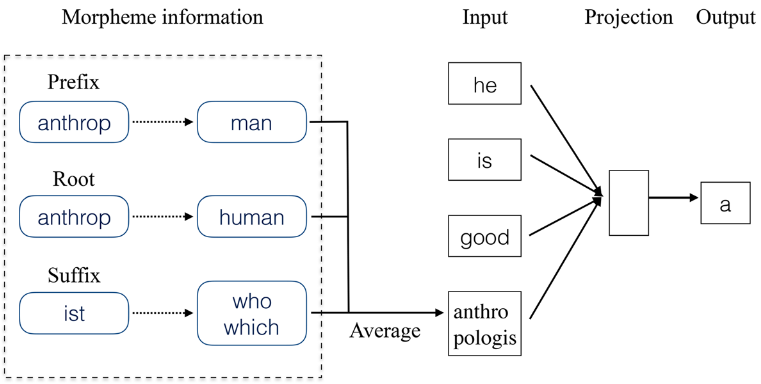


Fig. 6 The overall architecture of MWE model with an example

embedding $\vec{w}$ is obtained from a stem embedding $\vec{w}_{stem}$ and an affix embedding $\vec{w}_{affix}$ as follows:

$$\vec{w} = f\left(W_m \left[\vec{w}_{stem}; \vec{(w)}_{affix} \right] + b_m \right) \tag{14}$$

where $W_m \in R^{d \times 2d}$ is a matrix of morphemic parameters, $b_m \in R^{d \times 1}$ is an bias item, the activation function f can be set such as *tanh.*

3.3 Mwe

Considering the form information in the word vector model is a common enhancement. The general practice is to model the prefix, suffix and root of the word as independent tokens. As shown in Fig. 6, the idea of MWE is to represent the meaning of prefix and suffix [42]. This model is built on the assumption that all meanings of morphemes of token ti have equal contributions to ti. Given a sequence of tokens $W = \{w_1, w_2, \ldots, w_n\}$, we assume that the morphemes' meanings set of $w_i (i \in [1, n])$ is M_i. M_i can be divided into three parts P_i, R_i and S_i, which indicate the prefix meaning set, root meaning set and suffix meaning set of w_i, respectively. Hence, when w_i is the context word of w_j, the modified embedding of w_i can be defined as follows:

$$\widehat{\vec{w_i}} = \frac{1}{2}\left(\overrightarrow{w_i} + \frac{1}{N_i} \sum_{t \in M_i} \vec{t} \right) \tag{15}$$

where $\overrightarrow{w_i}$ is the original word embedding of w_i, N_i denotes the length of M_i and $\vec{t}$ indicates the embedding of t. Xu and Liu [42] assume the original word embedding and the sum of its morphemes' meanings embedding have equal weight. Namely, they are both 0.5.

Besides, there are also lots of works which concentrate on the sub-word-level information. Sennrich et al. [58] introduced BPE method which combined the most frequent adjacent elements into the sub-word to enhance the word representation, and they have also shown that their method performed better on neural machine translation task that other methods. Cotterell and Schütze [59] enhanced Logarithmic Bilinear Model (LBL) with the task of predicting its morpheme label and shown that their approach produced word embeddings that better preserve morphological relationships. Bian et al. [60] combined knowledge graph with text representation to enhance the performance of word embeddings, the experiments on word analogical reasoning, word similarity, and word completion tasks have shown that their approach can enhance the effectiveness of word embeddings. Cao and Rei [61] proposed a char2vec model, using Bi-LSTM to obtain the each character embedding of the word. The forward LSTM can obtain the representation of the prefix and the root, the reverse LSTM can obtain the representation of the suffix and the root. They also proved that their morphological analysis was comparable to dedicated morphological analyzers at the task of morpheme boundary recovery, and also performed well on the syntactic analogy answering task. Kim et al. [62] proposed a method which us Convolutional Neural Networks (CNN) with max pooling to embedding the each character of words and showed their model could use fewer parameters while raising the performance.

4 Embeddings based on context

Another challenge facing word embeddings is to combine context-specific representation problems. For most downstream task in natural language processing (NLP), understanding the actual context is necessary. For example, in order to generate a German translation of English, the translation model needs to understand how the words in the English sentence are organized together. The text summary model also requires context to exploit the most important words. Models that perform semantic sentiment analysis need to understand how to pick out keywords that can change the emotion of a sentence. The question-and-answer model relies on an understanding of how the lexical distribution in the question changes the lexical distribution in the answer. In general, these models need to understand how context affects the meaning of a word. Recently, some work attempts to combine the word embeddings with the language model to solve the context problem. Here we briefly introduce some popular models based on context.

4.1 ELMo

Peters et al. [52] believes that a good word representation model should take into account two issues: the complex nature of word usage in semantics and grammar and these usages should change as the language environment changes. Therefore, they propose a deep contextualized word representation method to solve the above two problems as shown in Fig. 7. The characteristic of this algorithm is that the representation of each word is a function of the entire input statement.

The specific method is to train the bidirectional LSTM model on the large corpus with the language model as the target, and then use LSTM to generate the representation of the words. ELMo is named (Embeddings from Language Models). In this model, they use a two-way LSTM language model consisting of a forward and

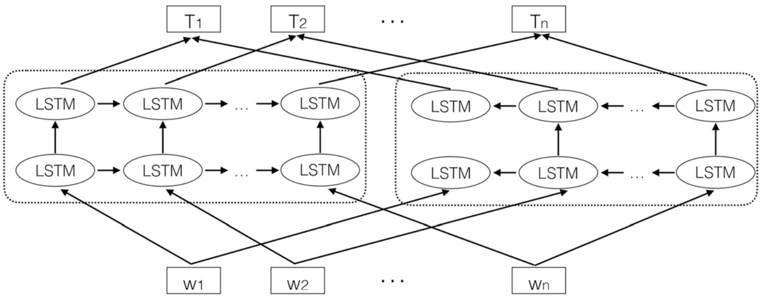


Fig. 7 The overall architecture of ELMo

a backward language model. The objective function is to take the maximum likelihood of the language models in both directions. Given a sequence of N tokens $\left(w_1, w_2, \ldots, w_N\right)$, the forward and backward language model as shown in formulas (16) and (17):

$$p\left(w_1, w_2, \ldots, w_N\right) = \prod_{k=1}^{N} p(w_k | w_1, w_2, \ldots, w_{k-1}) \tag{16}$$

$$p\left(w_1, w_2, \ldots, w_N\right) = \prod_{k=1}^{N} p(w_k | w_{k+1}, w_{k+2}, \ldots, t_N) \tag{17}$$

And the maximum likelihood of the language models is:

$$\sum_{k=1}^{N} (logp(w_k | w_1, w_2, \ldots w_{k-1}) + logp(w_k | w_{k+1}, w_{k+2}, \ldots, w_N)) \tag{18}$$

After pre-training the language model, ELMo uses the formula (19) as a word representation by summing each intermediate layer of the bidirectional language model.

$$R_k = \left\{ x_k^{LM}, \vec{h}_{k,j}^{LM}, \overset{\leftarrow}{h}_{k,j}^{LM} | j = 1, 2, \ldots, L \right\} = \left\{ h_{k,j}^{LM} | j = 0, \ldots, L \right\}. \tag{19}$$

4.2 OpenAI-GPT

Generative Pre-Training, also known as GPT [54], is a popular model that can effectively exploit the semantics of words in application context. Unlike ELMo, GPT uses Transformer [63] for feature extraction, while ELMo uses bidirectional LSTM, and GPT uses a one-way language model. The overall model of GPT as shown in Fig. 8. They still use the standard language model objective function as shown in

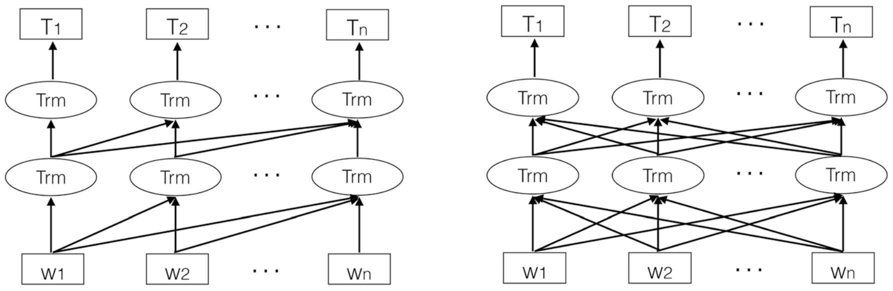


Fig. 8 The overall architecture of OpenAI-GPT (left) and BERT (right)

formula (20), which predicts the current word through the first k words, but on the language model network they use the Transformer decoder that utilizes a self-attention mechanism.

$$L_1(X) = \sum_i logP(w_i|w_{i-k}, \dots, w_{i-1};\theta) \tag{20}$$

In the feedforward network that processes the input text plus position information, the output is the conceptual distribution of words as follows:

$$h_0 = UW_e + W_p \tag{21}$$

$$h_l = Transformer_block\left(h_{l-1}\right) \tag{22}$$

$$P(u) = softmax\left(h_n W_e^T\right). \tag{23}$$

4.3 BERT

The commonality of the two methods above is that they use the same objective function in pre-training, and both use a one-way language model. Devlin et al. [53] consider that they do not make good use of contextual information. Although algorithms such as ELMo utilize forward and reverse language models, they are essentially a superposition of two unidirectional models. For many downstream tasks such as machine reading comprehension, it is important to be able to extract context information from both directions at the same time, but the existing methods have enormous limitations.

As a method of fine-tuning, Devlin et al. [53] proposes an improved scheme: BERT (Bidirectional Encoder Representations from Transformers) as shown in Fig. 8, which uses the bi-Transformer technique which can effectively exploit the deep semantic information of a sentence. The advantage of this is that the learned characterization can fuse the context in both directions. In terms of the input of the model, BERT also did more details. They used WordPiece embedding as a word embedding and added a position embedding and a sentence segmentation embedding. In the language model pre-training, they do not use the standard to predict the next word from left to right as the target task but propose two new tasks. The first task they call Masked Language Model (MLM), that is, in the input word sequence, randomly mask 15% of the words, and then the task is to predict the words which is masked.

In addition to modelling words based on context, there are also many models based on sentence level. Some previous works uses operations on embedding or matrix of words to obtain the phrase or sentence representations [64, 65]. Le and Mikolov [66] considered that words in the different paragraphs should have different embeddings. The empirical results showed that their paragraph embeddings performed well on several text classification and sentiment analysis tasks. Kiros et al. [67] proposed a model named Skip-thought which is based on Skip-gram. The difference is that Skip-gram uses word to predict the context of it while Skip-thought uses the central sentence to predict the context sentences. They evaluated their approach on semantic relatedness, paraphrase detection, image-sentence ranking, question-type classification and other four tasks, the experimental results showed that their model can produce robust sentence representations. Logeswaran and Lee [68] introduced Quick-thought model, which modify the predictive behavior of Skip-thought into a classification problem. The experimental results showed that this approach performed well on the classification tasks that require understanding sentence semantics. Conneau et al. [69] proposed InferSent that used Bi-LSTM as sentence encoder and proved their model performed better on natural language inference task.

5 Word embeddings of different language based on neural networks

The writing system and language structure of different languages are completely different, it is therefore necessary to design the specific word embeddings model for the specific language. As the Chinese for example, most of the existing Chinese word representation approaches are directly transferred from English while words are regarded as atomic tokens [9, 29, 30, 70]. Recently, there has been some methods for improving performance of word representations using subword-level information [32, 59, 71]. However, most of these methods are developed for European languages such as English and Spanish, due to the difference between Chinese and English writing systems, they cannot directly applicable to Chinese.

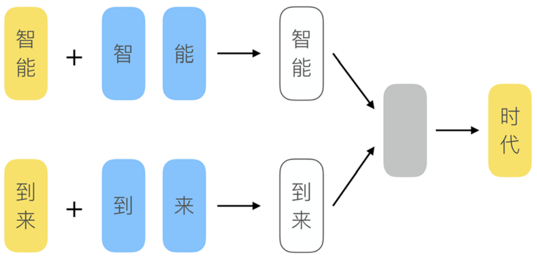
Different from English, the expression of Chinese characters is two dimensional that deliver information with the highest efficiency. Besides, Chinese is based on the characters rather than words, each word typically consists of less characters but can convey fruitful semantic information. Recently, there has a growing interest in approaches that exploiting the internal structural information of Chinese words, we take following popular models as examples.

5.1 CWE

Different from European languages, Chinese is based on characters rather than words. Therefore, characters in Chinese contain more semantic meanings than in English while most methods cannot fully utilize this. The character-enhanced word embedding model(CWE), presented by Chen et al. [38], is a word embeddings approach that combines the characters of Chinese with words into training.

As shown in Fig. 9, the model takes advantages of both internal characters and external contexts, extracts each characters of words, assigning the embeddings for them and then combine the character embeddings and word embeddings as input of model which is based on CBOW. This makes a connection between the words that share the same characters, because this paper's hypothesis is that the Chinese characters in the words "semantically compositional" have a certain characterization of the meaning of the words. In this model, the Chinese character set is denoted as C, using W to represent Chinese word vocabulary. Each character $c_i \in C$ is represented by vector c_i, and each

Fig. 9 The overall architecture of CWE model with an example



智能(Intelligence), 到来 (Arrive), 时代 (Era),
智(Intelligence), 能(Ability), 到(Reach), 来(Come)

word $w_i \in W$ is represented by vector w_i. The context word embeddings x_j is represented by word embeddings and character embeddings as following formula:

$$x_j = w_j \oplus \frac{1}{N_j} \sum_{k=1}^{N_j} c_k \tag{24}$$

where w_j is the word embedding of x_j, N_j is the number of characters in x_j, c_k is the embedding of the k-th character c_k in x_j, and $\oplus$ is the composition operation. And to ensure that such words are of similar length to words that have been added to character embedding, the author multiplies the embeddings obtained above by 1/2 for processing using formula (25):

$$x_j = \frac{1}{2}\left(w_j + \frac{1}{N_j} \sum_{k=1}^{N_j} c_k\right). \tag{25}$$

5.2 SCWE

Although CWE has considered the internal composition of words and increased the representation of semantic information, it ignores some problems between each word and their components (single words). CWE regards the contribution between words and words as consistent while Xu et al. [35] considers that the contribution between them should be different. They proposed a similarity-based character-enhanced word embedding model (SCWE), which considers the semantic of different characters of words. Specifically, they build a set F, which contains words, the similarity between words and characters, and the importance of different characters in the words. Suppose x_t in W is a compositional word, in SCWE:

$$\widehat{v_{x_t}} = \frac{1}{2}\left\{ v_{x_t} + \frac{1}{N_t \sum_{k=1}^{N_t} sim\left(x_t, c_k\right) v_c} \right\} \tag{26}$$

where v_{x_t} is the embedding of word x_t, N_t is the number of characters in word x_t, c_k is the k-th character in word x_t. And they also propose a model named SCWE+M to deal with the ambiguity problem. They consider that a character can have different semantics in specific position. Therefore, in SCWE+M, the embedding of word x_t is represented as shown in formula (27):

$$\widehat{v_{x_t}} = \frac{1}{2}\left\{ v_{x_t} + \frac{1}{N_t \sum_{k=1}^{N_t} sim(x_t, c_k) v_{c_k}^i} \right\}. \tag{27}$$

5.3 JWE

Chinese writing is a logical language, that is, a Chinese character can be a single word or part of a multi-syllable word. The characters themselves are usually composed of sub-characters, which is also semantic information. The characters that make up the Chinese word can indicate the semantics of the word, and the sub-character component (e.g., the radical and the component itself is a character) form a character that can indicate the semantic meaning of the character. The composition of characters can be roughly divided into two types: semantic components and phonetic components. The semantic component indicates the meaning of the character, while the voice component indicates the sound of the character. For example, the three-point water is the semantic component of lake and sea in Chinese. The use of sub-word information such as character and sub-character components can enhance Chinese word embedding with internal morphological semantics [72], the model is as shown in Fig. 10.

Based on the CBOW model, JWE predicts the target word using the average of the context word vectors, the average of the context character vectors, and the

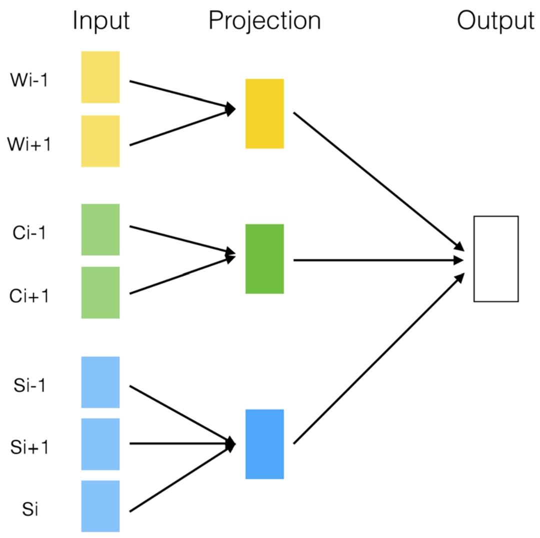


Fig. 10 The overall architecture of JWE model

average of the context sub-character vectors, and uses the sum of these three prediction losses as the objective function. Wi is the target word, w_{i-1} and w_{i+1} are its left and right words. c_{i-1} and c_{i+1} represent characters in the context. s_{i-1} and s_{i+1} represent sub-characters in the context, and s_i represents sub-characters of the target word w_i. And JWE aims to maximize the log likelihood of the three predictive conditional probabilities of the target word w_i:

$$L(w_i) = \sum_{3}^{k=1} logP(w_i|h_{i_k}) \tag{28}$$

where h_{i_1}, h_{i_2}, and h_{i_3} are combinations of context words, context characters, and context sub-characters, respectively. The conditional probability is defined by the softmax function as follows:

$$P(w_i|h_{i_k}) = \frac{exp\left(h_{i_k}^T \widehat{v_{w_i}}\right)}{\sum_{j=1}^{N} exp\left(h_{i_k}^T \widehat{v_{w_j}}\right)}, \quad k = 1, 2, 3 \tag{29}$$

where $\widehat{v_{w_i}}$ is the "output" vectors of word w_i, h_{i_1} is the average of the "input" vectors of words in the context, where T is the context window we defined, and so as h_{i_2} and h_{i_3}:

$$h_{i_i} = \frac{1}{2T} \sum_{-T \le j \le T, j \ne 0} v_{w_{i+J}}. \tag{30}$$

5.4 cw2vec

Chinese characters contain many internal structural features such as characters, radicals, and components. Since each character contains many strokes, it is similar to an English word containing many Latin letters. On this basis, Cao et al. [34] proposed a model cw2vec which used stroke information as a feature. As shown in Fig. 11. The model first divides the word into characters, extracts and combines the stroke sequence of characters then obtain the stroke n-grams information as the input information of model based on Skip-gram, they define the following similarity function between the target word w and context word c as following formula:

$$sim(w, c) = \sum_{q \in S(w)} \vec{q} \cdot \vec{c} \tag{31}$$

where q is an element of stroke n-grams set $S(w)$, $\vec{q}$ and $\vec{c}$ are the corresponding embeddings of w and c. And the softmax function to model the probability of predicting c given w they used according formula (32), Where c' is a word in the word vocabulary V.

$$p(c|w) = \frac{exp(sim(w, c))}{\sum_{c' \in V} exp(sim(w, c'))} \tag{32}$$

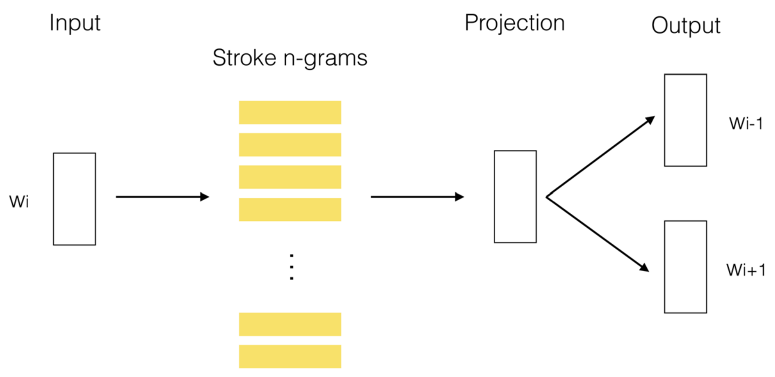


Fig. 11 The overall architecture of cw2vec model

Besides, there are also some methods aim at improving the Chinese word embeddings, Xu et al. [35] used external language information to obtain the semantics of words and better reflecting the difference in contribution of each character in a word. Experiments on the word similarity and text classification tasks showed that their method has the ability of disambiguating Chinese characters. Li et al. [73] extracted the components of characters for learning and also proved the effectiveness of their approach on the word similarity and text classification tasks. Su and Lee [74] designed a model called GWE that extract character features from the images using convolutional auto-encoders. Meanwhile, they also created several evaluation datasets in traditional Chinese. Sun et al. [75] used radical to enhanced the Chinese word embeddings, they showed that their radical-enhanced method performed well on both Chinese character similarity judgement and Chinese word segmentation tasks. In addition to this, there was a method of using the synonym dictionary to enhance the semantic information of the Chinese word embeddings [76], and also performed well on the word similarity, word analogy, and document classification tasks.

6 Neural networks applied to natural language processing

Neural network is a mathematical model that simulates the structure and function of human nerves. It is one of the most rapidly developing research directions in the field of machine learning in recent years. It is obtained in a series of important tasks of artificial intelligence such as image recognition, speech recognition, etc. At the same time, neural networks provide new perspectives and methods for in-depth research in the field of natural language processing. This chapter will briefly introduce several neural networks commonly used in the field of natural language processing.

6.1 RNN

The Recurrent Neural Network (RNN) is an important network structural model. By using the current input and the previous output at each moment, the output of the current time is generated in the same unit, so that it can be used to deal with problems with certain timing, such as signal processing, machine translation, stock trends and other natural language processing related tasks.

The overall structure of RNN is to accept the serialized input $x(0), x(1), \dots, x(t)$ and maintain a time-based hidden state vector $h(t)$. It adds the output h(t) of the previous moment to the input $x(t)$ of this moment. After an activation, such as tanh, it becomes the output of the next moment. Specifically, given the input x(t) and the output of the previous moment $h(t-1)$ in step t, the $h(t)$ is update according to the formula:

$$h(t) = f(Wh(t-1) + Vx(t)) \tag{33}$$

where W and V are weight matrices and f is the non-linear activation function such as tanh. $h(t)$ can be used as output directly and also can be processed as others such as the probability distributions.

6.2 LSTM

The RNN achieves a kind of continuous memory through its loop structure. However, as time goes on, it's easy to forget what happened a long time ago. The Long Short Term Memory Network (LSTM) solves this problem to a certain extent by making the processing of information in the neural network layer more complicated and precise.

Compare with RNN, in addition to maintaining the hidden state $h(t)$, the LSTM also maintains another memory vector $c(t)$ called the cell state. Furthermore, the LSTM explicitly controls the updating of $h(t)$ and $c(t)$ using the input gate $i(t)$, the forgetting gate $f(t)$ and the output gate $o(t)$. The three gate vectors are calculated according to the following formula:

$$i(t) = \sigma\left(W_i h(t-1) + V_i x(t)\right) \tag{34}$$

$$f(t) = \sigma\left(W_f h(t-1) + V_f x(t)\right) \tag{35}$$

$$i(t) = \sigma\left(W_o h(t-1) + V_o x(t)\right) \tag{36}$$

The cell state c (t) and hidden state h (t) are calculated as follows:

$$\tilde{c}(t) = tanh\left(W_c h(t-1) + V_c x(t)\right) \tag{37}$$

$$\tilde{c}(t) = f(t) \odot c(t-1) + i(t) \odot \tilde{c}(t). \tag{38}$$

$$h(t) = o(t) \odot c(t) \tag{39}$$

Based on the cell state and gates, the LSTM can maintain the memory information longer and has already been used for many natural language processing tasks.

6.3 GRU

The Gated Recurrent Unit (GRU) is a variant of LSTM. Compare with LSTM, GRU combines the forgetting gate and input gate into a single update gate. Because of this change, the final model is simpler than the standard LSTM model and is also a very popular variant.

The overall structure of GRU is simpler than LSTM, it only use the hidden state vector $h(t)$ and two gates include updating gate $z(t)$ and reset gate $r(t)$. These two gate vectors are calculated according to the following formula:

$$z(t) = \sigma\left(W_z h(t-1) + V_z x(t)\right) \tag{40}$$

$$r(t) = \sigma\left(W_r h(t-1) + V_r x(t)\right) \tag{41}$$

Then the hidden state $h(t)$ is calculated as follows:

$$\tilde{h}(t) = tanh(W_h h(t-1) + V_h r(t) \odot x(t) \tag{42}$$

$$h(t) = (1 - z(t)) \odot h(t-1) + z(t) \odot \tilde{h}(t-1). \tag{43}$$

7 Conclusions

Text representation is the basic work of natural language processing, with important theoretical significance and broad application prospects. This paper introduces the word embeddings based on neural network, compares the characteristics of different methods, and some applicable situations. The main work and contributions of this paper are summarized as follows.

This paper summarizes three main challenges that the word embeddings technology faced and also introduces some corresponding solutions. For the OOV problem, since the training data cannot completely include all the words, there are already many works attempt to obtain the solutions from the perspective of the sub-words. However, due to the diversity of words, the diversity of sub-words is quite sizable Therefore, how to reduce the length of the dictionary under the premise of maintaining the effect still needs to be explored.

From the application of sentence representations, the expression of each embeddings is highly dependent on specific tasks, that is, the key abstract features of the sentence required by different tasks are different levels. Therefore, in the future, we still need to explore how to better learn sentence representations and even higher levels of text representation for different languages in the future. Furthermore, although the current word vector model has been able to achieve breakthrough results in various tasks in the NLP field, these methods still have some shortcomings. For example, the parameters of the model are too large, the training process is very time consuming, and the current neural network-based methods are not interpretable. Therefore, how to reduce the cost of neural network training and improve the interpretability of the model is also a development direction.

Acknowledgements We thank the reviewers for their helpful comments and greatly acknowledge valuable contributions of our classmate Qiang Zhou in the preparation of our work. This work is supported by the Fundamental Research Funds for the Central Universities under Grant 2019XDA20.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Manning C, Raghavan P, Schütze H (2010) Introduction to information retrieval. Nat Lang Eng 16(1):100–103
2. Zhang Y, Jin R, Zhou Z-H (2010) Understanding bag-of-words model: a statistical framework. Int J Mach Learn Cybern 1(1–4):43–52
3. Firth JR (1957) A synopsis of linguistic theory, 1930–1955. In: Studies in linguistic analysis, Philological Society, Oxford
4. Harris ZS (1954) Distributional structure. Word 10(2–3):146–162
5. Dagan I, Lee L, Pereira FCN (1999) Similarity-based models of word co-occurrence probabilities. Mach Learn 34(1–3):43–69
6. Dagan I, Marcus S, Markovitch S (1993) Contextual word similarity and estimation from sparse data. In: Proceedings of the 31st annual meeting on association for computational linguistics, pp 164–171
7. Schütze H (1992) Context space. In: AAAI fall symposium on probabilistic approaches to natural language, pp 113–120
8. Schütze H (1992) Dimensions of meaning. In: Supercomputing'92: proceedings of the 1992 ACM/IEEE conference on supercomputing, IEEE, pp 787–796
9. Pennington J, Socher R, Manning CD (2014) Glove: global vectors for word representation. In: Empirical methods in natural language processing (EMNLP), pp 1532–1543
10. Baroni M, Dinu G, Kruszewski G (2014) Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In: Proceedings of the 52nd annual meeting of the association for computational linguistics, vol 1, pp 238–247
11. Turian J, Ratinov L, Bengio Y (2010) Word representations: a simple and general method for semi-supervised learning. In: Proceedings of the 48th annual meeting of the association for computational linguistics, Association for Computational Linguistics, pp 384–394
12. Lebret R, Collobert R (2014) Word embeddings through hellinger pca. In: EACL, p 482
13. Landauer TK, Foltz PW, Laham D (1998) An introduction to latent semantic analysis. Discourse Process 25(2–3):259–284
14. Dhillon PS, Foster DP, Ungar LH (2011) Multi-view learning of word embeddings via CCA. In: Advances in neural information processing systems, pp 199–207
15. Dhillon PS, Foster DP, Ungar LH (2015) Eigenwords: spectral word embeddings. J Mach Learn Res 16:3035–3078
16. Pereira F, Tishby N, Lee L (1993) Distributional clustering of english words. In: Proceedings of the 31st annual meeting on association for computational linguistics, pp 183–190
17. Brown PF, Desouza PV, Mercer RL, Pietra VJD, Lai JC (1992) Class-based n-gram models of natural language. Comput Linguist 18(4):467–479
18. Lin D, Wu X (2009) Phrase clustering for discriminative learning. In: Proceedings of the joint conference of the 47th annual meeting of the ACL and the 4th international joint conference on natural language processing, pp 1030–1038
19. Bengio Y, Ducharme R, Vincent P (2001) A neural probabilistic language model. In: Advances in neural information processing systems, pp 932–938
20. Bengio Y, Ducharme R, Vincent P, Jauvin C (2003) A neural probabilistic language model. J Mach Learn Res 3:1137–1155

21. Turian J, Ratinov L, Bengio Y (2010) Word representations: a simple and general method for semi-supervised learning. In: Proceedings of the 48th annual meeting of the association for computational linguistics (ACL), pp 384–394
22. Xu W, Rudnicky A (2000) Can artificial neural networks learn language models? In: Sixth international conference on spoken language processing
23. Mnih A, Hinton G (2007) Three new graphical models for statistical language modelling. In: Proceedings of the 24th international conference on machine learning, pp 641–648
24. Mnih A, Hinton GE (2008) A scalable hierarchical distributed language model. In: Advances in neural information processing systems, pp 1081–1088
25. Mnih A, Kavukcuoglu K (2013) Learning word embeddings efficiently with noise-contrastive estimation. In: Advances in neural information processing systems, pp 2265–2273
26. Morin F, Bengio Y (2005) Hierarchical probabilistic neural network language model. In: Proceedings of the international workshop on artificial intelligence and statistics, pp 246–252
27. Collobert R, Weston J (2008) A unified architecture for natural language processing: deep neural networks with multitask learning. In: International conference on machine learning
28. Mikolov T, Karafiát M, Burget L, Cernockỳ J, Khudanpur S (2010) Recurrent neural network based language model. In: 11th Annual conference of the international speech communication association, INTERSPEECH 2010, pp 1045–1048
29. Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. In: International conference on learning representations workshop Track
30. Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013) Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems, pp 3111–3119
31. Mikolov T, Yih WT, Zweig G (2013) Linguistic regularities in continuous space word representations. In: NAACL-HLT, pp 746–751
32. Bojanowski P, Grave E, Joulin A, Mikolov T (2017) Enriching word vectors with subword information. Trans Assoc Comput Linguist 5:135–146
33. Tissier J, Gravier C, Habrard A (2017) Dict2vec: learning word embeddings using lexical dictionaries. In: Conference on empirical methods in natural language processing (EMNLP), pp 254–263
34. Cao S, Lu W, Zhou J, Li X (2018) cw2vec: learning chinese word embeddings with stroke n-gram information. In: Thirty-second AAAI conference on artificial intelligence
35. Xu J, Liu J, Zhang L, Li Z, Chen H (2016) Improve chinese word embeddings by exploiting internal structure. In: NAACL-HLT
36. Collobert R, Weston J, Bottou L, Karlen M, Kavukcuoglu K, Kuksa P (2011) Natural language processing (almost) from scratch. J Mach Learn Res 12:2493–2537
37. Botha J, Blunsom P (2014) Compositional morphology for word representations and language modelling. Comput Sci 2014:1899–1907
38. Chen X, Lei X, Liu Z, Sun M, Luan H (2015) Joint learning of character and word embeddings. In: International conference on artificial intelligence
39. Kalchbrenner N, Blunsom P (2013) Recurrent convolutional neural networks for discourse compositionality. In: Workshop on CVSC, pp 119–126
40. Kalchbrenner N, Grefenstette E, Blunsom P (2014) A convolutional neural network for modelling sentences. In: Proceedings of the 52nd annual meeting of the association for computational linguistics, pp 655–665
41. Kim Y (2014) Convolutional neural networks for sentence classification. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp 1746–1751
42. Xu Y, Liu J (2017) Implicitly incorporating morphological information into word embedding. arXiv preprint arXiv:170102481
43. Conneau A, Kiela D, Schwenk H, Barrault L, Bordes A (2018) Supervised learning of universal sentence representations from natural language inference data. In: Conference on empirical methods in natural language processing
44. Talman A, Yli-Jyra A, Tiedemann J (2018) Natural language inference with hierarchical Bilstm max pooling architecture. arXiv preprint arXiv:180808762
45. Chung T, Xu B, Liu Y, Ouyang C, Li S, Luo L (2019) Empirical study on character level neural network classifier for chinese text. Eng Appl Artif Intell 80:1–7
46. Martinez-Rico JR, Martinez-Romo J, Araujo L (2019) Can deep learning techniques improve classification performance of vandalism detection in wikipedia? Eng Appl Artif Intell 78:248–259
47. Yao L, Zhang Y, Chen Q, Qian H, Wei B, Hu Z (2017) Mining coherent topics in documents using word embeddings and large-scale text data. Eng Appl Artif Intell 64:432–439

48. Ma X, Hovy E (2016) End-to-end sequence labeling via bi-directional lstm-cnns-crf. arXiv preprint arXiv:160301354
49. Shijia E, Xiang Y (2017) Chinese named entity recognition with character word mixed embedding. In: ACM on conference on information knowledge management
50. Sun Y, Lei L, Tang D, Nan Y, Ji Z, Wang X (2015) Modeling mention, context and entity with neural networks for entity disambiguation. In: Twenty-fourth international joint conference on artificial intelligence
51. Li J, Zhao S, Yang J et al (2018) WCP-RNN: a novel RNN-based approach for bio-NER in chinese EMRs. J Supercomput 2018:1–18
52. Peters ME, Neumann M, Iyyer M, Gardner M, Clark C, Lee K, Zettlemoyer L (2018) Deep contextualized word representations. arXiv preprint arXiv:1802.05365
53. Devlin J, Chang MW, Lee K, Toutanova K (2018) Bert: pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805
54. Radford A, Narasimhan K, Salimans T, Sutskever I (2018) Improving language understanding by generative pre-training. URL https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/languageunsupervised/languageunderstandingpaper.pdf
55. Joulin A, Grave E, Bojanowski P, Mikolov T (2016) Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759
56. Luong T, Socher R, Manning CD (2013) Better word representations with recursive neural networks for morphology. In: Proceedings of the conference
57. Socher R, Lin C, Manning C, Ng AY (2011) Parsing natural scenes and naturallanguage with recursive neural networks. In: Proceedings of the 28th international conference on machine learning (ICML-11), pp 129–136
58. Sennrich R, Haddow B, Birch A (2015) Neural machine translation of rare words with subword units. arXiv preprint arXiv:150807909
59. Cotterell R, Schütze H (2015) Morphological word-embeddings. In: Proceedings of the 2015 conference of the North American chapter of the association for computational linguistics: human language technologies, pp 1287–1292
60. Bian J, Gao B, Liu TY (2014) Knowledge-powered deep learning for word embedding. In: Joint European conference on machine learning and knowledge discovery in databases, Springer, pp 132–148
61. Cao K, Rei M (2016) A joint model for word embedding and word morphology. arXiv preprint arXiv:1606.02601
62. Kim Y, Jernite Y, Sontag D, Rush AM (2016) Character-aware neural language models. In: Thirtieth AAAI conference on artificial intelligence
63. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Polosukhin I (2017) Attention is all you need. In: Advances in neural information processing systems, pp 5998–6008
64. Mitchell J, Lapata M (2008) Vector-based models of semantic composition. In: Proceedings of ACL-08: HLT, pp 236–244
65. Blacoe W, Lapata M (2012) A comparison of vector-based representations for semantic composition. In: Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning, Association for Computational Linguistics, pp 546–556
66. Le QV, Mikolov T (2014) Distributed representations of sentences and documents. In: International conference on machine learning, pp 1188–1196
67. Kiros R, Zhu Y, Salakhutdinov RR, Zemel R, Urtasun R, Torralba A, Fidler S (2015) Skip-thought vectors. In: Advances in neural information processing systems, pp 3294–3302
68. Logeswaran L, Lee H (2018) An efficient framework for learning sentence representations. arXiv preprint arXiv:1803.02893
69. Conneau A, Kiela D, Schwenk H et al (2017) Supervised learning of universal sentence representations from natural language inference data. arXiv preprint arXiv:1705.02364
70. Levy O, Goldberg Y (2014) Linguistic regularities in sparse and explicit word representations. In: Proceedings of the eighteenth conference on computational natural language learning, pp 171–180
71. Heinzerling B, Strube M (2017) Bpemb: tokenization-free pre-trained subword embeddings in 275 languages. arXiv preprint arXiv:1710.02187
72. Xin JYXJHH, Song Y (2017) Joint embeddings of chinese words, characters, and fine-grained subcharacter components. In: EMNLP
73. Li Y, Li W, Sun F, Li S (2015) Component-enhanced chinese character embeddings. arXiv preprint arXiv:150806669

74. Su TR, Lee HY (2017) Learning chinese word representations from glyphs of characters. arXiv preprint arXiv:170804755
75. Sun Y, Lei L, Nan Y, Ji Z, Wang X (2014) Radical-enhanced chinese character embedding. Lect Not Comput Sci 8835:279–286
76. Yang L, Sun M (2015) Improved learning of Chinese word embeddings with semantic knowledge. In: Chinese computational linguistic and natural language processing based on naturally annotated big data. Springer, pp 15–25