Chapter 4
Sentence Representation

Abstract Sentence is an important linguistic unit of natural language. Sentence Representation has remained as a core task in natural language processing, because many important applications in related fields lie on understanding sentences, for example, summarization, machine translation, sentiment analysis, and dialogue system. Sentence representation aims to encode the semantic information into a real-valued representation vector, which will be utilized in further sentence classification or matching tasks. With large-scale text data available on the Internet and recent advances on deep neural networks, researchers tend to employ neural networks (e.g., convolutional neural networks and recurrent neural networks) to learn low-dimensional sentence representations and achieve great progress on relevant tasks. In this chapter, we first introduce the one-hot representation for sentences and the n-gram sentence representation (i.e., probabilistic language model). Then we extensively introduce neural-based models for sentence modeling, including feedforward neural network, convolutional neural network, recurrent neural network, and the latest Transformer, and pre-trained language models. Finally, we introduce several typical applications of sentence representations.

4.1 Introduction

Natural language sentences consist of words or phrases, follow grammatical rules, and convey complete semantic information. Compared with words and phrases, sentences have more complex structures, including both sequential and hierarchical structures, which are essential for understanding sentences. In NLP, how to represent sentences is critical for related applications, such as sentence classification, sentiment analysis, sentence matching, and so on.

Before deep learning took off, sentences were usually represented as one-hot vectors or TF-IDF vectors, following the assumption of bag-of-words. In this case, a sentence is represented as a vocabulary-sized vector, in which each element represents the importance of a specific word (either term frequency or TF-IDF) to the sentence. However, this method confronts two issues. Firstly, the dimension of such representation vectors is usually up to thousands or millions. Thus, they usually face sparsity problem and bring in computational efficiency problem. Secondly, such a

Z. Liu et al., *Representation Learning for Natural Language Processing*,
https://doi.org/10.1007/978-981-15-5573-2_4

representation method follows the bag-of-words assumption and ignores the sequential and structural information, which can be crucial for understanding the semantic meanings of sentences.

Inspired by recent advances of deep learning models in computer vision and speech, researchers proposed to model sentences with deep neural networks, such as convolutional neural network, recurrent neural network, and so on. Compared with conventional word frequency-based sentence representations, deep neural networks can capture the internal structures of sentences, e.g., sequential and dependency information, through convolutional or recurrent operations. Thus, neural network-based sentence representations have achieved great success in sentence modeling and NLP tasks.

4.2 One-Hot Sentence Representation

One-hot representation is the most simple and straightforward method for word representation tasks. This method represents each word with a fixed length binary vector. Specifically, for a vocabulary $V = \{w_1, w_2, \ldots, w_{|V|}\}$, the one-hot representation of word w is $\mathbf{w} = [0, \ldots, 0, 1, 0, \ldots, 0]$. Based on the one-hot word representation and the vocabulary, it can be extended to represent a sentence $s = \{w_1, w_2, \ldots, w_l\}$ as

$$\mathbf{s} = \sum_{k=1}^{l} \mathbf{w}_i, \tag{4.1}$$

where l indicates the length of the sentence s. The sentence representation $\mathbf{s}$ is the sum of the one-hot representations of n words within the sentence, i.e., each element in $\mathbf{s}$ represents the Term Frequency (TF) of the corresponding word.

Moreover, researchers usually take the importance of different words into consideration, rather than treat all the words equally. For example, the function words such as "a", "an", and "the" usually appear in different sentences, and reserve little meanings. Therefore, the Inverse Document Frequency (IDF) is employed to measure the importance of w_i in V as follows:

$$\text{idf}_{w_i} = \log \frac{|D|}{\text{df}_{w_i}}, \tag{4.2}$$

where $|D|$ is the number of all documents in the corpus D and df_{w_i} represents the Document Frequency (DF) of w_i.

With the importance of each word, the sentences are represented more precisely as follows:

$$\hat{\mathbf{s}} = \mathbf{s} \otimes \text{idf}, \tag{4.3}$$

where $\otimes$ is the element-wise product.

Here, $\hat{\mathbf{s}}$ is the TF-IDF representation of the sentence s.

4.3 Probabilistic Language Model

One-hot sentence representation usually neglects the structure information in a sentence. To address this issue, researchers propose probabilistic language model, which treats n-grams rather than words as the basic components. An n-gram means a subsequence of words in a context window of length n, and probabilistic language model defines the probability of a sentence $s = [w_1, w_2, \ldots, w_l]$ as

$$P(s) = \prod_{i=1}^{l} P(w_i | w_1^{i-1}). \tag{4.4}$$

Actually, model indicated in Eq. (4.4) is not practicable due to its enormous parameter space. In practice, we simplify the model and set an n-sized context window, assuming that the probability of word w_i only depends on $[w_{i-n+1} \cdots w_{i-1}]$. More specifically, an n-gram language model predicts word w_i in the sentence s based on its previous $n-1$ words. Therefore, the simplified probability of a sentence is formalized as

$$P(s) = \prod_{i=1}^{l} P(w_i | w_{i-n+1}^{i-1}), \tag{4.5}$$

where the probability of selecting the word w_i can be calculated from n-gram model frequency counts:

$$P(w_i | w_{i-n+1}^{i-1}) = \frac{P(w_{i-n+1}^{i})}{P(w_{i-n+1}^{i-1})}. \tag{4.6}$$

Typically, the conditional probabilities in n-gram language models are not calculated directly from the frequency counts, since it suffers severe problems when confronted with any n-grams that have not explicitly been seen before. Therefore, researchers proposed several types of smoothing approaches, which assign some of the total probability mass to unseen words or n-grams, such as "add-one" smoothing, Good-Turing discounting, or back-off models.

n-gram model is a typical probabilistic language model for predicting the next word in an n-gram sequence, which follows the Markov assumption that the probability of the target word only relies on the previous $n-1$ words. The idea is employed by most of current sentence modeling methods. n-gram language model is used as an approximation of the true underlying language model. This assumption is crucial because it massively simplifies the problem of learning the parameters of language models from data. Recent works on word representation learning [3, 40, 43] are mainly based on the n-gram language model.

4.4 Neural Language Model

Although smoothing approaches could alleviate the sparse problem in the probabilistic language model, it still performs poorly for those unseen or uncommon words and n-grams. Moreover, since probabilistic language models are constructed on larger and larger texts, the number of unique words (the vocabulary) increases and the number of possible sequences of words increases exponentially with the size of the vocabulary, causing a data sparsity problem. Thus statistics are needed to estimate probabilities accurately.

To address this issue, researchers propose neural language models which use continuous representations or embeddings of words and neural networks to make their predictions, in which embeddings in the continuous space help to alleviate the curse of dimensionality in language modeling, and neural networks avoid this problem by representing words in a distributed way, as nonlinear combinations of weights in a neural net [2]. An alternate description is that a neural network approximates the language function. The neural net architecture might be feedforward or recurrent, and while the former is simpler, the latter is more common.

Similar to probabilistic language models, neural language models are constructed and trained as probabilistic classifiers that learn to predict a probability distribution:

$$P(s) = \prod_{i=1}^{l} P(w_i|\mathbf{w}_1^{i-1}), \tag{4.7}$$

where the conditional probability of the selecting word w_i can be calculated by various kinds of neural networks such as feedforward neural networks, recurrent neural networks, and so on. In the following sections, we will introduce these neural language models in detail.

4.4.1 Feedforward Neural Network Language Model

The goal of neural network language model is to estimate the conditional probability $P(w_i|w_1, \ldots, w_{i-1})$. However, the feedforward neural network (FNN) lacks an effective way to represent the long-term historical context. Therefore, it adopts the idea of n-gram language models to approximate the conditional probability, which assumes that each word in a word sequence more statistically depends on those words closer to it, and only $n-1$ context words are used to calculate the conditional probability, i.e., $P(w_i|\mathbf{w}_1^{i-1}) \approx P(w_i|\mathbf{w}_{i-n+1}^{i-1})$.

The overall architecture of the FNN language model is proposed by [3]. To evaluate the conditional probability of the word w_i, it first projects its $n-1$ context-related words to their word vector representations $\mathbf{x} = [\mathbf{w}_{i-n+1}, \ldots, \mathbf{w}_{i-1}]$, and then feeds them into an FNN, which can be generally represented as

$$\mathbf{y} = \mathbf{M} f(\mathbf{W}\mathbf{x} + \mathbf{b}) + \mathbf{d}, \tag{4.8}$$

where $\mathbf{W}$ is a weighted matrix to transform word vectors to hidden representations, $\mathbf{M}$ is a weighted matrix for the connections between the hidden layer and the output layer, and $\mathbf{b}$, $\mathbf{d}$ are bias vectors. And then the conditional probability of the word w_i can be calculated as

$$P(w_i|\mathbf{w}_{i-n}^{i-1}) = \frac{\exp(\mathbf{y}_{w_i})}{\sum_j \exp(\mathbf{y}_j)}. \tag{4.9}$$

4.4.2 Convolutional Neural Network Language Model

The Convolutional Neural Network (CNN) is the family of neural network models that features a type of layer known as the convolutional layer. This layer can extract features by a learnable filter (or kernel) at the different positions of an input. Pham et al. [47] propose the CNN language model to enhance the FNN language model. The proposed CNN network is produced by injecting a convolutional layer after the word input representation $\mathbf{x} = [\mathbf{w}_{i-n}, \ldots, \mathbf{w}_{i-1}]$. Formally, the convolutional layer involves a sliding window of the input vectors centered on each word vector using a parameter matrix W_c, which can be generally represented as

$$\mathbf{y} = \mathbf{M}\big(\max(\mathbf{W_c}\mathbf{x})\big), \tag{4.10}$$

where $\max(\cdot)$ indicates a max-pooling layer. The architecture of CNN is shown in Fig. 4.1.

Moreover, [12] also introduces a convolutional neural network for language modeling with a novel gating mechanism.

4.4.3 Recurrent Neural Network Language Model

To address the lack of ability for modeling long-term dependency in the FNN language model, [41] proposes a Recurrent Neural Network (RNN) language model which applies RNN in language modeling. RNNs are fundamentally different from FNNs in the sense that they operate on not only an input space but also an internal state space, and the internal state space enables the representation of sequentially extended dependencies. Therefore, the RNN language model can deal with those sentences of arbitrary length. At every time step, its input is the vector of its previous word instead of the concatenation of vectors of its n previous words, and the information of all other previous words can be taken into account by its internal state. Formally, the RNN language model can be defined as

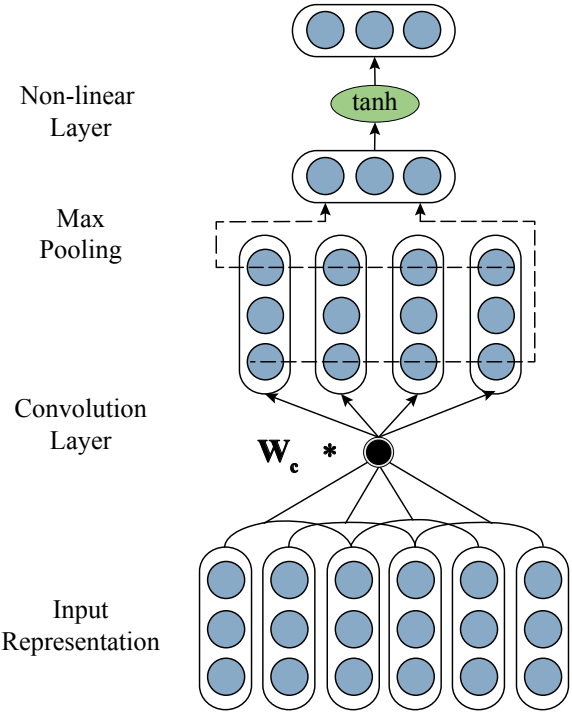

Fig. 4.1 The architecture of CNN

$$\mathbf{h}_i = f(\mathbf{W}_1\mathbf{h}_{i-1} + \mathbf{W}_2\mathbf{w}_i + \mathbf{b}), \tag{4.11}$$

$$\mathbf{y} = \mathbf{M}\mathbf{h}_{i-1} + \mathbf{d}, \tag{4.12}$$

where $\mathbf{W}_1$, $\mathbf{W}_2$, $\mathbf{M}$ are weighted matrices and $\mathbf{b}$, $\mathbf{d}$ are bias vectors. Here, the RNN unit can also be implemented by LSTM or GRU. The architecture of RNN is shown in Fig. 4.2.

Recently, researchers make some comparisons among neural network language models with different architectures on both small and large corpora. The experimental results show that, generally, the RNN language model outperforms the CNN language model.

4.4.4 Transformer Language Model

In 2018, Google proposed a pre-trained language model (PLM), called BERT, which achieved state-of-the-art results on a variety of NLP tasks. At that time, it was very big news. Since then, all the NLP researchers began to consider how PLMs can benefit their research tasks.

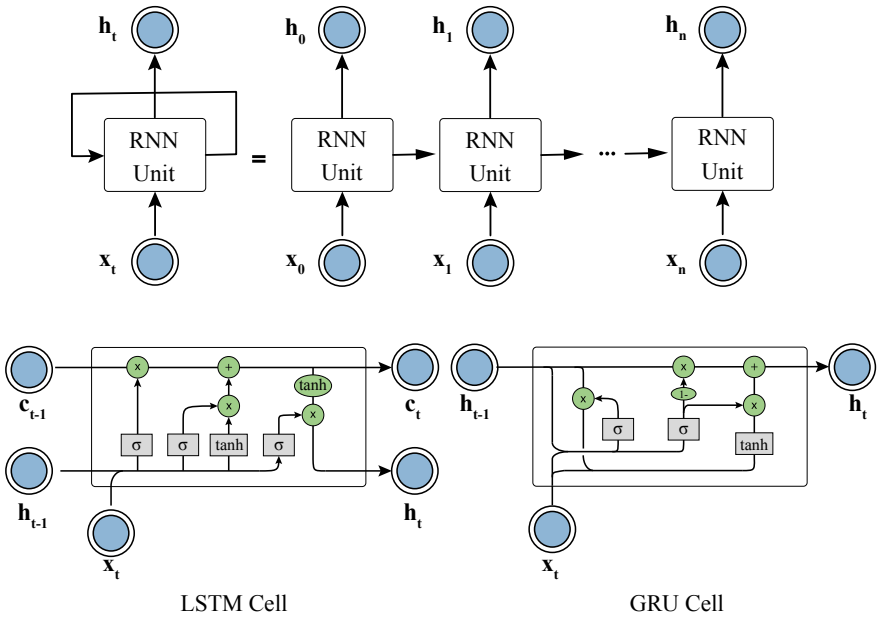

Fig. 4.2 The architecture of RNN

In this section, we will first introduce the Transformer architecture and then talk about BERT and other PLMs in detail.

4.4.4.1 Transformer

Transformer [65] is a nonrecurrent encoder-decoder architecture with a series of attention-based blocks. For the encoder, there are 6 layers and each layer is composed of a multi-head attention sublayer and a position-wise feedforward sublayer. And there is a residual connection between sublayers. The architecture of the Transformer is as shown in Fig. 4.3.

There are several attention heads in the multi-head attention sublayer. A *head* represents a scaled dot-product attention structure, which takes the query matrix $\mathbf{Q}$, the key matrix $\mathbf{K}$, and the value matrix $\mathbf{V}$ as the inputs, and the output is computed by

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}, \tag{4.13}$$

where d_k is the dimension of query matrix.

The multi-head attention sublayer linearly projects the input hidden states H several times into the query matrix, the key matrix, and the value matrix for h heads. The dimensions of the query, key, and value vectors are d_k, d_k, and d_v, respectively.

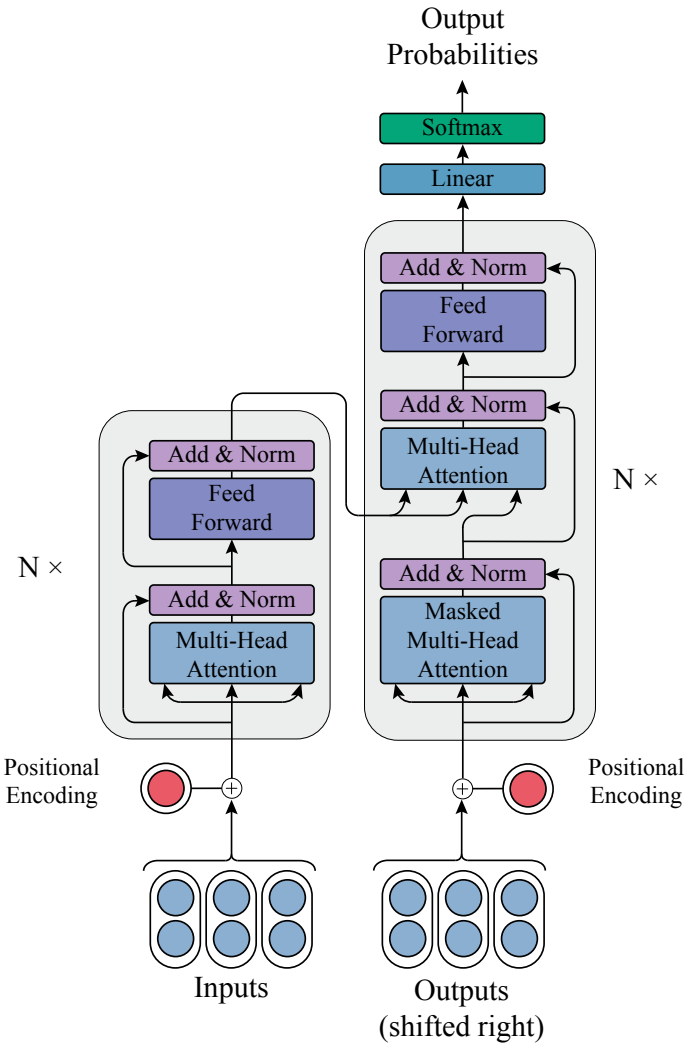

Fig. 4.3 The architecture of Transformer

The multi-head attention sublayer could be formulated as

$$\text{Multihead}(H) = [head_1, head_2, \ldots, head_h]\mathbf{W}^O, \tag{4.14}$$

where $head_i = \text{Attention}(\mathbf{H}\mathbf{W}_i^Q, \mathbf{H}\mathbf{W}_i^K, \mathbf{H}\mathbf{W}_i^V)$, and $\mathbf{W}_i^Q$, $\mathbf{W}_i^K$ and $\mathbf{W}_i^V$ are linear projections. $\mathbf{W}^O$ is also a linear projection for the output. Here, the fully connected position-wise feedforward sublayer contains two linear transformations with ReLU activation:

$$\text{FFN}(x) = \mathbf{W}_2 \max(0, \mathbf{W}_1 x + \mathbf{b}_1) + \mathbf{b}_2. \tag{4.15}$$

Transformer is better than RNNs for modeling the long-term dependency, where all tokens will be equally considered during the attention operation. The Transformer was proposed to solve the problem of machine translation. Since Transformer has a very powerful ability to model sequential data, it becomes the most popular backbone of NLP applications.

4.4.4.2 Transformer-Based PLM

Neural models can learn large amounts of language knowledge from language modeling. Since the language knowledge covers the demands of many downstream NLP tasks and provides powerful representations of words and sentences, some researchers found that knowledge can be transferred to other NLP tasks easily. The transferred models are called Pre-trained Language Models (PLMs).

Language modeling is the most basic and most important NLP task. It contains a variety of knowledge for language understanding, such as linguistic knowledge and factual knowledge. For example, the model needs to decide whether it should add an article before a noun. This requires linguistic knowledge about articles. Another example is the question of what is the following word after "Trump is the president of". The answer is "America", which requires factual knowledge. Since language modeling is very complex, the models can learn a lot from this task.

On the other hand, language modeling only requires plain text without any human annotation. With this feature, the models can learn complex NLP abilities from a very large-scale corpus. Since deep learning needs large amounts of data and language modeling can make full use of all texts in the world, PLMs significantly benefit the development of NLP research.

Inspired by the success of the Transformer, GPT [50] and BERT [14] begin to adopt the Transformer as the backbone of the pre-trained language models. GPT and BERT are the most representative Transformer-based pre-trained language models (PLMs). Since they achieved state-of-the-art performance on various NLP tasks, nearly all PLMs after them are based on the Transformer. In this subsection, we will talk about GPT and BERT in more detail.

GPT is the first work to pretrain a PLM based on the Transformer. The training procedure of GPT [50] contains two classic stages: generative pretraining and discriminative fine-tuning.

In the pretraining stage, the input of the model is a large-scale unlabeled corpus denoted as $\mathscr{U} = \{u_1, u_2, \ldots, u_n\}$. The pretraining stage aims to optimize a language model. The learning objective over the corpus is to maximize a conditional likelihood in a fixed-size window:

$$\mathscr{L}_1(\mathscr{U}) = \sum_i \log P(u_i | u_{i-k}, \ldots, u_{i-1}; \Theta), \tag{4.16}$$

where k represents the size of the window, the conditional likelihood P is modeled by a neural network with parameters Θ.

For a supervised dataset χ, the input is a sequence of words $s = (w_1, w_2, .., w_l)$ and the output is a label y. The pretraining stage provides an advantageous start point of parameters that can be used to initialize subsequent supervised tasks. At this occasion, the objective is a discriminative task that maximizes the conditional possibility distribution:

$$\mathscr{L}_2(\chi) = \sum_{(s,y)} \log P(y|w_1, \ldots, w_l), \tag{4.17}$$

where $P(y|w_1, \ldots, w_l)$ is modeled by a K-layer Transformer. After the input tokens pass through the pretrained GPT, a hidden vector of the final layer $\mathbf{h}_l^K$ will be produced. To obtain the output distribution, a linear transformation layer is added, which has the same size as the number of labels:

$$P(y|w_1, \ldots, w_m) = \text{Softmax}(\mathbf{W}_y \mathbf{h}_l^K). \tag{4.18}$$

The final training objective is combined with a language modeling $\mathscr{L}_1$ for better generalization:

$$\mathscr{L}(\chi) = \mathscr{L}_2(\chi) + \lambda * \mathscr{L}_1(\chi), \tag{4.19}$$

where λ is a weight hyperparameter.

BERT [14] is a milestone work in the field of PLM. BERT achieved significant empirical results on 17 different NLP tasks, including SQuAD (outperform human being), GLUE (7.7% point absolute improvement), MultiNLI (4.6% point absolute improvement), etc. Compared to GPT, BERT uses a bidirectional deep Transformer as the model backbone. As illustrated in Fig. 4.4, BERT contains pretraining and fine-tuning stages.

In the pretraining stage, two objectives are designed: *Masked Language Model (MLM)* and *Next Sentence Prediction (NSP)*. (1) For MLM, tokens are randomly

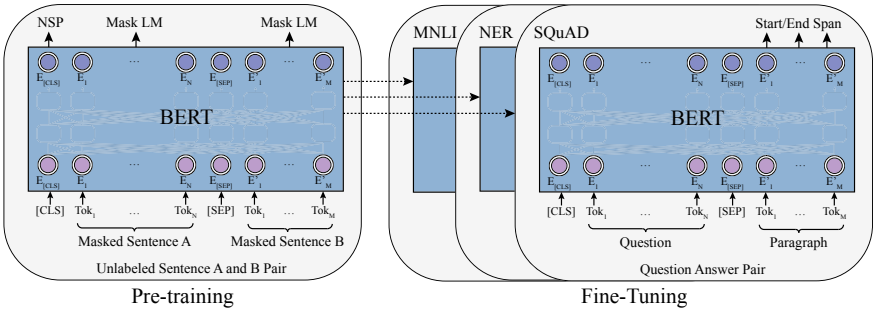

Fig. 4.4 The pretraining and fine-tuning stages for BERT

masked with a special token [MASK]. The training objective is to predict the masked tokens based on the contexts. Compared with the standard unidirectional conditional language model, which can only be trained in one direction, MLM aims to train a deep bidirectional representation model. This task is inspired by *Cloze* [64]. (2) The objective of NSP is to capture relationships between sentences for some sentence-based downstream tasks such as natural language inference (NLI) and question answering (QA). In this task, a binary classifier is trained to predict whether the sentence is the next sentence for the current. This task effectively captures the deep relationship between sentences, exploring semantic information from a different level.

After pretraining, BERT can capture various language knowledge for downstream supervised tasks. By modifying inputs and outputs, BERT can be fine-tuned for any NLP tasks, which contain the applications with the input of single text or text pairs. The input consists of sentence A and sentence B, which can represent (1) sentence pairs in paraphrase, (2) hypothesis-premise pairs in entailment, (3) question-passage pairs in QA, and (4) text-Ø for text classification task or sequence tagging. For the output, BERT can produce the token-level representation for each token, which is used to sequence tagging task or question answering. Besides, the special token [CLS] in BERT is fed into the classification layer for sequence classification.

4.4.4.3 PLM Family

Pre-trained language models have rapid progress after BERT. We summarize several important directions of PLMs and show some representative models and their relationship in Fig. 4.5.

Here is a brief introduction of the PLMs after BERT. Firstly, there are some variants of BERT for better general language representation, such as RoBERTa [38] and XLNet [70]. These models mainly focus on the improvement of pretraining tasks. Secondly, some people work on pretrained generation models, such as MASS [57] and UniLM [15]. These models achieve promising results on the generation tasks instead of the Natural Language Understanding (NLU) tasks used by BERT. Thirdly, the sentence pair format of BERT inspired works on the cross-lingual and cross-modal fields. XLM [8], ViLBERT [39], and VideoBERT [59] are the important works in this direction. Lastly, there are some works [46, 81] that explore to incorporate external knowledge into PLMs since some low-frequency knowledge cannot be efficiently learned by PLMs.

4.4.5 Extensions

4.4.5.1 Importance Sampling

Inspired by the contrastive divergence model, [4] proposes to adopt importance sampling to accelerate the training of neural language models. They first normalize the

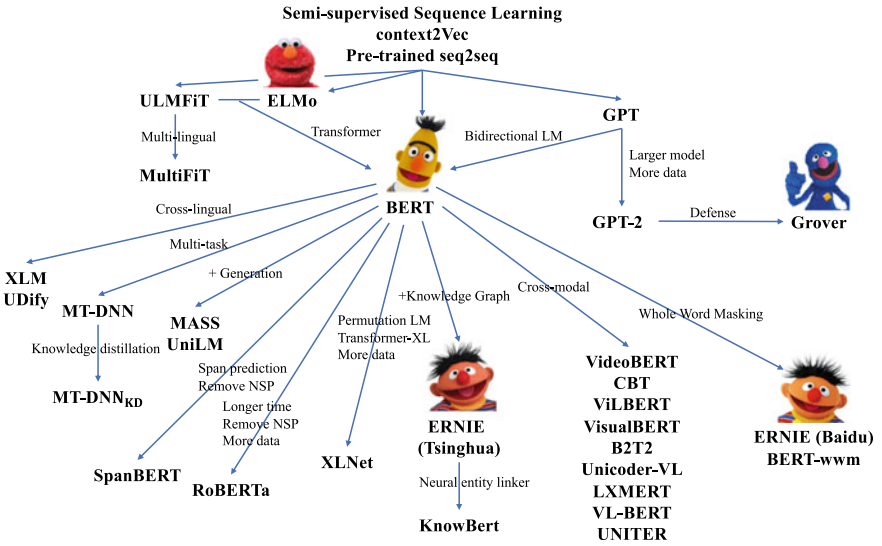

Fig. 4.5 The Pre-trained language model family

outputs of neural network language model and view neural network language models as a special case of energy-based probability models as following:

$$P(w_i|\mathbf{w}_{i-n}^{i-1}) = \frac{\exp(-y_{w_i})}{\sum_j \exp(-y_j)}. \tag{4.20}$$

The key idea of importance sampling is to approximate the mean of log-likelihood gradient of the loss function of neural network language model by sampling several important words instead of calculating the explicit gradient. Here, the log-likelihood gradient of the loss function of neural network language model can be generally represented as

$$\begin{aligned}\frac{\partial P(w_i|\mathbf{w}_{i-n}^{i-1})}{\partial \theta} &= -\frac{\partial y_{w_i}}{\partial \theta} + \sum_{j=1}^{|V|} P(w_j|\mathbf{w}_{i-n}^{i-1}) \frac{\partial y_j}{\partial \theta} \\ &= -\frac{\partial y_i}{\partial \theta} + \mathbb{E}_{w_k \sim P}\left[\frac{\partial y_k}{\partial \theta}\right],\end{aligned} \tag{4.21}$$

where θ indicates all parameters of the neural network language model. Here, the log-likelihood gradient of the loss function consists of two parts including positive gradient for target word w_i and negative gradient for all words w_j, i.e., $\mathbb{E}_{w_i \sim P}[\frac{\partial y_j}{\partial \theta}]$. Here, the second part can be approximated by sampling important words following the probability distribution P:

$$\mathbb{E}_{w_k \sim P}\left[\frac{\partial y_k}{\partial \theta}\right] \approx \sum_{w_k \in V'} \frac{1}{|V'|} \frac{\partial y_k}{\partial \theta}, \tag{4.22}$$

where V' is the word set sampled under P.

However, since we cannot obtain probability distribution P in advance, it is impossible to sample important words following the probability distribution P. Therefore, importance sampling adopts a Monte Carlo scheme which uses an existing proposal distribution Q to approximate P, and then we have

$$\mathbb{E}_{w_k \sim P}\left[\frac{\partial y_k}{\partial \theta}\right] \approx \frac{1}{|V''|} \sum_{w_l \in V''} \frac{\partial y_l}{\partial \theta} P(w_l | \mathbf{w}_{i-n}^{i-1}) / Q(w_l), \tag{4.23}$$

where V'' is the word set sampled under Q. Moreover, the sample size of importance sampling approach should be increased as training processes in order to avoid divergence, which aims to ensure its effective sample size S:

$$S = \frac{(\sum_{w_l \in V''} r_l)^2}{\sum_{w_l \in V''} r_l^2}, \tag{4.24}$$

where r_l is further defined as

$$r_l = \frac{P(w_l | \mathbf{w}_{i-n}^{i-1}) / Q(w_l)}{\sum_{w_j \in V''} P(w_j | \mathbf{w}_{i-n}^{i-1}) / Q(w_j)}. \tag{4.25}$$

4.4.5.2 Word Classification

Besides important sampling, researchers [7, 22] also propose class-based language model, which adopts word classification to improve the performance and speed of a language model. In class-based language model, all words are assigned to a unique class, and the conditional probability of a word given its context can be decomposed into the probability of the word's class given its previous words and the probability of the word given its class and history, which is formally defined as

$$P(w_i | \mathbf{w}_{i-n}^{i-1}) = \sum_{c(w_i) \in C} P(w_i | c(w_i)) P(c(w_i) | \mathbf{w}_{i-n}^{i-1}), \tag{4.26}$$

where C indicates the set of all classes and $c(w_i)$ indicates the class of word w_i.

Moreover, [44] proposes a hierarchical neural network language model, which extends word classification to a hierarchical binary clustering of words in a language model. Instead of simply assigning each word with a unique class, it first builds a hierarchical binary tree of words according to the word similarity obtained from WordNet. Next, it assigns a unique bit vector $c(w_i) = [c_1(w_i), c_2(w_i), \ldots, c_l(w_i)]$ for

each word, which indicates the hierarchical classes of them. And then the conditional probability of each word can be defined as

$$P(w_i|\mathbf{w}_{i-n}^{i-1}) = \prod_{j=0}^{l} P(c_j(w_i)|c_1(w_i), c_2(w_i), \ldots, c_{j-1}(w_i), \mathbf{w}_{i-n}^{i-1}). \tag{4.27}$$

The hierarchical neural network language model can achieve $O(k/\log k)$ speed up as compared to a standard language model. However, the experimental results of [44] show that although the hierarchical neural network language model achieves an impressive speed up for modeling sentences, it has worse performance than the standard language model. The reason is perhaps that the introduction of hierarchical architecture or word classes imposes a negative influence on the word classification by neural network language models.

4.4.5.3 Caching

Caching is also one of the important extensions in language model. A type of cache-based language model assumes that each word in recent context is more likely to appear again [58]. Hence, the conditional probability of a word can be calculated by the information from history and caching:

$$P(w_i|\mathbf{w}_{i-n}^{i-1}) = \lambda P_s(w_i|\mathbf{w}_{i-n}^{i-1}) + (1-\lambda) P_c(w_i|\mathbf{w}_{i-n}^{i-1}), \tag{4.28}$$

where $P_s(w_i|\mathbf{w}_{i-n}^{i-1})$ indicates the conditional probability generated by standard language and $P_c(w_i|\mathbf{w}_{i-n}^{i-1})$ indicates the conditional probability generated by caching, and λ is a constant.

Another cache-based language model is also used to speed up the RNN language modeling [27]. The main idea of this approach is to store the outputs and states of language models for future predictions given the same contextual history.

4.5 Applications

In this section, we will introduce two typical sentence-level NLP applications including text classification and relation extraction, as well as how to utilize sentence representation for these applications.

4.5.1 Text Classification

Text classification is a typical NLP application and has lots of important real-world tasks such as parsing and semantic analysis. Therefore, it has attracted the interest of many researchers. The conventional text classification models (e.g., the LDA [6] and tree kernel [48] models) focus on capturing more contextual information and correct word order by extracting more useful and distinct features, but still expose a few issues (e.g., data sparseness) which has the significant impact on the classification accuracy. Recently, with the development of deep learning in the various fields of artificial intelligence, neural models have been introduced into the text classification field due to their abilities of text representation learning. In this section, we will introduce the two typical tasks of text classification, including sentence classification and sentiment classification.

4.5.1.1 Sentence Classification

Sentence classification aims to assign a sentence an appropriate category, which is a basic task of the text classification application.

Considering the effectiveness of the CNN models in capturing sentence semantic meanings, [31] first proposes to utilize the CNN models trained on the top of pre-trained word embeddings to classify sentences, which achieved promising results on several sentence classification datasets. Then, [30] introduces a dynamic CNN model to model the semantic meanings of sentences. This model handles sentences of varying lengths and uses dynamic max-pooling over linear sequences, which could help the model capture both short-range and long-range semantic relations in sentences. Furthermore, [9] proposes a novel CNN-based model named as Very Deep CNN, which operates directly at the character level. It shows that those deeper models have better results on sentence classification and can capture the hierarchical information from scattered characters to whole sentences. Yin and Schütze [74] also propose MV-CNN, which utilizes multiple types of pretrained word embeddings and extracts features from multi-granular phrases with variable-sized convolutional layers. To address the drawbacks of MV-CNN such as model complexity and the requirement for the same dimension of embeddings, [80] proposes a novel model called MG-CNN to capture multiple features from multiple sets of embeddings that are concatenated at the penultimate layer. Zhang et al. [79] present RA-CNN to jointly exploit labels on documents and their constituent sentences, which can estimate the probability that a given sentence is informative and then scales the contribution of each sentence to aggregate a document representation in proportion to the estimates.

The RNN model which aims to capture the sequential information of sentences is also widely used in sentence classification. Lai et al. [32] propose a neural network for text classification, which applies a recurrent structure to capture contextual information. Moreover, [37] introduces a multitask learning framework based on the RNN to jointly learn across multiple sentence classification tasks, which employs

three different mechanisms of sharing information to model sentences with both task-specific and shared layers. Yang et al. [71] introduce word-level and sentence-level attention mechanisms into an RNN-based model as well as a hierarchical structure to capture the hierarchical information of documents for sentence classification.

4.5.1.2 Sentiment Classification

Sentiment classification is a special task of the sentence classification application, whose objective is to classify the sentimental polarities of opinions a piece of text contains, e.g., favorable or unfavorable, positive or negative. This task appeals the NLP community since it has lots of potential downstream applications such as movie review suggestions.

Similar to text classification, the sentence representation based on neural models has also been widely explored for sentiment classification. Glorot et al. [20] use a stacked denoising autoencoder in sentiment classification for the first time. Then, a series of recursive neural network models based on the recursive tree structure of sentences are conducted to learn sentence representations for sentiment classification, including the recursive autoencoder (RAE) [55], matrix-vector recursive neural network (MV-RNN) [54], and recursive neural tensor network (RNTN) [56]. Besides, [29] adopts a CNN to learn sentence representations and achieves promising performance in sentiment classification.

The RNN models also benefit sentiment classification as they are able to capture the sequential information. Li et al. [35] and Tai et al. [62] investigate a tree-structured LSTM model on text classification. There are also some hierarchical models proposed to deal with document-level sentiment classification [5, 63], which generate semantic representations at different levels (e.g., phrase, sentence, or document) within a document. Moreover, the attention mechanism is also introduced into sentiment classification, which aims to select important words from a sentence or important sentences from a document [71].

4.5.2 Relation Extraction

To enrich existing KGs, researchers have devoted many efforts to automatically finding novel relational facts in text. Therefore, relation extraction (RE), which aims at extracting relational facts according to semantic information in plain text, has become a crucial NLP application. As RE is also an important downstream application of sentence representation, we will, respectively, introduce the techniques and extensions to show how to utilize sentence representation for different RE scenarios. Considering neural networks have become the backbone of the recent NLP research, we mainly focus on Neural RE (NRE) models in this section.

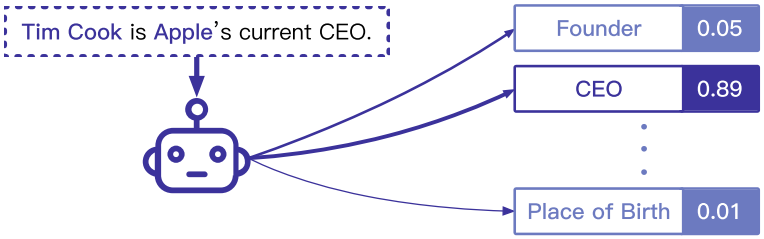

Fig. 4.6 An example of sentence-level relation extraction

4.5.2.1 Sentence-Level NRE

Sentence-level NRE aims at predicting the semantic relations between the given entity (or nominal) pair in a sentence. As shown in Fig. 4.6, given the input sentence s which consists of n words $s = \{w_1, w_2, \ldots, w_n\}$ and its corresponding entity pair e_1 and e_2 as input, sentence-level NRE wants to obtain the conditional probability $P(r|s, e_1, e_2)$ of relation r ($r \in \mathscr{R}$) via a neural network, which can be formalized as

$$P(r|s, e_1, e_2) = P(r|s, e_1, e_2, \theta), \tag{4.29}$$

where θ is all parameters of the neural network and r is a relation in the relation set $\mathscr{R}$.

A basic form of sentence-level NRE consists of three components: (a) an input encoder to give a representation for each input word, (b) a sentence encoder which computes either a single vector or a sequence of vectors to represent the original sentence, and (c) a relation classifier which calculates the conditional probability distribution of all relations.

Input Encoder. First, a sentence-level NRE system projects the discrete words of the source sentence into a continuous vector space, and obtains the input representation $\mathbf{w} = \{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m\}$ of the source sentence.

(1) Word Embeddings. Word embeddings aim to transform words into distributed representations to capture the syntactic and semantic meanings of the words. In the sentence s, every word w_i is represented by a real-valued vector. Word representations are encoded by column vectors in an embedding matrix $\mathbf{E} \in \mathbb{R}^{d^a \times |V|}$ where V is a fixed-sized vocabulary. Although word embeddings are the most common way to represent input words, there are also efforts made to utilize more complicated information of input sentences for RE.

(2) Position Embeddings. In RE, the words close to the target entities are usually informative to determine the relation between the entities. Therefore, position embeddings are used to help models keep track of how close each word is to the head or tail entities. It is defined as the combination of the relative distances from the current word to the head or tail entities. For example, in the sentence `Bill_Gates is the founder of Microsoft.`, the relative distance

from the word `founder` to the head entity `Bill_Gates` is −3 and the tail entity `Microsoft` is 2. Besides word position embeddings, more linguistic features are also considered in addition to the word embeddings to enrich the linguistic features of the input sentence.

(3) Part-of-speech (POS) Tag Embeddings. POS tag embeddings are to represent the lexical information of the target word in the sentence. Because word embeddings are obtained from a large-scale general corpus, the general information they contain may not be in accordance with the meaning in a specific sentence. Hence, it is necessary to align each word with its linguistic information considering its specific context, e.g., noun and verb. Formally, each word w_i is encoded by the corresponding column vector in an embedding matrix $\mathbf{E}^p \in \mathbb{R}^{d^p \times |V^p|}$, where d^p is the dimension of embedding vector and V^p indicates a fixed-sized POS tag vocabulary.

(4) WordNet Hypernym Embeddings. WordNet hypernym embeddings aim to take advantages of the prior knowledge of hypernym to help RE models. When given the hypernym information of each word in WordNet (e.g., noun.food and verb.motion), it is easier to build the connections between different but conceptually similar words. Formally, each word w_i is encoded by the corresponding column vector in an embedding matrix $\mathbf{E}^h \in \mathbb{R}^{d^h \times |V^h|}$, where d^h is the dimension of embedding vector and V^h indicates a fixed-sized hypernym vocabulary.

For each word, the NRE models often concatenate some of the above four feature embeddings as their input embeddings. Therefore, the feature embeddings of all words are concatenated and denoted as a final input sequence $\mathbf{w} = \{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m\}$, where $\mathbf{w}_i \in \mathbb{R}^d$, d is the total dimension of all feature embeddings concatenated for each word.

Sentence Encoder. The sentence encoder is the core for sentence representation, which encodes input representations into either a single vector or a sequence of vectors $\mathbf{x}$ to represent sentences. We will introduce the different sentence encoders in the following.

(1) Convolutional Neural Network Encoder. Zeng et al. [76] propose to encode input sentences using a CNN model, which extracts local features by a convolutional layer and combines all local features via a max-pooling operation to obtain a fixed-sized vector for the input sentence. Formally, a convolutional layer is defined as an operation on a vector sequence $\mathbf{w}$:

$$\mathbf{p} = \text{CNN}(\mathbf{w}), \tag{4.30}$$

where CNN indicates the convolution operation inside the convolutional layer.

And the ith element of the sentence vector $\mathbf{x}$ can be calculated as follows:

$$[\mathbf{x}]_i = f(\max(\mathbf{p}_i)), \tag{4.31}$$

where f is a nonlinear function applied at the output, such as the hyperbolic tangent function.

Further, PCNN [75], which is a variation of CNN, adopts a piece-wise max-pooling operation. All hidden vectors $\{\mathbf{p}_1, \mathbf{p}_2, \ldots\}$ are divided into three segments by the head and tail entities. The max-pooling operation is performed over the three segments separately, and the $\mathbf{x}$ is the concatenation of the pooling results over the three segments.

(2) Recurrent Neural Network Encoder. Zhang and Wang [78] propose to embed input sentences using an RNN model which can learn the temporal features. Formally, each input word representation is put into recurrent layers step by step. For each step i, the network takes the ith word representation vector $\mathbf{w}_i$ and the output of the previous $i-1$ steps $\mathbf{h}_{i-1}$ as input:

$$\mathbf{h}_i = \text{RNN}(\mathbf{w}_i, \mathbf{h}_{i-1}), \tag{4.32}$$

where RNN indicates the transform function inside the RNN cell, which can be the LSTM units or the GRU units mentioned before.

The conventional RNN models typically deal with text sequences from start to end, and build the hidden state of each word only considering its preceding words. It has been verified that the hidden state considering its following words is more effective. Hence, the bi-directional RNN (BRNN) [52] is adopted to learn hidden states using both preceding and following words.

Similar to the previous CNN models in RE, the RNN model combines the output vectors of the recurrent layer as local features, and then uses a max-pooling operation to extract the global feature, which forms the representation of the whole input sentence. The max-pooling layer could be formulated as

$$[\mathbf{x}]_j = \max_i [\mathbf{h}_i]_j. \tag{4.33}$$

Besides max-pooling, word attention can also combine all local feature vectors together. The attention mechanism [1] learns attention weights on each step. Supposing $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_m]$ is the matrix consisting of all output vectors produced by the recurrent layer, the feature vector of the whole sentence $\mathbf{x}$ is formed by a weighted sum of these output vectors:

$$\alpha = \text{Softmax}(\mathbf{s}^\top \tanh(\mathbf{H})), \tag{4.34}$$

$$\mathbf{x} = \mathbf{H}\alpha^\top, \tag{4.35}$$

where $\mathbf{s}$ is a trainable query vector.

Besides, [42] proposes a model that captures information from both word sequence and tree-structured dependency by stacking bidirectional path-based LSTM-RNNs (i.e., bottom-up and top-down). More specifically, it focuses on the shortest path between the two target entities in the dependency tree, and utilizes the stacked layers to encode the shortest path for the whole sentence representation. In fact, some preliminary work [69] has shown that these paths are useful in RE, and various

recursive neural models are also proposed for this. Next, we will introduce these recursive models in detail.

(3) Recursive Neural Network Encoder. The recursive encoder aims to extract features from the information of syntactic parsing trees, considering the syntactic information is beneficial for extracting relations from sentences. Generally, these encoders treat the tree structure inside syntactic parsing trees as a strategy of composition as well as a direction to combine each word feature.

Socher et al. [54] propose a recursive matrix-vector model (MV-RNN) which captures the structure information by assigning a matrix-vector representation for each constituent of the constituents in parsing trees. The vector captures the meaning of the constituent itself and the matrix represents how it modifies the meaning of the word it combines with. Tai et al. [62] further propose two types of tree-structured LSTMs including the Child-Sum Tree-LSTM and the N-ary Tree-LSTM to capture tree structure information. For the Child-Sum Tree-LSTM, given a tree, let $C(t)$ denote the set of children of node t. Its transition equations are defined as follows:

$$\hat{\mathbf{h}}_t = \sum_{k \in C(t)} \text{TLSTM}(\mathbf{h}_k), \tag{4.36}$$

where $\text{TLSTM}(\cdot)$ indicates a Tree-LSTM cell, which is simply modified from LSTM cell. The N-ary Tree-LSTM has similar transition equations as the Child-Sum Tree-LSTM. The only difference is that it limits the tree structures to have at most N branches.

Relation Classifier. When obtaining the representation $\mathbf{x}$ of the input sentence, relation classifier calculates the conditional probability $P(r|x, e_1, e_2)$ via a softmax layer as follows:

$$P(r|x, e_1, e_2) = \text{Softmax}(\mathbf{M}\mathbf{x} + \mathbf{b}), \tag{4.37}$$

where $\mathbf{M}$ indicates the relation matrix and $\mathbf{b}$ is a bias vector.

4.5.2.2 Bag-Level NRE

Although existing neural models have achieved great success for extracting novel relational facts, it always suffers the lack of training data. To address this issue, researchers proposed a distant supervision assumption to generate training data via aligning KGs and plain text automatically. The intuition of distant supervision assumption is that all sentences that contain two entities will express their relations in KGs. For example, (`New York`, `city_of`, `United States`) is a relational fact in a KG, distant supervision assumption will regard all sentences that contain these two entities as positive instances for the relation `city_of`. It offers a natural way of utilizing information from multiple sentences (bag-level) rather than a single sentence (sentence-level) to decide if a relation holds between two entities.

Therefore, bag-level NRE aims to predict the semantic relations between an entity pair using all involved sentences. As shown in Fig. 4.7, given the input sentence set

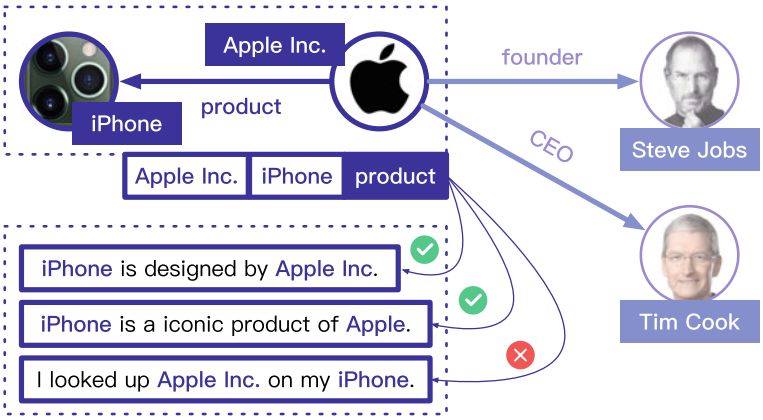

Fig. 4.7 An example of bag-level relation extraction

S which consists of n sentences $S = \{s_1, s_2, \ldots, s_n\}$ and its corresponding entity pair e_1 and e_2 as inputs, bag-level NRE wants to obtain the conditional probability $P(r|S, e_1, e_2)$ of relation r ($r \in \mathbb{R}$) via a neural network, which can be formalized as

$$P(r|S, e_1, e_2) = P(r|S, e_1, e_2, \theta). \tag{4.38}$$

A basic form of bag-level NRE consists of four components: (a) an input encoder similar to sentence-level NRE, (b) a sentence encoder similar to sentence-level NRE, (c) a bag encoder which computes a vector representing all related sentences in a bag, and (d) a relation classifier similar to sentence-level NRE which takes bag vectors as input instead of sentence vectors. As the input encoder, sentence encoder, and relation classifier of bag-level NRE are similar to the ones of sentence-level NRE, we will thus mainly focus on introducing the bag encoder in detail.

Bag Encoder. The bag encoder encodes all sentence vectors into a single vector $\mathbf{S}$. We will introduce the different bag encoders in the following:

(1) Random Encoder. It simply assumes that each sentence can express the relation between two target entities and randomly select one sentence to represent the bag. Formally, the bag representation is defined as

$$\mathbf{S} = \mathbf{s}_i \ (i \in \{1, 2, \ldots, n\}), \tag{4.39}$$

where $\mathbf{s}_i$ indicates the sentence representation of $s_i \in S$ and i is a random index.

(2) Max Encoder. As introduced above, not all sentences containing two target entities can express their relations. For example, the sentence `New York City is the premier gateway for legal immigration to the United States` does not express the relation `city of`. Hence, in [75], they follow the at-least-one assumption which assumes that at least one sentence that contains these two target entities can express their relations, and select the sentence

with the highest probability for the relation to represent the bag. Formally, bag representation is defined as

$$\mathbf{S} = \mathbf{s}_i \ (i = \arg\max_i P(r|s_i, e_1, e_2)). \tag{4.40}$$

(3) Average Encoder. Both random encoder or max encoder use only one sentence to represent the bag, which ignores the rich information of different sentences. To exploit the information of all sentences, [36] believes that the representation $\mathbf{S}$ of the bag depends on all sentences' representations. Each sentence representation $\mathbf{s}_i$ can give the relation information about two entities to a certain extent. The average encoder assumes that all sentences contribute equally to the representation of the bag. It means the embedding $\mathbf{S}$ of the bag is the average of all the sentence vectors:

$$\mathbf{S} = \sum_i \frac{1}{n}\mathbf{s}_i. \tag{4.41}$$

(4) Attentive Encoder. Due to the wrong label issue brought by distant supervision assumption inevitably, the performance of average encoder will be influenced by those sentences that contain no relation information. To address this issue, [36] further proposes to employ a selective attention to reduce those noisy sentences. Formally, the bag representation is defined as a weighted sum of sentence vectors:

$$\mathbf{S} = \sum_i \alpha_i \mathbf{s}_i, \tag{4.42}$$

where α_i is defined as

$$\alpha_i = \frac{\exp(\mathbf{s}_i^\top \mathbf{A}\mathbf{r})}{\sum_j \exp(\mathbf{x}_j^\top \mathbf{A}\mathbf{r})}, \tag{4.43}$$

where $\mathbf{A}$ is a diagonal matrix and $\mathbf{r}$ is the representation vector of relation r.

Relation Classifier. Similar to sentence-level NRE, when obtaining the bag representation $\mathbf{S}$, relation classifier also calculates the conditional probability $P(r|S, e_1, e_2)$ via a softmax layer as follows:

$$P(r|S, e_1, e_2) = \text{Softmax}(\mathbf{MS} + \mathbf{b}), \tag{4.44}$$

where $\mathbf{M}$ indicates the relation matrix and $\mathbf{b}$ is a bias vector.

4.5.2.3 Extensions

Recently, NRE systems have achieved significant improvements in both, the supervised and distantly supervised scenarios. However, there are still many challenges in the task of RE, and many researchers have been focusing on other aspects to improve

the performance of NRE as well. In this section, we will introduce these extensions in detail.

Utilization of External Information. Most existing NRE systems stated above only concentrate on the sentences which are extracted, regardless of the rich external information such as KGs. This heterogeneous information could provide additional knowledge from KG and is essential when extracting new relational facts.

Han et al. [24] propose a novel joint representation learning framework for knowledge acquisition. The key idea is that the joint model learns knowledge and text representations within a unified semantic space via KG-text alignments. For the text part, the sentence with two entities `Mark Twain` and `Florida` is regarded as the input for a CNN encoder, and the output of CNN is considered to be the latent relation `PlaceOfBirth` of this sentence. For the KG part, entity and relation representations are learned via translation-based methods. The learned representations of KG and text parts are aligned during training. Besides this preliminary attempt, many efforts have been devoted to this direction [25, 28, 51, 67, 68].

Incorporating Relational Paths. Although existing NRE systems have achieved promising results, they still suffer a major problem: the models can only directly learn from those sentences which contain both two-target entities. However, those sentences containing only one of the entities could also provide useful information and help build inference chains. For example, if we know that "A is the son of B" and "B is the son of C", we can infer that A is the grandson of C.

To utilize the information of both direct and indirect sentences, [77] introduces a path-based NRE model that incorporates textual relational paths. The model first employs a CNN encoder to embed the semantic meanings of sentences. Then, the model builds a relation path encoder, which measures the probability of relations given an inference chain in the text. Finally, the model combines information from both direct sentences and relational paths, and then predicts the confidence of each relationship. This work is the first effort to consider the knowledge of relation path in text for NRE, and there are also several methods later to consider the reasoning path of sentence semantic meanings for RE [11, 19].

Document-level Relation Extraction. In fact, not all relational facts can be extracted by sentence-level RE, i.e., a large number of relational facts are expressed in multiple sentences. Taking Fig. 4.9 as an example, multiple entities are mentioned in the document and exhibit complex interactions. In order to identify the relational fact (`Riddarhuset`, `country`, `Sweden`), one has to first identify the fact that `Riddarhuset` is located in `Stockholm` from Sentence 4, then identify the facts `Stockholm` is the capital of `Sweden` and `Sweden` is a country from Sentence 1. With the above facts, we can finally infer that the sovereign state of `Riddarhuset` is `Sweden`. This process requires reading and reasoning over multiple sentences in a document, which is intuitively beyond the reach of sentence-level RE methods. According to the statistics on a human-annotated corpus sampled from Wikipedia documents [72], at least 40.7% relational facts can only be extracted from multiple sentences, which is not negligible. Swampillai and Stevenson [61] and Verga et al. [66] also report similar observations. Therefore, it is necessary to move RE

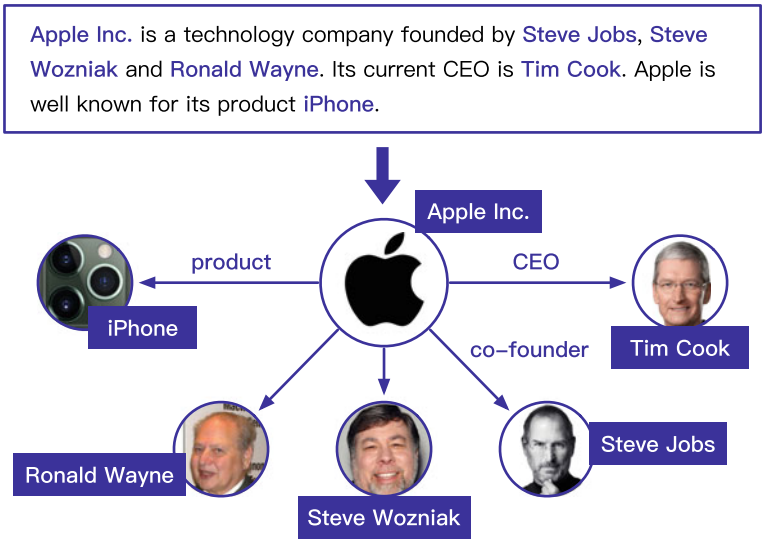

Fig. 4.8 An example of document-level relation extraction

forward from the sentence level to the document level. Figure 4.8 is an example for document-level RE.

Kungliga Hovkapellet

[1] ***Kungliga Hovkapellet*** (The ***Royal Court Orchestra***) is a *Swedish* orchestra, originally part of the *Royal Court* in ***Sweden***'s capital *Stockholm*. [2] The orchestra originally consisted of both musicians and singers. [3] It had only male members until *1727*, when *Sophia Schröder* and *Judith Fischer* were employed as vocalists; in the *1850s*, the harpist *Marie Pauline Åhman* became the first female instrumentalist. [4] From *1731*, public concerts were performed at ***Riddarhuset*** in *Stockholm*. [5] Since *1773*, when the ***Royal Swedish Opera*** was founded by *Gustav III* of ***Sweden***, the ***Kungliga Hovkapellet*** has been part of the opera's company.

Subject: ***Kungliga Hovkapellet; Royal Court Orchestra***
Object: ***Royal Swedish Opera***
Relation: `part_of` **Supporting Evidence:** 5

Subject: ***Riddarhuset***
Object: ***Sweden***
Relation: `country` **Supporting Evidence:** 1, 4

Fig. 4.9 An example from DocRED [72]

However, existing datasets for document-level RE either only have a small number of manually annotated relations and entities [34], or exhibit noisy annotations from distant supervision [45, 49], or serve specific domains or approaches [33]. To address this issue, [72] constructs a large-scale, manually annotated, and general-purpose document-level RE dataset, named as DocRED. DocRED is constructed from Wikipedia and Wikidata, and has two key features. First, DocRED contains 132, 375 entities and 56, 354 relational facts annotated on 5, 053 Wikipedia documents, which is the largest human-annotated document-level RE dataset now. Second, over 40% of the relational facts in DocRED can only be extracted from multiple sentences. This makes DocRED require reading multiple sentences in a document to recognize entities and inferring their relations by synthesizing all information of the document.

The experimental results on DocRED show that the performance of existing sentence-level RE methods declines significantly on DocRED, indicating the task document-level RE is more challenging than sentence-level RE and remains an open problem. It also relates to the document representation which will be introduced in the next chapter.

Few-shot Relation Extraction.

As we mentioned before, the performance of the conventional RE models [23, 76] heavily depend on time-consuming and labor-intensive annotated data, which make themselves hard to generalize well. Although adopting distant supervision is a primary approach to alleviate this problem, the distantly supervised data also exhibits a long-tail distribution, where most relations have very limited instances. Furthermore, distant supervision suffers the wrong labeling problem, which makes it harder to classify long-tail relations. Hence, it is necessary to study training RE models with insufficient training instances. Figure 4.10 is an example for few-shot RE.

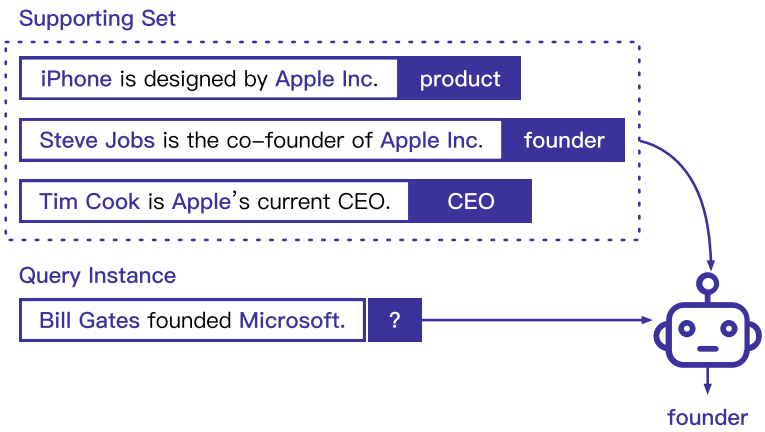

Fig. 4.10 An example of few-shot relation extraction

Table 4.1 An example for a 3 way 2 shot scenario. Different colors indicate different entities, underline for head entity, and *emphasize* for tail entity

Supporting set	
(A) capital_of	(1) London is the capital of *the U.K* (2) Washington is the capital of *the U.S.A*
(B) member_of	(1) Newton served as the president of *the Royal Society* (2) Leibniz was a member of *the Prussian Academy of Sciences*
(C) birth_name	(1) *Samuel Langhorne Clemens*, better known by his pen name Mark Twain, was an American writer (2) Alexei Maximovich Peshkov, primarily known as *Maxim Gorky*, was a Russian and Soviet writer
Test instance	
(A) or (B) or (C)	Euler was elected a foreign member of *the Royal Swedish Academy of Sciences*

FewRel [26] is a new large-scale supervised few-shot RE dataset, which requires models capable of handling classification task with a handful of training instances, as shown in Table 4.1. Benefiting from the FewRel dataset, there are some efforts to exploring few-shot RE [17, 53, 73] and achieve promising results. Yet, few-shot RE still remains a challenging problem for further research [18].

4.6 Summary

In this chapter, we introduce sentence representation learning. Sentence representation encodes the semantic information of a sentence into a real-valued representation vector, and can be utilized in further sentence classification or matching tasks. First, we introduce the one-hot representation for sentences and probabilistic language models. Secondly, we extensively introduce several neural language models, including adopting the feedforward neural networks, the convolutional neural networks, the recurrent neural networks, and the Transformer for language models. These neural models can learn rich linguistic and semantic knowledge from language modeling. Benefiting from this, the pre-trained language models trained with large-scale corpora have achieved state-of-the-art performance on various downstream NLP tasks by transferring the learned semantic knowledge from general corpora to the target tasks. Finally, we introduce several typical applications of sentence representation including text classification and relation extraction.

For further understanding of sentence representation learning and its applications, there are also some recommended surveys and books including

- Yoav, Neural network methods for natural language processing [21].
- Deng & Liu, Deep learning in natural language processing [13].

In the future, for better sentence representation, some directions are requiring further efforts:

(1) **Exploring Advanced Architectures.** The improvement of model architectures is the key factor in the success of sentence representation. From the feedforward neural networks to the Transformer, people are designing more suitable neural models for sequential inputs. Based on the Transformer, some researchers are working on new NLP architectures. For instance, Transformer-XL [10] is proposed to solve the problem of fixed-length context in the Transformer. Since the Transformer is the state-of-the-art NLP architecture, current works mainly adopt attention mechanisms. Beyond these works, is it possible to introduce more human cognitive mechanisms to neural models?

(2) **Modeling Long Documents.** The representation of long documents is an important extension of sentence representation. There are some new challenges during modeling long documents, such as discourse analysis and co-reference resolution. Although some existing works already provide document-level NLP tasks (e.g., DocRED [72]), the model performance on these tasks is still much lower than the human performance. We will also introduce the advances in document representation learning in the following chapter.

(3) **Performing Efficient Representation.** Although the combination of Transformer and large-scale data leads to very powerful sentence representation, these representation models require expensive computational cost, which limits the applications in downstream tasks. Some existing works explore to use model compression techniques for more efficient models. These techniques include knowledge distillation [60], parameter pruning [16], etc. Beyond these works, there remain lots of unsolved problems for developing better representation models, which can efficiently learn from large-scale data and provide effective vectors in downstream tasks.

References

1. Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*, 2015.
2. Yoshua Bengio. Neural net language models. *Scholarpedia*, 3(1):3881, 2008.
3. Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3(Feb):1137–1155, 2003.
4. Yoshua Bengio, Jean-Sébastien Senécal, et al. Quick training of probabilistic neural nets by importance sampling. In *Proceedings of AISTATS*, 2003.

5. Parminder Bhatia, Yangfeng Ji, and Jacob Eisenstein. Better document-level sentiment analysis from rst discourse parsing. In *Proceedings of EMNLP*, 2015.
6. David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.
7. 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.
8. Alexis Conneau and Guillaume Lample. Cross-lingual language model pretraining. In *Proceedings of NeurIPS*, 2019.
9. Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. Very deep convolutional networks for text classification. In *Proceedings of EACL*, volume 1, 2017.
10. Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of ACL*, page 2978–2988, 2019.
11. Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. Chains of reasoning over entities, relations, and text using recurrent neural networks. In *Proceedings of EACL*, pages 132–141, 2017.
12. Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *Proceedings of ICML*, 2017.
13. Li Deng and Yang Liu. *Deep learning in natural language processing*. Springer, 2018.
14. 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.
15. Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. Unified language model pre-training for natural language understanding and generation. In *Proceedings of NeurIPS*, 2019.
16. Angela Fan, Edouard Grave, and Armand Joulin. Reducing transformer depth on demand with structured dropout. In *Proceedings of ICLR*, 2020.
17. Tianyu Gao, Xu Han, Zhiyuan Liu, and Maosong Sun. Hybrid attention-based prototypical networks for noisy few-shot relation classification. In *Proceedings of AAAI*, pages 6407–6414, 2019.
18. Tianyu Gao, Xu Han, Hao Zhu, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. FewRel 2.0: Towards more challenging few-shot relation classification. In *Proceedings of EMNLP-IJCNLP*, pages 6251–6256, 2019.
19. Michael Glass, Alfio Gliozzo, Oktie Hassanzadeh, Nandana Mihindukulasooriya, and Gaetano Rossiello. Inducing implicit relations from text using distantly supervised deep nets. In *International Semantic Web Conference*, pages 38–55. Springer, 2018.
20. Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of ICML*, 2011.
21. Yoav Goldberg. Neural network methods for natural language processing. *Synthesis Lectures on Human Language Technologies*, 10(1):1–309, 2017.
22. Joshua Goodman. Classes for fast maximum entropy training. In *Proceedings of ASSP*, 2001.
23. Matthew R Gormley, Mo Yu, and Mark Dredze. Improved relation extraction with feature-rich compositional embedding models. In *Proceedings of EMNLP*, 2015.
24. Xu Han, Zhiyuan Liu, and Maosong Sun. Joint representation learning of text and knowledge for knowledge graph completion. *arXiv preprint* arXiv:1611.04125, 2016.
25. Xu Han, Zhiyuan Liu, and Maosong Sun. Neural knowledge acquisition via mutual attention between knowledge graph and text. In *Proceedings of AAAI*, pages 4832–4839, 2018.
26. Xu Han, Hao Zhu, Pengfei Yu, Ziyun Wang, Yuan Yao, Zhiyuan Liu, and Maosong Sun. FewRel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. In *Proceedings of EMNLP*, 2018.
27. Zhiheng Huang, Geoffrey Zweig, and Benoit Dumoulin. Cache based recurrent neural network language model inference for first pass speech recognition. In *Proceedings of ICASSP*, 2014.
28. Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. Distant supervision for relation extraction with sentence-level attention and entity descriptions. In *Proceedings of AAAI*, pages 3060–3066, 2017.

29. Rie Johnson and Tong Zhang. Effective use of word order for text categorization with convolutional neural networks. In *Proceedings of ACL-HLT*, 2015.
30. Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. In *Proceedings of ACL*, 2014.
31. Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of EMNLP*, 2014.
32. Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. Recurrent convolutional neural networks for text classification. In *Proceedings of AAAI*, 2015.
33. Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. Zero-shot relation extraction via reading comprehension. In *Proceedings of CoNLL*, 2017.
34. Jiao Li, Yueping Sun, Robin J. Johnson, Daniela Sciaky, Chih-Hsuan Wei, Robert Leaman, Allan Peter Davis, Carolyn J. Mattingly, Thomas C. Wiegers, and Zhiyong Lu. BioCreative V CDR task corpus: a resource for chemical disease relation extraction. *Database*, pages 1–10, 2016.
35. Jiwei Li, Minh-Thang Luong, Dan Jurafsky, and Eduard Hovy. When are tree structures necessary for deep learning of representations? In *Proceedings of EMNLP*, 2015.
36. Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. Neural relation extraction with selective attention over instances. In *Proceedings of ACL*, 2016.
37. Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Recurrent neural network for text classification with multi-task learning. In *Proceedings of IJCAI*, 2016.
38. Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint* arXiv:1907.11692, 2019.
39. Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *Proceedings of NeurIPS*, 2019.
40. Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Proceedings of ICLR*, 2013.
41. Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Proceedings of InterSpeech*, 2010.
42. Makoto Miwa and Mohit Bansal. End-to-end relation extraction using lstms on sequences and tree structures. In *Proceedings of ACL*, 2016.
43. Andriy Mnih and Yee Whye Teh. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of ICML*, 2012.
44. Frederic Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In *Proceedings of Aistats*, 2005.
45. Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen-tau Yih. Cross-sentence n-ary relation extraction with graph LSTMs. *Transactions of the Association for Computational Linguistics*, 5:101–115, 2017.
46. Matthew E Peters, Mark Neumann, Robert Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A Smith. Knowledge enhanced contextual word representations. In *Proceedings of EMNLP-IJCNLP*, 2019.
47. Ngoc-Quan Pham, German Kruszewski, and Gemma Boleda. Convolutional neural network language models. In *Proceedings of EMNLP*, 2016.
48. Matt Post and Shane Bergsma. Explicit and implicit syntactic features for text classification. In *Proceedings of ACL*, 2013.
49. Chris Quirk and Hoifung Poon. Distant supervision for relation extraction beyond the sentence boundary. In *Proceedings of EACL*, 2017.
50. Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *URL* https://s3-us-west-2.amazonaws.com/openai-assets/researchcovers/languageunsupervised/languageunderstandingpaper.pdf, 2018.
51. Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. Relation extraction with matrix factorization and universal schemas. In *Proceedings of NAACL-HLT*, pages 74–84, 2013.

52. Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.
53. Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. Matching the Blanks: Distributional similarity for relation learning. In *Proceedings of ACL*, pages 2895–2905, 2019.
54. Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of EMNLP*, 2012.
55. Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of EMNLP*, 2011.
56. Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*, 2013.
57. Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mass: Masked sequence to sequence pre-training for language generation. In *Proceedings of ICML*, 2019.
58. Daniel Soutner, Zdeněk Loose, Luděk Müller, and Aleš Pražák. Neural network language model with cache. In *Proceedings of ICTSD*, 2012.
59. Chen Sun, Austin Myers, Carl Vondrick, Kevin Murphy, and Cordelia Schmid. Videobert: A joint model for video and language representation learning. In *Proceedings of ICCV*, 2019.
60. Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. Patient knowledge distillation for bert model compression. In *Proceedings of EMNLP-IJCNLP*, page 4314–4323, 2019.
61. Kumutha Swampillai and Mark Stevenson. Inter-sentential relations in information extraction corpora. In *Proceedings of LREC*, 2010.
62. Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of ACL*, 2015.
63. Duyu Tang, Bing Qin, and Ting Liu. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of EMNLP*, 2015.
64. Wilson L Taylor. “cloze procedure": A new tool for measuring readability. *Journalism Bulletin*, 30(4):415–433, 1953.
65. Ashish Vaswani, Noam Shazeer, Niki Parmar, Llion Jones, Jakob Uszkoreit, Aidan N Gomez, and Lukasz Kaiser. Attention is all you need. In *Proceedings of NeurIPS*, 2017.
66. Patrick Verga, Emma Strubell, and Andrew McCallum. Simultaneously self-attending to all mentions for full-abstract biological relation extraction. In *Proceedings of NAACL-HLT*, 2018.
67. Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph and text jointly embedding. In *Proceedings of EMNLP*, pages 1591–1601, 2014.
68. Zhigang Wang and Juan-Zi Li. Text-enhanced representation learning for knowledge graph. In *Proceedings of IJCAI*, pages 1293–1299, 2016.
69. Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. Semantic relation classification via convolutional neural networks with simple negative sampling. In *Proceedings of EMNLP*, 2015.
70. Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Proceedings of NeurIPS*, 2019.
71. Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of NAACL*, 2016.
72. Yuan Yao, Deming Ye, Peng Li, Xu Han, Yankai Lin, Zhenghao Liu, Zhiyuan Liu, Lixin Huang, Jie Zhou, and Maosong Sun. DocRED: A large-scale document-level relation extraction dataset. In *Proceedings of ACL*, 2019.
73. Zhi-Xiu Ye and Zhen-Hua Ling. Multi-level matching and aggregation network for few-shot relation classification. In *Proceedings of ACL*, pages 2872–2881, 2019.
74. Wenpeng Yin and Hinrich Schütze. Multichannel variable-size convolution for sentence classification. In *Proceedings of CoNLL*, 2015.
75. Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of EMNLP*, 2015.

76. Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. Relation classification via convolutional deep neural network. In *Proceedings of COLING*, 2014.
77. Wenyuan Zeng, Yankai Lin, Zhiyuan Liu, and Maosong Sun. Incorporating relation paths in neural relation extraction. In *Proceedings of EMNLP*, 2017.
78. Dongxu Zhang and Dong Wang. Relation classification via recurrent neural network. *arXiv preprint* arXiv:1508.01006, 2015.
79. Ye Zhang, Iain Marshall, and Byron C Wallace. Rationale-augmented convolutional neural networks for text classification. In *Proceedings of EMNLP*, 2016.
80. Ye Zhang, Stephen Roller, and Byron C Wallace. Mgnc-cnn: A simple approach to exploiting multiple word embeddings for sentence classification. In *Proceedings of NAACL*, 2016.
81. Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. Ernie: Enhanced language representation with informative entities. In *Proceedings of ACL*, 2019.