Chapter 9
Knowledge Representation Learning and Knowledge-Guided NLP

Xu Han, Weize Chen, Zhiyuan Liu, Yankai Lin, and Maosong Sun

Abstract Knowledge is an important characteristic of human intelligence and reflects the complexity of human languages. To this end, many efforts have been devoted to organizing various human knowledge to improve the ability of machines in language understanding, such as world knowledge, linguistic knowledge, commonsense knowledge, and domain knowledge. Starting from this chapter, our view turns to representing rich human knowledge and using knowledge representations to improve NLP models. In this chapter, taking world knowledge as an example, we present a general framework of organizing and utilizing knowledge, including knowledge representation learning, knowledge-guided NLP, and knowledge acquisition. For linguistic knowledge, commonsense knowledge, and domain knowledge, we will introduce them in detail in subsequent chapters considering their unique knowledge properties.

9.1 Introduction

The discussion of knowledge is far earlier than the exploration of NLP and is highly related to the study of human languages, which can be traced back to Plato in the Classical Period of ancient Greece [15]. Over the next thousand years, the discussion of knowledge gradually leads to many systematic philosophical theories, such as epistemology [146] and ontology [140], reflecting the long-term analysis and exploration of human intelligence.

X. Han · W. Chen · Z. Liu (✉) · M. Sun
Department of Computer Science and Technology, Tsinghua University, Beijing, China
e-mail: hanxu2022@tsinghua.edu.cn; chenwz21@mails.tsinghua.edu.cn; liuzy@tsinghua.edu.cn; sms@tsinghua.edu.cn

Y. Lin
Gaoling School of Artificial Intelligence, Renmin University of China, Beijing, China
e-mail: yankailin@ruc.edu.cn

Z. Liu et al. (eds.), *Representation Learning for Natural Language Processing*,
https://doi.org/10.1007/978-981-99-1600-9_9

Although the question *what is knowledge* is a controversial philosophical question with no definitive and generally accepted answer, we deeply touch on the concept of *knowledge* in every aspect of our lives. At the beginning of the twentieth century, analytic philosophy, which is advocated by Gottlob Frege, Bertrand Russell, and Ludwig Wittgenstein [131], inspires the establishment of symbolic systems to formalize human knowledge [68], significantly contributing to the later development of mathematical logic and philosophy of language.

As NLP is associated with human intelligence, the basic theory of NLP is also highly related to the above knowledge-related theories. As shown in Fig. 9.1, since the Dartmouth Summer Research Project on AI in 1956 [104], knowledge has played a significant role in the development history of NLP. Influenced by mathematical logic and linguistics, early NLP studies [2, 24, 25, 64] mainly focus on exploring symbolic knowledge representations and using symbolic systems to enable machines to understand and reason languages.

Due to the generalization and coverage problems of symbolic representations, ever since the 1990s, data-driven methods [63] are widely applied to represent human knowledge in a distributed manner. Moreover, after 2010, with the boom of deep learning [83], distributed knowledge representations are increasingly expressive from shallow to deep, providing a powerful tool of leveraging knowledge to understand complex semantics.

Making full use of knowledge is crucial to achieving better language understanding. To this end, we describe the general framework of organizing and utilizing knowledge, including knowledge representation learning, knowledge-guided NLP, and knowledge acquisition. With this knowledgeable NLP framework, we show how knowledge can be represented and learned to improve the performance of NLP models and how to acquire rich knowledge from text. As shown in Fig. 9.2, knowledge representation learning aims to encode symbolic knowledge into distributed representations so that knowledge can be more accessible to machines. Then, knowledge-guided NLP is explored to leverage knowledge representations to improve NLP models. Finally, based on knowledge-guided models, we can perform knowledge acquisition to extract more knowledge from plain text to enrich existing knowledge systems.

In the real world, people organize many kinds of knowledge, such as world knowledge, linguistic knowledge, commonsense knowledge, and domain knowledge. In this chapter, we focus on introducing the knowledgeable framework from the perspective of world knowledge since world knowledge is well-defined and general enough. Then, in the following chapters, we will show more details about other kinds of knowledge.

In Sect. 9.2, we will briefly introduce the important properties of symbolic knowledge and distributed model knowledge, aiming to indicate the core motivation for transforming symbolic knowledge into model knowledge. In Sects. 9.3 and 9.4, we will present typical approaches to encoding symbolic knowledge into distributed representations and show how to use knowledge representations to improve NLP

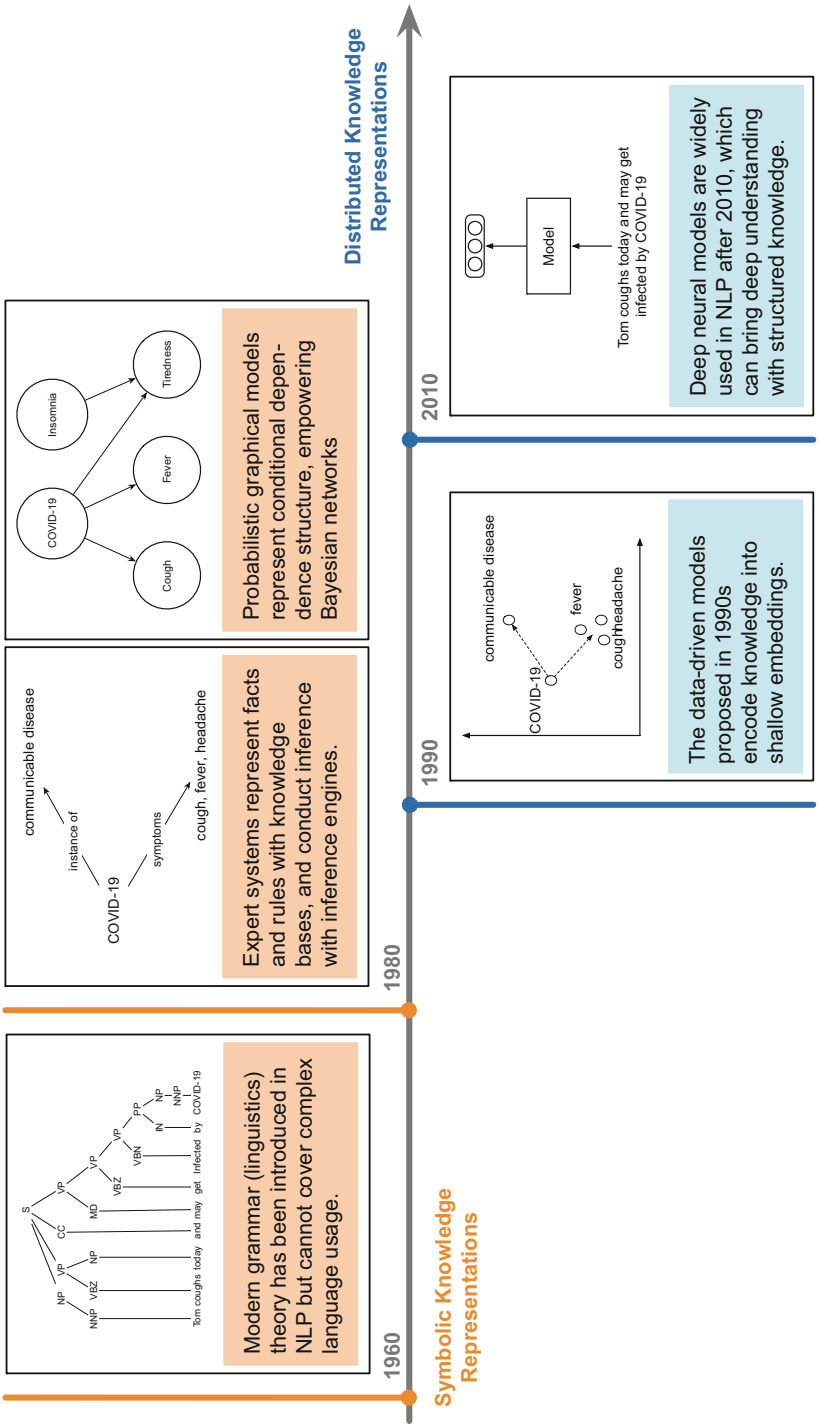

Fig. 9.1 Typical ways to organize and utilize knowledge in the development of NLP

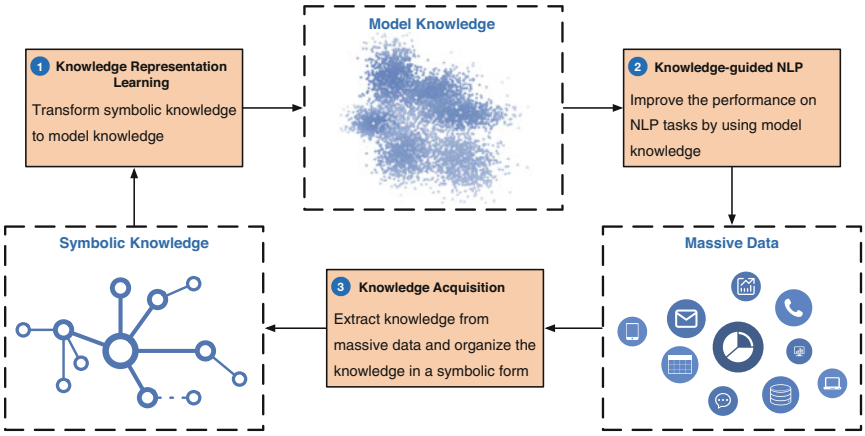

Fig. 9.2 The framework for organizing and utilizing knowledge for NLP tasks includes knowledge representation learning, knowledge-guided NLP, and knowledge acquisition

models, respectively. In Sect. 9.5, we will detail several scenarios for acquiring knowledge to ensure that we can acquire sufficient knowledge to help various NLP models.

9.2 Symbolic Knowledge and Model Knowledge

Before detailing the framework of knowledge representation learning, knowledge-guided NLP, and knowledge acquisition, we briefly present the necessary background information, especially various effective systems to organize knowledge. In this section, we will first introduce typical symbolic knowledge systems, which are the common way of organizing knowledge. Then, we will present more details of model knowledge obtained by projecting knowledge into machine learning models via distributed representation learning, providing a more machine-friendly way of organizing knowledge. Finally, we will show the recent trend in fusing symbolic knowledge and model knowledge, and indicate the importance of knowledge representation learning as well as knowledge-guided NLP in this fusion trend.

9.2.1 Symbolic Knowledge

During the two decades from the 1950s to 1970s, the efforts orienting NLP are mainly committed to symbolic computation systems. In 1956, Allen Newell and

Herbert A. Simon write the first AI program *Logic Theorist* that can perform automated reasoning [113]. The Logic Theorist can prove 38 of the first 52 theorems given by Bertrand Russell in *Principia Mathematica*, by using logic and heuristic rules to prune the search tree of reasoning. Over the same period, Noam Chomsky proposes syntactic structures [24] and transformational grammars [25], using formal languages with precise mathematical notations to drive machine processing of natural languages. Inspired by the Logic Theorist and syntactic structures, Herbert A. Simon and John McCarthy develop information processing language (IPL) [114] and list processing (LISP) [105], respectively, and these two programming languages significantly support computer programming for machine intelligence.

Since neither logic nor grammar rules can well solve complex and diverse problems in practical scenarios, the direction of *deriving general intelligence from symbolic systems* held by early researchers has gradually fallen into a bottleneck. After the 1970s, researchers turn to designing domain-specific intelligence systems for each specific application. The representative work of this period is the expert system [2] initiated by Edward Feigenbaum. An expert system generally consists of a knowledge base (KB) and an inference engine. KBs store a wealth of human knowledge, including domain-specific expertise and rules established by experts in various fields. Inference engines can leverage expertise and rules in KBs to solve specific problems.

Compared with the early AI methods entirely based on mathematical systems, expert systems work well in some practical fields such as business and medicine. Edward Feigenbaum further proposes knowledge engineering [42] in the 1980s, indicating the importance of knowledge acquisition, knowledge representation, and knowledge application to machine intelligence. As shown in Fig. 9.3, inspired by knowledge engineering, various KBs have emerged, such as the commonsense base Cyc [84] and Semantic Web [7]. The most notable achievement of expert systems is the Watson system developed by IBM. IBM Watson beats two human contestants on the quiz show Jeopardy, demonstrating the potential effectiveness of a KB with rich knowledge.

With the Internet thriving in the twenty-first century, massive messages have flooded into the World Wide Web, and knowledge is transferred to the semi-structured textual information on the Web. However, due to the information explosion, extracting the knowledge we want from the significant but noisy plain text on the Internet is not easy. During seeking effective ways to organize knowl-

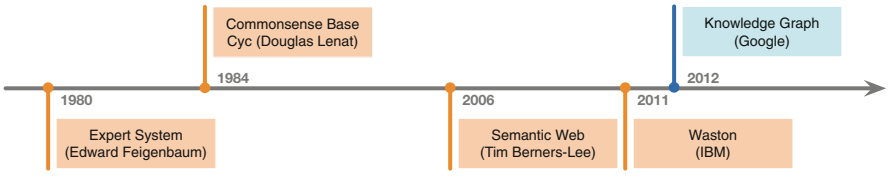

Fig. 9.3 The development of symbolic knowledge systems

edge, Google proposes the concept of knowledge graphs (KGs) in 2012 [37]. KGs arrange the structured multi-relational data of both concrete and abstract entities in the real world, which can be regarded as graph-structured KBs. In addition to describing world knowledge in conventional forms such as strings, the emergence of KGs provides a new tool to organize world knowledge from the perspective of entities and relations. Since KGs are very suitable for organizing the massive amount of knowledge stored in the Web corpora for faster knowledge retrieval, the construction of KGs has been blooming in recent years and has attracted wide attention from academia and industry.

KGs are usually constructed from existing Semantic Web datasets in resource description framework (RDF) [81] with the help of manual annotation. At the same time, KGs can also be automatically enriched by extracting knowledge from the massive plain text on the Web. As shown in Fig. 9.4, a typical KG usually contains two elements: entities and relations. Both concrete objects and abstract concepts in the real world are defined as entities, while complex associations between entities are defined as relations. Knowledge is usually represented in the triplet form of $\langle$*head entity*, `relation`, *tail entity*$\rangle$, and we abridge this as $\langle h, r, t \rangle$. For example, *Mark Twain* is a famous American writer, and *The Million Pound Bank Note* is one of his masterpieces. In a KG, this knowledge will be represented as $\langle$*The Million Pound Bank Note*, `Author`, *Mark Twain*$\rangle$. Owing to the well-structured form, KGs are widely used in various applications to improve system performance. There are several KGs widely utilized nowadays in NLP, such as Freebase [8], DBpedia [106], YAGO [147], and Wikidata [156]. There are also many comparatively smaller KGs in specific domains whose knowledge can function in domain-specific tasks.

9.2.2 Model Knowledge

For grammar rules, expert systems, and even KGs, one of the pain points of these symbolic knowledge systems is their weak generalization. In addition, it is also difficult to process symbolic knowledge using the numerical computing operations that machines are good at. Therefore, it becomes important to establish a knowledge framework based on numerical computing and with a strong generalization ability to serve the processing of natural languages. To this end, statistical learning [80] has been widely applied after the 1990s, including support vector machines [14], decision trees [16], conditional random fields [79], and so on. These data-driven statistical learning methods can acquire knowledge from data, use numerical features to implicitly describe knowledge, use probability models to represent rules behind knowledge implicitly, and perform knowledge reasoning based on probability computing.

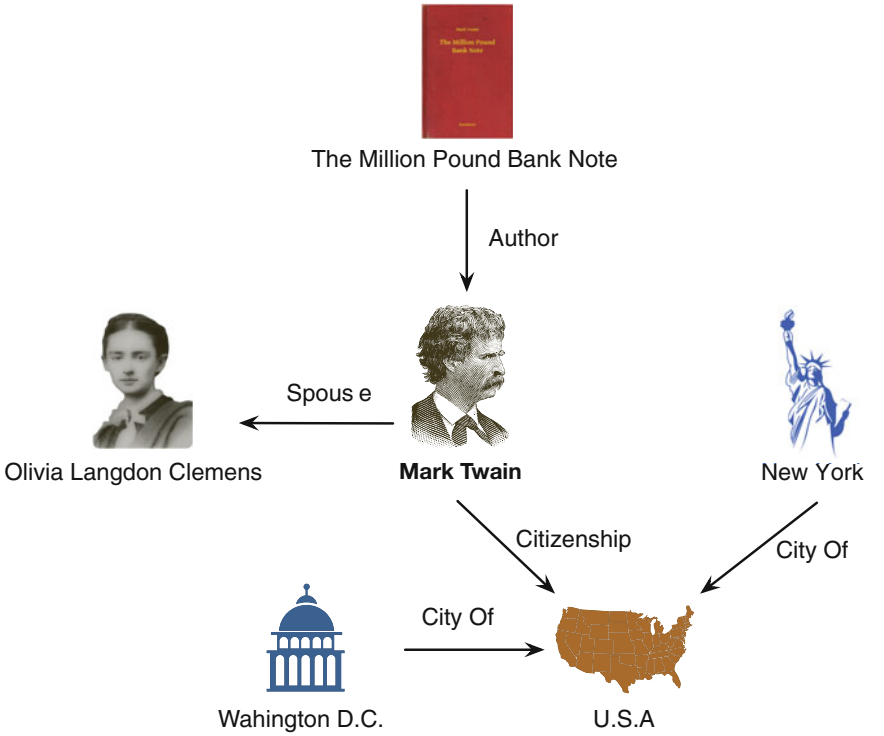

Fig. 9.4 An example of knowledge graphs. The graphics in the figure come from Vecteezy or Wikidata (Vecteezy: https://www.vecteezy.com; Wikidata: https://www.wikidata.org). And all these graphics are available for free or under the Creative Commons CC0 license

Different from symbolic knowledge, which is abstracted by human beings and regarded as *human-friendly* knowledge, the intrinsic nature of statistical learning is to capture the distribution patterns of data from statistics and uses these patterns to abstract implicit knowledge that can be used to solve specific problems. Although such implicit knowledge captured by statistical learning methods may not directly satisfy human intuition, the knowledge is adept at describing correlation information in data and is easy to compute numerically. In other words, this kind of knowledge based on numerical features and continuous probability models is more *machine-friendly*. Considering that the structure of probability models is also a kind of prior knowledge, here we introduce the concept of *model knowledge* to describe this machine-friendly knowledge.[2]

[2] There are also some works [22] using *continuous knowledge* to refer to this machine-friendly implicit knowledge.

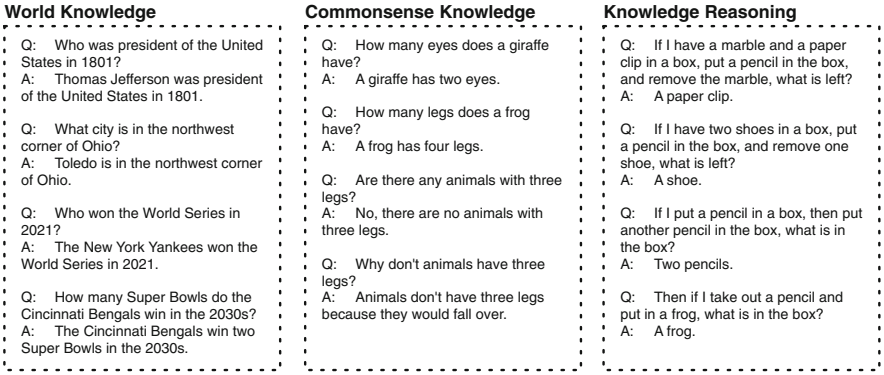

Fig. 9.5 Some question-answering examples of GPT-3. All these examples come from Kevin Lacker's blog *Giving GPT-3 a Turing Test* (https://lacker.io/ai/2020/07/06/giving-gpt-3-a-turing-test.html). These examples are also shown in the survey of PTMs [59]

In recent years, the boom of neural networks has provided a more powerful tool to capture model knowledge from data. Compared to conventional statistical models, neural networks are more expressive and can obtain more complex patterns from data. After the success of representing words as distributed representations [107], using shallow neural networks to learn low-dimensional continuous representations for concerned objects, such as words [123], graphs [152], sentences, and documents [82], has become a standard paradigm for accomplishing various NLP tasks. With the emergence of techniques that support increasing network depth and parameter size, such as Transformers [154], large-scale pre-trained models (PTMs) [33, 99, 206] based on deep neural networks are proposed. Recent works show that PTMs can capture rich lexical knowledge[69], syntactic knowledge[66], semantic knowledge[192], and factual knowledge[126] from data during the self-supervised pre-training stage. As shown in Fig. 9.5, we can find that GPT-3 (a PTM with 175 billion parameters) holds a certain amount of facts and commonsense and can perform logical reasoning [17]. By stimulating the task-specific model knowledge distributed in PTMs via various tuning methods [34], PTMs achieve state-of-the-art results on many NLP tasks.

9.2.3 Integrating Symbolic Knowledge and Model Knowledge

In the previous sections, we have briefly described many efforts made by researchers to enhance the processing of natural languages with knowledge. Briefly, symbolic knowledge is suited for reasoning and modeling causality, and model knowledge is suited for integrating information and modeling correlation. Symbolic knowledge and model knowledge have their own strengths, and utilizing both is crucial to drive the language understanding of machines. Over the past few decades, the practice has

also shown that NLP models with deep language understanding cannot be achieved by using a certain kind of knowledge alone.

The early explorations of NLP researchers relied entirely on handcrafted rules and systems, the limitations of which have been revealed over the two AI winters [54]. These AI winters have shown that it is challenging to make machines master versatile language abilities using only symbolic knowledge. In recent years, researchers have devoted great attention to deep neural networks and automatically learning model knowledge from massive data, leading to breakthroughs such as word representations and PTMs. However, these data-driven methods that focus on model knowledge still have some obvious limitations and face the challenges of robustness and interpretability [83]. In terms of robustness, for neural-based NLP models, it is not difficult to build adversarial examples to induce model errors [157], considering the knowledge automatically summarized by models may be a shortcut [48] or even a bias [148]. In terms of interpretability, the predictions given by models are also based on black-box correlations. Moreover, current data-driven methods may suffer from data-hungry issues. Model knowledge needs to be learned based on massive data, but obtaining high-quality data itself is very difficult. Humans can learn skills with a few training examples, which is challenging for machines. Therefore, relying solely on data-driven methods and model knowledge to advance NLP also seems unsustainable.

We have systematically discussed the symbolic and distributed representations of text in the previous chapters, and here we have made a further extension to form a broader discussion of representing knowledge. From these discussions, we can observe that taking full advantage of knowledge, i.e., utilizing both symbolic or model knowledge, is an important way to obtain better language understanding. Some recent works [60, 193] have also shown a trend toward the integration of symbolic and model knowledge and, more specifically, a trend of using symbolic knowledge to improve deep neural models that already have strong model knowledge. In order to integrate both symbolic and model knowledge, three challenges have to be addressed:

1. How to represent knowledge (especially symbolic knowledge) in a machine-friendly form so that current NLP models can utilize the knowledge?
2. How to use knowledge representations to guide specific NLP models?
3. How to continually acquire knowledge from large-scale plain text instead of handcrafted efforts?

We will next introduce knowledge representation learning, knowledge-guided NLP, and knowledge acquisition for these challenges.

9.3 Knowledge Representation Learning

As we mentioned before, we can organize knowledge using symbolic systems. However, as the scale of knowledge increases, using these symbolic systems

naturally faces two challenges: data sparsity and computational inefficiency. Despite the importance of symbolic knowledge for NLP, these challenges indicate that symbolic systems are not an inherently machine-friendly form of knowledge organization. Specifically, data sparsity is a common problem in many fields. For example, when we use KGs to describe general world knowledge, the number of entities (nodes) in KGs can be enormous, while the number of relations (edges) in KGs is typically few, i.e., there are often no relations between two randomly selected entities in the real world, resulting in the sparsity of KGs. Computational inefficiency is another challenge we have to overcome since computers are better suited to handle numerical data and less adept at handling symbolic knowledge in KGs. As the size of KGs continues to grow, this efficiency challenge may become more severe.

To solve the above problems, distributed knowledge representations are introduced, i.e., low-dimensional continuous embeddings are used to represent symbolic knowledge. The sparsity problem is alleviated owing to using these distributed representations, and the computational efficiency is also improved. In addition, using embeddings to represent knowledge makes it more feasible and convenient to integrate symbolic knowledge into neural NLP models, motivating the exploration of knowledge-guided NLP. Up to now, distributed knowledge representations have been widely used in many applications requiring the support of human knowledge. Moreover, distributed knowledge representations can also significantly improve the ability of knowledge completion, knowledge fusion, and knowledge reasoning.

In this section, we take KGs that organize rich world knowledge as an example to introduce how to obtain distributed knowledge representations. Hereafter, we use $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$ to denote a KG, in which $\mathcal{E} = \{e_1, e_2, \dots\}$ is the entity set, $\mathcal{R} = \{r_1, r_2, \dots\}$ is the relation set, and $\mathcal{T}$ is the fact set. We use $h, t \in \mathcal{E}$ to represent the head and tail entities, and $\mathbf{h}, \mathbf{t}$ to represent their entity embeddings. A triplet $\langle h, r, t\rangle \in \mathcal{T}$ is a factual record, where h, t are entities and r is the relation between h and t.

Given a triplet $\langle h, r, t\rangle$, a score function $f(h, r, t)$ is used by knowledge representation learning methods to measure whether $\langle h, r, t\rangle$ is a fact or a fallacy. Generally, the larger the value of $f(h, r, t)$, the higher the probability that $\langle h, r, t\rangle$ is true.[3] Based on $f(h, r, t)$, knowledge representations can be learned with

$$\arg\min_{\theta} \sum_{\langle h,r,t\rangle \in \mathcal{T}} \sum_{\langle \tilde{h},\tilde{r},\tilde{t}\rangle \in \tilde{\mathcal{T}}} \max\left\{0, f(\tilde{h}, \tilde{r}, \tilde{t}) + \gamma - f(h, r, t)\right\}, \tag{9.1}$$

where θ is the learnable embeddings of entities and relations, $\langle h, r, t\rangle$ indicates positive facts (i.e., triplets in $\mathcal{T}$), and $\langle \tilde{h}, \tilde{r}, \tilde{t}\rangle$ indicates negative facts (triplets that

[3] For some methods, the smaller the value of $f(h, r, t)$, the higher the probability that $\langle h, r, t\rangle$ is true. We re-formalize the score functions of these methods by taking the opposite of the score functions so that we can present all knowledge representation learning methods within a unified framework.

do not exist in KGs). $\gamma > 0$ is a hyper-parameter used as a margin. A bigger γ means to learn a wider gap between $f(h, r, t)$ and $f(\tilde{h}, \tilde{r}, \tilde{t})$. Considering that there are no explicit negative triplets in KGs, $\tilde{\mathcal{T}}$ is usually defined as

$$\begin{aligned}\tilde{\mathcal{T}} = \{\langle \tilde{h}, r, t\rangle | \tilde{h} \in \mathcal{E}, \langle h, r, t\rangle \in \mathcal{T}\} \cup \{\langle h, \tilde{r}, t\rangle | \tilde{r} \in \mathcal{R}, \langle h, r, t\rangle \in \mathcal{T}\} \\ \cup \{\langle h, r, \tilde{t}\rangle | \tilde{t} \in \mathcal{E}, \langle h, r, t\rangle \in \mathcal{T}\} - \mathcal{T},\end{aligned} \tag{9.2}$$

which means $\tilde{\mathcal{T}}$ is built by corrupting the entities and relations of the triplets in $\mathcal{T}$. Different from the margin-based loss function in Eq. (9.1), some methods apply a likelihood-based loss function to learn knowledge representations as

$$\arg\min_{\theta} \sum_{\langle h,r,t\rangle \in \mathcal{T}} \log\left[1 + \exp(-f(h, r, t))\right] + \sum_{\langle \tilde{h},\tilde{r},\tilde{t}\rangle \in \tilde{\mathcal{T}}} \log\left[1 + \exp(f(\tilde{h}, \tilde{r}, \tilde{t}))\right]. \tag{9.3}$$

Next, we present some typical knowledge representation learning methods as well as their score functions, including (1) linear representation methods that formalize relations as linear transformations between entities, (2) translation representation methods that formalize relations as translation operations between entities, (3) neural representation methods that apply neural networks to represent entities and relations, and (4) manifold representation methods that use complex manifold spaces instead of simple Euclidean spaces to learn representations.

9.3.1 Linear Representation

Linear representation methods formalize relations as linear transformations between entities, which is a simple and basic way to learn knowledge representations.

Structured Embeddings (SE) SE [13] is a typical linear method to represent KGs. In SE, all entities are embedded into a d-dimensional space. SE designs two relation-specific matrices $\mathbf{M}_{r,1}, \mathbf{M}_{r,2} \in \mathbb{R}^{d\times d}$ for each relation r, and these two matrices are used to transform the embeddings of entities. The score function of SE is defined as

$$f(h, r, t) = -\|\mathbf{M}_{r,1}\mathbf{h} - \mathbf{M}_{r,2}\mathbf{t}\|, \tag{9.4}$$

where $\|\cdot\|$ is the vector norm. The assumption of SE is that the head and tail embeddings should be as close as possible after being transformed into a relation-specific space. Therefore, SE uses the margin-based loss function to learn representations.

Semantic Matching Energy (SME) SME [11] builds more complex linear transformations than SE. Given a triplet $\langle h, r, t\rangle$, $\mathbf{h}$ and $\mathbf{r}$ are combined using a projection function to get a new embedding $\mathbf{l}_{h,r}$. Similarly, given $\mathbf{t}$ and $\mathbf{r}$, we can get $\mathbf{l}_{t,r}$. Then,

a point-wise multiplication function is applied on $\mathbf{l}_{h,r}$ and $\mathbf{l}_{t,r}$ to get the score of this triplet. SME introduces two different projection functions to build $f(h, r, t)$: one is in the linear form

$$f(h, r, t) = \mathbf{l}_{h,r}^{\top} \mathbf{l}_{t,r}, \quad \mathbf{l}_{h,r} = \mathbf{M}_1 \mathbf{h} + \mathbf{M}_2 \mathbf{r} + \mathbf{b}_1, \quad \mathbf{l}_{t,r} = \mathbf{M}_3 \mathbf{t} + \mathbf{M}_4 \mathbf{r} + \mathbf{b}_2, \tag{9.5}$$

and the other is in the bilinear form

$$f(h, r, t) = \mathbf{l}_{h,r}^{\top} \mathbf{l}_{t,r}, \quad \mathbf{l}_{h,r} = (\mathbf{M}_1 \mathbf{h} \odot \mathbf{M}_2 \mathbf{r}) + \mathbf{b}_1, \quad \mathbf{l}_{t,r} = (\mathbf{M}_3 \mathbf{t} \odot \mathbf{M}_4 \mathbf{r}) + \mathbf{b}_2, \tag{9.6}$$

where $\odot$ is the element-wise (Hadamard) product. $\mathbf{M}_1$, $\mathbf{M}_2$, $\mathbf{M}_3$, and $\mathbf{M}_4$ are learnable transformation matrices, and $\mathbf{b}_1$ and $\mathbf{b}_2$ are learnable bias vectors. Empirically, the margin-based loss function is suitable for dealing with the score functions built with vector norm operations, while the likelihood-based loss function is more usually used to process the score functions built with inner product operations. Since SME uses the inner product operation to build its score function, the likelihood-based loss function is thus used to learn representations.

Latent Factor Model (LFM) LFM [70] aims to model large KGs based on a bilinear structure. By modeling entities as embeddings and relations as matrices, the score function of LFM is defined as

$$f(h, r, t) = \mathbf{h}^{\top} \mathbf{M}_r \mathbf{t}, \tag{9.7}$$

where the matrix $\mathbf{M}_r$ is the representation of the relation r. Similar to SME, LFM adopts the likelihood-based loss function to learn representations. Based on LFM, DistMult [186] further restricts $\mathbf{M}_r$ to be a diagonal matrix. As compared with LFM, DistMult not only reduces the parameter size but also reduces the computational complexity and achieves better performance.

RESCAL RESCAL [118, 119] is a representation learning method based on matrix factorization. By modeling entities as embeddings and relations as matrices, RESCAL adopts a score function the same to LFM. However, RESCAL employs neither the margin-based nor the likelihood-based loss function to learn knowledge representations. Instead, in RESCAL, a three-way tensor $\overrightarrow{\mathbf{X}} \in \mathbb{R}^{|\mathcal{E}| \times |\mathcal{E}| \times |\mathcal{R}|}$ is adopted. In the tensor $\overrightarrow{\mathbf{X}}$, two modes respectively stand for head and tail entities, while the third mode stands for relations. The entries of $\overrightarrow{\mathbf{X}}$ are determined by the existence of the corresponding triplet facts. That is, $\overrightarrow{\mathbf{X}}_{ijk} = 1$ if the triplet $\langle i\text{-}th\ entity, k\text{-}th\ relation, j\text{-}th\ entity \rangle$ exists in the training set, and otherwise $\overrightarrow{\mathbf{X}}_{ijk} = 0$. To capture the inherent structure of all triplets, given $\overrightarrow{\mathbf{X}} = \{\mathbf{X}_1, \cdots, \mathbf{X}_{|\mathcal{R}|}\}$, for each slice $\mathbf{X}_n = \overrightarrow{\mathbf{X}}_{[:,:,n]}$, RESCAL assumes the following factorization for $\mathbf{X}_n$ holds

$$\mathbf{X}_n \approx \mathbf{E} \mathbf{M}_{r_n} \mathbf{E}^{\top}, \tag{9.8}$$

where $\mathbf{E} \in \mathbb{R}^{|\mathcal{E}| \times d}$ stands for the d-dimensional entity representations of all entities and $\mathbf{M}_{r_n} \in \mathbb{R}^{d \times d}$ represents the interactions between entities specific to the n-th relation r_n. Following this tensor factorization assumption, the learning objective of RESCAL is defined as

$$\arg\min_{\mathbf{E},\mathbf{M}} \frac{1}{2}\left(\sum_{n=1}^{|\mathcal{R}|} \|\mathbf{X}_n - \mathbf{E}\mathbf{M}_{r_n}\mathbf{E}^\top\|_F^2\right) + \frac{1}{2}\lambda\left(\|\mathbf{E}\|_F^2 + \sum_{n=1}^{|\mathcal{R}|} \|\mathbf{M}_{r_n}\|_F^2\right), \tag{9.9}$$

where $\mathbf{M} = \{\mathbf{M}_{r_1}, \mathbf{M}_{r_2}, \cdots, \mathbf{M}_{r_{|\mathcal{R}|}}\}$ is the collection of all relation matrices, $\|\cdot\|_F$ is the Frobenius vector norm, and λ is a hyper-parameter to control the second regularization term.

Holographic Embeddings (HolE) HolE [117] is proposed as an enhanced version of RESCAL. RESCAL works well with multi-relational data but suffers from a high computational complexity. To achieve high effectiveness and efficiency at the same time, HolE employs an operation named circular correlation to generate representations. The circular correlation operation $\star : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$ between two entities h and t is

$$[\mathbf{h} \star \mathbf{t}]_k = \sum_{i=1}^{d} [\mathbf{h}]_i [\mathbf{t}]_{(k+i) \bmod d+1}, \tag{9.10}$$

where $[\cdot]_i$ means the i-th vector element. The score function is defined as

$$f(h, r, t) = -\mathbf{r}^\top(\mathbf{h} \star \mathbf{t}). \tag{9.11}$$

HolE adopts the likelihood-based loss function to learn representations.

The circular correlation operation brings several advantages. First, the circular correlation operation is noncommutative (i.e., $\mathbf{h} \star \mathbf{t} \neq \mathbf{t} \star \mathbf{h}$), which makes it capable of modeling asymmetric relations in KGs. Second, the circular correlation operation has a lower computational complexity compared to the tensor product operation in RESCAL. Moreover, the circular correlation operation could be further accelerated with the help of fast Fourier transform (FFT), which is formalized as

$$\mathbf{h} \star \mathbf{t} = \mathcal{F}^{-1}(\overline{\mathcal{F}(\mathbf{h})} \odot \mathcal{F}(\mathbf{t})), \tag{9.12}$$

where $\mathcal{F}(\cdot)$ and $\mathcal{F}(\cdot)^{-1}$ represent the FFT operation and its inverse operation, respectively, $\overline{\mathcal{F}(\cdot)}$ denotes the complex conjugate of $\mathcal{F}(\cdot)$, and $\odot$ stands for the element-wise (Hadamard) product. Due to the FFT operation, the computational complexity of the circular correlation operation is $\mathcal{O}(d \log d)$, which is much lower than that of the tensor product operation.

9.3.2 Translation Representation

Translation methods are another effective way to obtain distributed representations of KGs. To help readers better understand different translation representation methods, we first introduce their motivations.

The primary motivation is that it is natural to consider relations between entities as translation operations. For distributed representations, entities are embedded into a low-dimensional space, and ideal representations should embed entities with similar semantics into the nearby regions, while entities with different meanings should belong to distinct clusters. For example, *William Shakespeare* and *Jane Austen* may be in the same cluster of writers, *Romeo and Juliet* and *Pride and Prejudice* may be in another cluster of books. In this case, they share the same relation `Notable Work`, and the translations from writers to their books in the embedding space are similar.

The secondary motivation of translation methods derives from the breakthrough in word representation learning. Word2vec [108] proposes two simple models, skip-gram and CBOW, to learn distributed word representations from large-scale corpora. The learned word embeddings perform well in measuring word similarities and analogies. And these word embeddings have some interesting phenomena: if the same semantic or syntactic relations are shared by two word pairs, the translations within the two word pairs are similar. For instance, we have

$$\mathbf{w}(king) - \mathbf{w}(man) \approx \mathbf{w}(queen) - \mathbf{w}(woman), \tag{9.13}$$

where $\mathbf{w}(\cdot)$ represents the embedding of the word. We know that the semantic relation between *king* and *man* is similar to the relation between *queen* and *woman*, and the above case shows that this relational knowledge is successfully embedded into word representations. Apart from semantic relations, syntactic relations can also be well represented by word2vec, as shown in the following example:

$$\mathbf{w}(bigger) - \mathbf{w}(big) \approx \mathbf{w}(smaller) - \mathbf{w}(small). \tag{9.14}$$

Since word2vec implies that the implicit relations between words can be seen as translations, it is reasonable to assume that relations in KGs can also be modeled as translation embeddings. More intuitively, if we represent a word pair and its implicit relation using a triplet, e.g., $\langle big, \texttt{Comparative}, bigger \rangle$, we can obviously observe the similarity between word representation learning and knowledge representation learning.

The last motivation comes from the consideration of the computational complexity. On the one hand, the substantial increase in the model complexity will result in high computational costs and obscure model interpretability, and a complex model may lead to overfitting. On the other hand, the experimental results on the model complexity demonstrate that the simpler models perform almost as well as more expressive models in most knowledge-related applications [117, 186], in the

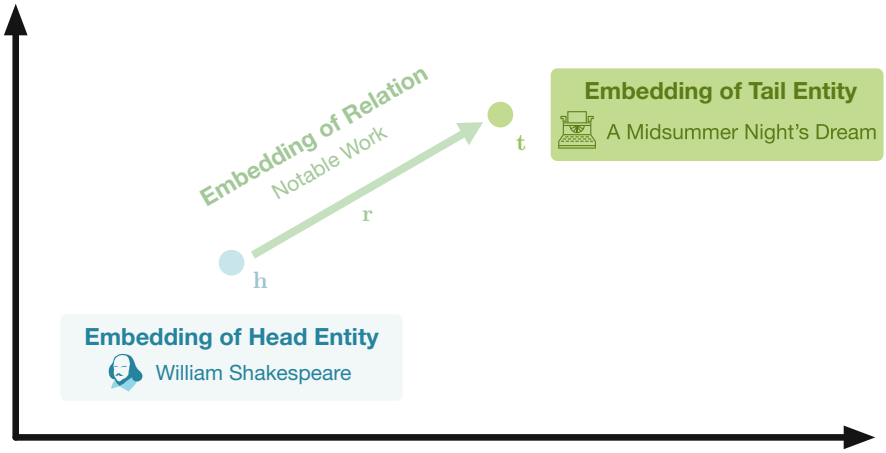

Fig. 9.6 The architecture of TransE [12]

condition that large-scale datasets and a relatively large amount of relations can be used for training models. As KG size increases, the computational complexity becomes the primary challenge for knowledge representation learning. The intuitive assumption of modeling relations as translations rather than matrices leads to a better trade-off between effectiveness and efficiency.

Since the translation-based methods are all extended from TransE [12], we thus first introduce TransE in detail and then introduce its extensions.

TransE As illustrated in Fig. 9.6, TransE embeds entities as well as relations into the same space. In the embedding space, relations are considered as translations from head entities to tail entities. With this translation assumption, given a triplet $\langle h, r, t \rangle$ in $\mathcal{T}$, we want $\mathbf{h} + \mathbf{r}$ to be the nearest neighbor of the tail embedding $\mathbf{t}$. The score function of TransE is then defined as

$$f(h, r, t) = -\|\mathbf{h} + \mathbf{r} - \mathbf{t}\|. \tag{9.15}$$

TransE uses the margin-based loss function for training. Although TransE is effective and efficient, it still has several challenges to be further explored.

First, considering that there may be multiple correct answers given two elements in a triplet, under the translation assumption in TransE, each entity has only one embedding in all triplets, which may lead to reducing the discrimination of entity embeddings. In TransE, according to the entity cardinalities of relations, all relations are classified into four categories, 1-to-1, 1-to-Many, Many-to-1, and Many-to-Many. A relation is considered as 1-to-1 if one head appears with only one tail and vice versa, 1-to-Many if a head can appear with many tails, Many-to-1 if a tail can appear with many heads, and Many-to-Many if multiple heads appear with multiple tails. Statistics demonstrate that the 1-to-Many, Many-to-1, and Many-to-Many

relations occupy a large proportion. TransE performs well on 1-to-1 relations, but has problems when handling 1-to-Many, Many-to-1, and Many-to-Many relations. For instance, given the head entity *William Shakespeare* and the relation `Notable Work`, we can get a list of masterpieces, such as *Hamlet*, *A Midsummer Night's Dream*, and *Romeo and Juliet*. These books share the same writer information while differing in many other fields such as theme, background, and famous roles in the book. Due to the entity *William Shakespeare* and the relation `Notable Work`, these books may be assigned similar embeddings and become indistinguishable.

Second, although the translation operation is intuitive and effective, only considering the simple one-step translation may limit the ability to model KGs. Taking entities and relations as nodes and edges, the nodes that are not directly connected may be linked by a path of more than one edge. However, TransE focuses on minimizing $\|\mathbf{h}+\mathbf{r}-\mathbf{t}\|$, which only utilizes the one-step relation information in KGs, regardless of the latent relationships located in long-distance paths. For example, if we know $\langle$*The forbidden city*, `Located in`, *Beijing*$\rangle$ and $\langle$*Beijing*, `Capital of`, *China*$\rangle$, we can infer that *The forbidden city* locates in *China*. TransE can be further enhanced with the favor of multistep information.

Third, the representation and the score function in TransE are oversimplified for the consideration of efficiency. Therefore, TransE may not be capable enough of modeling those complex entities and relations in KGs. There are still some challenges in how to balance effectiveness and efficiency as well as avoid overfitting and underfitting.

After TransE, there are lots of subsequent methods addressing the above challenges. Specifically, TransH [165], TransR [90], TransD [102], and TranSparse [71] are proposed to solve the challenges in modeling complex relations, PTransE is proposed to encode long-distance information located in multistep paths, and CTransR, TransG, and KG2E further extend the oversimplified model of TransE. Next, we will discuss these subsequent methods in detail.

TransH TransH [165] enables an entity to have multiple relation-specific representations to address the issue that TransE cannot well model 1-to-Many, Many-to-1, and Many-to-Many relations. As we mentioned before, in TransE, entities are embedded to the same semantic embedding space and similar entities tend to be in the same cluster. However, it seems that *William Shakespeare* should be in the neighborhood of *Isaac Newton* when talking about `Nationality`, while it should be close to *Mark Twain* when talking about `Occupation`. To accomplish this, entities should have multiple representations in different triplets.

As illustrated in Fig. 9.7, TransH proposes a hyperplane $\mathbf{w}_r$ for each relation, and computes the translation on the hyperplane $\mathbf{w}_r$. Given a triplet $\langle h, r, t\rangle$, TransH projects $\mathbf{h}$ and $\mathbf{t}$ to the corresponding hyperplane $\mathbf{w}_r$ to get the projection $\mathbf{h}_\perp$ and $\mathbf{t}_\perp$, and $\mathbf{r}$ is used to connect $\mathbf{h}_\perp$ and $\mathbf{t}_\perp$:

$$\mathbf{h}_\perp = \mathbf{h} - \mathbf{w}_r^\top \mathbf{h}\mathbf{w}_r, \quad \mathbf{t}_\perp = \mathbf{t} - \mathbf{w}_r^\top \mathbf{t}\mathbf{w}_r, \tag{9.16}$$

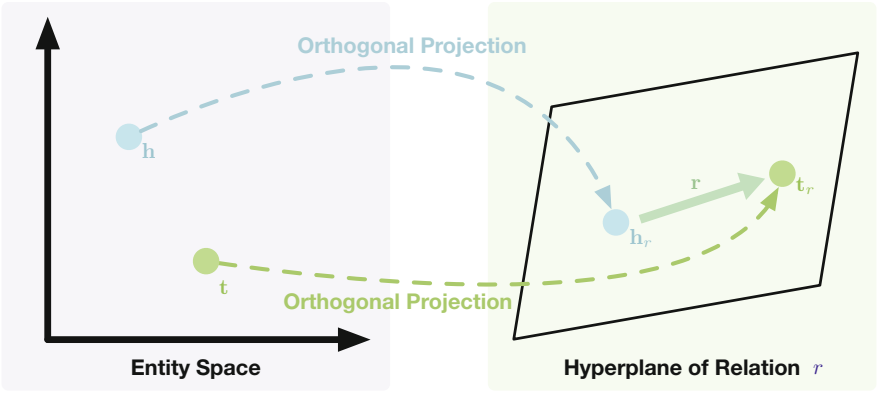

Fig. 9.7 The architecture of TransH [165]

where $\mathbf{w}_r$ is a vector and $\|\mathbf{w}_r\|_2$ is restricted to 1. The score function is

$$f(h, r, t) = -\|\mathbf{h}_\perp + \mathbf{r} - \mathbf{t}_\perp\|. \tag{9.17}$$

As for training, TransH also minimizes the margin-based loss function with negative sampling, which is similar to TransE.

TransR TransR [90] takes full advantage of linear methods and translation methods. As in Eq. (9.16), TransH enables entities to have multiple relation-specific representations by projecting them to different hyperplanes, while entity embeddings and relation embeddings are still restricted in the same space, which may limit the ability for modeling entities and relations. TransR assumes that entity embeddings and relation embeddings should be in different spaces.

As illustrated in Fig. 9.8, For a triplet $\langle h, r, t\rangle$, TransR projects $\mathbf{h}$ and $\mathbf{t}$ to the relation space of r, and this projection is defined as

$$\mathbf{h}_r = \mathbf{h}\mathbf{M}_r, \quad \mathbf{t}_r = \mathbf{t}\mathbf{M}_r, \tag{9.18}$$

where $\mathbf{M}_r$ is the projection matrix. $\mathbf{h}_r$ and $\mathbf{t}_r$ stand for the relation-specific entity representations in the relation space of r, respectively. This means that each entity has a relation-specific representation for each relation, and all translation operations are processed in the relation-specific space. The score function of TransR is

$$f(h, r, t) = -\|\mathbf{h}_r + \mathbf{r} - \mathbf{t}_r\|. \tag{9.19}$$

TransR constrains the norms of the embeddings and has $\|\mathbf{h}\|_2 \leq 1$, $\|\mathbf{t}\|_2 \leq 1$, $\|\mathbf{r}\|_2 \leq 1$, $\|\mathbf{h}_r\|_2 \leq 1$, $\|\mathbf{t}_r\|_2 \leq 1$. As for training, TransR uses the same margin-based loss function as TransE.

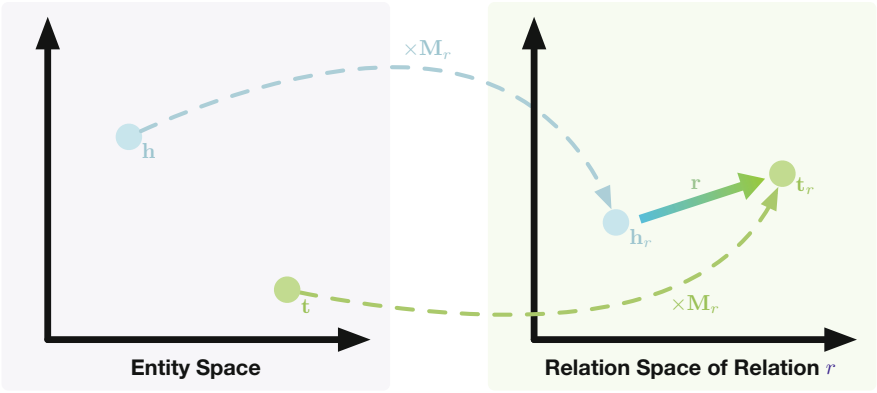

Fig. 9.8 The architecture of TransR [90]

Furthermore, a relation should also have multiple representations since the meanings of a relation with different head and tail entities differ slightly. For example, the relation `Contains the Location` has head-tail patterns like *city-street*, *country-city*, and even *country-university*, each conveys different attribute information. To handle these subtle differences, entities for a same relation should also be projected differently.

To this end, cluster-based TransR (CTransR) is then proposed, which is an enhanced version of TransR by taking the nuance in meaning for a same relation with different entities into consideration. More specifically, for each relation, all entity pairs of the relation are first clustered into several groups. The clustering process depends on the result of $\mathbf{t} - \mathbf{h}$ for each entity pair (h, t), and $\mathbf{h}$ and $\mathbf{t}$ are the embeddings learned by TransE. Then, we assign a distinct sub-relation embedding $\mathbf{r}_c$ for each cluster of the relation r according to cluster-specific entity pairs, and the original score function of TransR is modified as

$$f(h, r, t) = -\|\mathbf{h}_r + \mathbf{r}_c - \mathbf{t}_r\| - \lambda\|\mathbf{r}_c - \mathbf{r}\|, \tag{9.20}$$

where λ is a hyper-parameter to control the regularization term and $\|\mathbf{r}_c - \mathbf{r}\|$ is to make the sub-relation embedding $\mathbf{r}_c$ and the unified relation embedding $\mathbf{r}$ not too distinct.

TransD TransD [102] is an extension of TransR that uses dynamic mapping matrices to project entities into relation-specific spaces. TransR focuses on learning multiple relation-specific entity representations. However, TransR projects entities according to only relations, ignoring the entity diversity. Moreover, the projection operations based on matrix-vector multiplication lead to a higher computational complexity compared to TransE, which is time-consuming when applied on large-scale KGs.

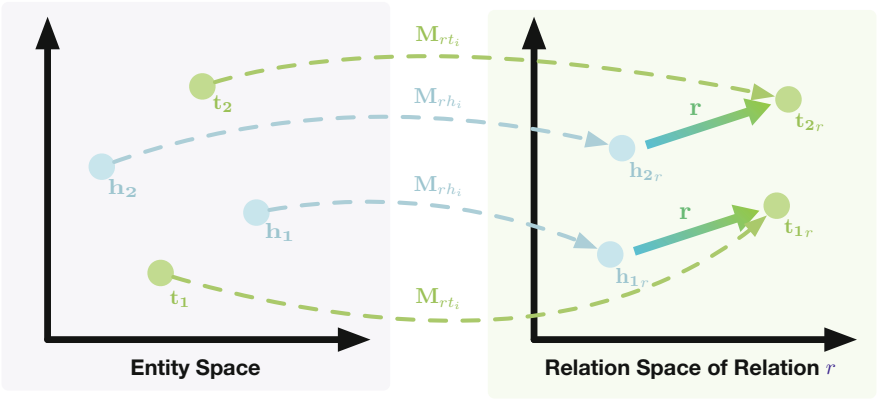

Fig. 9.9 The architecture of TransD [102]

For each entity and relation, TransD defines two vectors: one is used as the embedding, and the other is used to construct projection matrices to map entities to relation spaces. As illustrated in Fig. 9.9, We use $\mathbf{h}$, $\mathbf{t}$, $\mathbf{r}$ to denote the embeddings of entities and relations, and $\mathbf{h}_p$, $\mathbf{t}_p$, $\mathbf{r}_p$ to represent the projection vectors. There are two projection matrices $\mathbf{M}_{rh}, \mathbf{M}_{rt}$ used to project entities to relation spaces, and these projection matrices are dynamically constructed as

$$\mathbf{M}_{rh} = \mathbf{r}_p \mathbf{h}_p^\top + \mathbf{I}, \quad \mathbf{M}_{rt} = \mathbf{r}_p \mathbf{t}_p^\top + \mathbf{I}, \tag{9.21}$$

which means the projection vectors of both entities and relations are combined to determine dynamic projection matrices. The score function is

$$f(h, r, t) = -\|\mathbf{M}_{rh}\mathbf{h} + \mathbf{r} - \mathbf{M}_{rt}\mathbf{t}\|. \tag{9.22}$$

These projection matrices are initialized with identity matrices by setting all the projection vectors to $\mathbf{0}$ at initialization, and the normalization constraints in TransR are also used for TransD.

TransD proposes a dynamic method to construct projection matrices by considering the diversities of both entities and relations, achieving better performance compared to existing methods in knowledge completion. Moreover, TransD lowers both computational and spatial complexity compared to TransR.

TranSparse TranSparse [71] is also a subsequent work of TransR. Although TransR has achieved promising results, there are still two challenges remained. One is the heterogeneity challenge. Relations in KGs differ in granularity. Some relations express complex semantics between entities, while some other relations are relatively simple. The other is the imbalance challenge. Some relations have more valid head entities and fewer valid tail entities, while some are the opposite.

If we consider these challenges rather than merely treating all relations equally, we can obtain better knowledge representations.

Existing methods such as TransR build projection matrices for each relation, and these projection matrices have the same parameter scale, regardless of the variety in the complexity of relations. TranSparse is proposed to address this issue. The underlying assumption of TranSparse is that complex relations should have more parameters to learn while simple relations should have fewer parameters, where the relation complexity is judged from the number of triplets or entities linked to the relation. To accomplish this, two models are proposed, including TranSparse-share and TranSparse-separate.

Inspired by TransR, given a relation r, TranSparse-share builds a relation-specific projection matrix $\mathbf{M}_r(\theta_r)$ for the relation. $\mathbf{M}_r(\theta_r)$ is sparse and the sparse degree θ_r mainly depends on the number of entity pairs linked to r. Suppose N_r is the number of linked entity pairs, N_r^* represents the maximum number of N_r, and $\theta_{\min}$ denotes the minimum sparse degree of projection matrices that $0 \leq \theta_{\min} \leq 1$. The sparse degree of relation r is defined as

$$\theta_r = 1 - (1 - \theta_{\min})\frac{N_r}{N_r^*}. \tag{9.23}$$

Both head and tail entities share the same sparse projection matrix $\mathbf{M}_r(\theta_r)$. The score function is

$$f(h, r, t) = -\|\mathbf{M}_r(\theta_r)\mathbf{h} + \mathbf{r} - \mathbf{M}_r(\theta_r)\mathbf{t}\|. \tag{9.24}$$

Different from TranSparse-share, TranSparse-separate builds two different sparse matrices $\mathbf{M}_{rh}(\theta_{rh})$ and $\mathbf{M}_{rt}(\theta_{rt})$ for head and tail entities, respectively. Then, the sparse degree θ_{rh} (or θ_{rt}) depends on the number of head (or tail) entities linked to r. We have N_{rh} (or N_{rt}) to represent the number of head (or tail) entities, as well as N_{rh}^* (or N_{rt}^*) to represent the maximum number of N_{rh} (or N_{rt}). And $\theta_{\min}$ will also be set as the minimum sparse degree of projection matrices that $0 \leq \theta_{\min} \leq 1$. We have

$$\theta_{rh} = 1 - (1 - \theta_{\min})N_{rh}/N_{rh}^*, \quad \theta_{rt} = 1 - (1 - \theta_{\min})N_{rt}/N_{rt}^*. \tag{9.25}$$

The final score function of TranSparse-separate is

$$f(h, r, t) = -\|\mathbf{M}_{rh}(\theta_{rh})\mathbf{h} + \mathbf{r} - \mathbf{M}_{rt}(\theta_{rt})\mathbf{t}\|. \tag{9.26}$$

Through sparse projection matrices, TranSparse solves the heterogeneity challenge and the imbalance challenge simultaneously.

PTransE PTransE [89] is an extension of TransE that considers multistep relational paths. All abovementioned translation methods only consider simple one-step paths (i.e., relation) to perform the translation operation, ignoring the rich global

information located in the whole KGs. For instance, if we notice the multistep relational path that ⟨*The forbidden city*, `Located in`, *Beijing*⟩ → ⟨*Beijing*, `Capital of`, *China*⟩, we can inference with confidence that the triplet ⟨*The forbidden city*, `Located in`, *China*⟩ may exist. Relational paths provide us a powerful way to build better representations for KGs and even help us better understand knowledge reasoning.

There are two main challenges when encoding the information in multistep relational paths. First, how to select reliable and meaningful relational paths among enormous path candidates in KGs, since there are lots of paths that cannot indicate reasonable relations. Consider two triplet facts ⟨*The forbidden city*, `Located in`, *Beijing*⟩ → ⟨*Beijing*, `held`, *2008 Summer Olympics*⟩, it is hard to describe the relation between *The forbidden city* and *2008 Summer Olympics*. Second, how to model the meaningful relational paths? It is not easy to handle the problem of semantic composition in relational paths.

To select meaningful relational paths, PTransE uses a path-constraint resource allocation (PCRA) algorithm to judge the path reliability. Suppose there is information (or resource) in the head entity h which will flow to the tail entity t through some certain paths. The basic assumption of PCRA is that the reliability of the path ℓ depends on the amount of resource that eventually flows from head to tail. Formally, we denote a certain path between h and t as $\ell = (r_1, \ldots, r_l)$. The resource that travels from h to t following the path could be represented as $S_0/h \xrightarrow{r_1} S_1 \xrightarrow{r_2} \ldots \xrightarrow{r_l} S_l/t$. For an entity $m \in S_i$, the amount of resource that belongs to m is defined as

$$R_\ell(m) = \sum_{n \in S_{i-1}(\cdot, m)} \frac{1}{|S_i(n, \cdot)|} R_\ell(n), \tag{9.27}$$

where $S_{i-1}(\cdot, m)$ indicates all direct predecessors of the entity m along with the relation r_i in S_{i-1} and $S_i(n, \cdot)$ indicates all direct successors of $n \in S_{i-1}$ with the relation r_i. Finally, the amount of resource that flows to the tail $R_\ell(t)$ is used to measure the reliability of ℓ, given the triplet $\langle h, \ell, t \rangle$.

Once we finish selecting those meaningful relational path candidates, the next challenge is to model the semantic composition of these multistep paths. PTransE proposes three composition operations, namely, addition, multiplication, and recurrent neural networks, to get the path representation $\mathbf{l}$ based on the relations in $\ell = (r_1, \ldots, r_l)$. The score function is

$$f(h, \ell, t) = -\|\mathbf{l} - (\mathbf{t} - \mathbf{h})\| \approx -\|\mathbf{l} - \mathbf{r}\| = f(\ell, r), \tag{9.28}$$

where r indicates the golden relation between h and t. Since PTransE also wants to meet the assumption in TransE that $\mathbf{r} \approx \mathbf{t} - \mathbf{h}$, PTransE directly utilizes $\mathbf{r}$ in training. The optimization objective of PTransE is

$$\arg\min_{\theta} \sum_{(h,r,t) \in \mathcal{T}} [\mathcal{L}(h, r, t) + \frac{1}{Z} \sum_{\ell \in P(h,t)} R(\ell|h, t) \mathcal{L}(\ell, r)], \tag{9.29}$$

where $\mathcal{L}(h, r, t)$ is the margin-based loss function with $f(h, r, t)$, $\mathcal{L}(\ell, r)$ is the margin-based score function with $f(\ell, r)$, and $Z = \sum_{\ell \in P(h,t)} R(\ell|h, t)$ is a normalization factor. The reliability $R(\ell|h, t)$ of ℓ in (h, ℓ, t) is well considered in the overall loss function. For the path ℓ, the initial resource is set as $R_\ell(h) = 1$. By recursively performing PCRA from h to t through ℓ, the resource $R_\ell(t)$ can indicate how much information can be well translated, and $R_\ell(t)$ is thus used to measure the reliability of the path ℓ, i.e., $R(\ell|h, t) = R_\ell(t)$. Besides PTransE, similar ideas [47, 50] also consider multistep relational paths and demonstrate that there is plentiful information located in multistep relational paths which could significantly improve knowledge representation.

KG2E KG2E [65] introduces multidimensional Gaussian distributions to represent KGs. Existing translation methods usually consider entities and relations as vectors embedded in low-dimensional spaces. However, as explained above, entities and relations in KGs are diverse at different granularities. Therefore, the margin in the margin-based loss function that is used to distinguish positive triplets from negative triplets should be more flexible due to the diversity, and the uncertainties of entities and relations should be taken into consideration.

KG2E represents entities and relations with Gaussian distributions. Specifically, the mean vector denotes the central position of an entity or a relation, and the covariance matrix denotes its uncertainties. Following the score function proposed in TransE, for $\langle h, r, t \rangle$, the Gaussian distributions of entities and relations are defined as

$$\mathbf{h} \sim \mathcal{N}(\boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h), \quad \mathbf{t} \sim \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t), \quad \mathbf{r} \sim \mathcal{N}(\boldsymbol{\mu}_r, \boldsymbol{\Sigma}_r). \tag{9.30}$$

Note that the covariances are diagonal for efficient computation. KG2E hypothesizes that head and tail entities are independent with specific relations; then, the translation could be defined as

$$\mathbf{e} \sim \mathcal{N}(\boldsymbol{\mu}_h - \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_h + \boldsymbol{\Sigma}_t). \tag{9.31}$$

To measure the dissimilarity between $\mathbf{e}$ and $\mathbf{r}$, KG2E considers both asymmetric similarity and symmetric similarity, and then proposes two methods.

The asymmetric similarity is based on the KL divergence between $\mathbf{e}$ and $\mathbf{r}$, which is a typical method to measure the similarity between two distributions. The score function is

$$\begin{aligned} f(h, r, t) &= -D_{\mathrm{KL}}(\mathbf{e} \| \mathbf{r}) \\ &= -\int_{x \in \mathbb{R}^d} \mathcal{N}(x; \boldsymbol{\mu}_r, \boldsymbol{\Sigma}_r) \log \frac{\mathcal{N}(x; \boldsymbol{\mu}_e, \boldsymbol{\Sigma}_e)}{\mathcal{N}(x; \boldsymbol{\mu}_r, \boldsymbol{\Sigma}_r)} dx \\ &= -\frac{1}{2}\{\mathrm{tr}(\boldsymbol{\Sigma}_r^{-1}\boldsymbol{\Sigma}_r) + (\boldsymbol{\mu}_r - \boldsymbol{\mu}_e)^\top \boldsymbol{\Sigma}_r^{-1}(\boldsymbol{\mu}_r - \boldsymbol{\mu}_e) - \log \frac{\det(\boldsymbol{\Sigma}_e)}{\det(\boldsymbol{\Sigma}_r)} - d\}, \end{aligned} \tag{9.32}$$

where $\mathrm{tr}(\boldsymbol{\Sigma})$ indicates the trace of $\boldsymbol{\Sigma}$ and $\boldsymbol{\Sigma}^{-1}$ indicates the inverse of $\boldsymbol{\Sigma}$.

The symmetric similarity is built on the expected likelihood and probability product kernel. KE2G takes the inner product between the probability density functions of **e** and **r** as the measurement of similarity. The logarithm of score function defined is

$$\begin{aligned} f(h,r,t) &= -\int_{x\in\mathbb{R}^d} \mathcal{N}(x;\boldsymbol{\mu}_e,\boldsymbol{\Sigma}_e)\mathcal{N}(x;\boldsymbol{\mu}_r,\boldsymbol{\Sigma}_r)dx \\ &= -\log \mathcal{N}(0;\boldsymbol{\mu}_e-\boldsymbol{\mu}_r,\boldsymbol{\Sigma}_e+\boldsymbol{\Sigma}_r) \\ &= -\frac{1}{2}\{(\boldsymbol{\mu}_e-\boldsymbol{\mu}_r)^\top(\boldsymbol{\Sigma}_e+\boldsymbol{\Sigma}_r)^{-1}(\boldsymbol{\mu}_e-\boldsymbol{\mu}_r) \\ &\quad + \log\det(\boldsymbol{\Sigma}_e+\boldsymbol{\Sigma}_r) + d\log(2\pi)\}. \end{aligned} \tag{9.33}$$

The optimization objective of KG2E is also margin-based similar to TransE. Both asymmetric and symmetric similarities are constrained by some regularizations to avoid overfitting:

$$\forall l \in \mathcal{E}\cup\mathcal{R}, \quad \|\boldsymbol{\mu}_l\|_2 \le 1, \quad c_{\min}\mathbf{I} \le \boldsymbol{\Sigma}_l \le c_{\max}\mathbf{I}, \quad c_{\min} > 0, \tag{9.34}$$

where $c_{\min}$ and $c_{\max}$ are the hyper-parameters as the restriction values for covariance.

TransG TransG [174] discusses the problem that some relations in KGs such as `Contains the Location` or `Part of` may have multiple sub-meanings, which is also discussed in TransR. In fact, these complex relations could be divided into several more precise sub-relations. To address this issue, CTransR is proposed with a preprocessing that clusters sub-relation according to entity pairs.

As illustrated in Fig. 9.10, TransG assumes that the embeddings containing several semantic components should follow a Gaussian mixture model. The generative process is:

1. For each entity $e \in E$, TransG sets a standard normal distribution: $\boldsymbol{\mu}_e \sim \mathcal{N}(\mathbf{0},\mathbf{I})$.
2. For a triplet $\langle h,r,t\rangle$, TransG uses the Chinese restaurant process (CRP) to automatically detect semantic components (i.e., sub-meanings in a relation): $\pi_{r,n} \sim \mathrm{CRP}(\beta)$. $\pi_{r,n}$ is the weight of the i-th component generated by the Chinese restaurant process from the data.
3. Draw the head embedding from a standard normal distribution: $\mathbf{h} \sim \mathcal{N}(\boldsymbol{\mu}_h,\sigma_h^2\mathbf{I})$.
4. Draw the tail embedding from a standard normal distribution: $\mathbf{t} \sim \mathcal{N}(\boldsymbol{\mu}_t,\sigma_t^2\mathbf{I})$.
5. Calculate the relation embedding for this semantic component: $\boldsymbol{\mu}_{r,n} = \mathbf{t}-\mathbf{h}$.

Finally, the score function is

$$f(h,r,t) \propto \sum_{n=1}^{N_r} \pi_{r,n}\mathcal{N}(\boldsymbol{\mu}_{r,n};\boldsymbol{\mu}_t-\boldsymbol{\mu}_h,(\sigma_h^2+\sigma_t^2)\mathbf{I}), \tag{9.35}$$

in which N_r is the number of semantic components of the relation r.

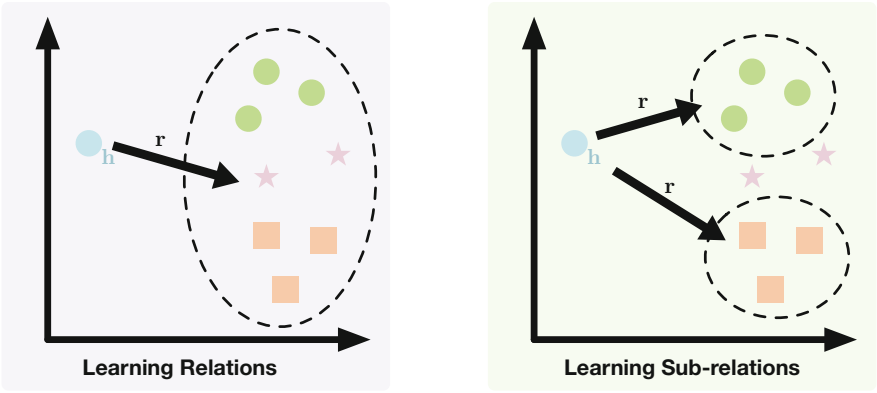

Fig. 9.10 The architecture of TransG [174]. The figure is redrawn according to Fig. 2 from TransG paper [174]

9.3.3 Neural Representation

With the development of neural networks, several efforts have been devoted to exploring neural networks for modeling KGs. Next, we will introduce how to represent KGs with neural networks.

Single Layer Model (SLM) Inspired by the previous works in representing KGs, SLM represents both entities and relations in low-dimensional spaces, and uses relation-specific matrices to project entities to relation spaces. Similar to the linear method SE, the score function of SLM is

$$f(h, r, t) = \mathbf{r}^\top \tanh(\mathbf{M}_{r,1}\mathbf{h} + \mathbf{M}_{r,2}\mathbf{t}), \tag{9.36}$$

where $\mathbf{h}, \mathbf{t} \in \mathbb{R}^{d_e}$ represent head and tail embeddings, $\mathbf{r} \in \mathbb{R}^{d_r}$ represents relation embedding, and $\mathbf{M}_{r,1}, \mathbf{M}_{r,2} \in \mathbb{R}^{d_e \times d_r}$ stand for the relation-specific matrices.

Neural Tensor Network (NTN) Although SLM has introduced relation embeddings as well as a nonlinear neural layer to build the score function, the representation capability is still restricted. NTN [143] is then proposed by introducing tensors into the SLM framework, which can be seen as an enhanced version of SLM. Besides the original linear neural network layer that projects entities to the relation space, NTN adds another tensor-based neural layer which combines head and tail embeddings with a relation-specific tensor. The score function of NTN is

$$f(h, r, t) = \mathbf{r}^\top \tanh(\mathbf{h}^\top \overrightarrow{\mathbf{M}}_r \mathbf{t} + \mathbf{M}_{r,1}\mathbf{h} + \mathbf{M}_{r,2}\mathbf{t} + \mathbf{b}_r), \tag{9.37}$$

where $\overrightarrow{\mathbf{M}}_r \in \mathbb{R}^{d_e \times d_e \times d_r}$ is a three-way relation-specific tensor, $\mathbf{b}_r$ is the bias, and $\mathbf{M}_{r,1}, \mathbf{M}_{r,2} \in \mathbb{R}^{d_e \times d_r}$ are the relation-specific matrices. Note that SLM can be seen as a simplified version of NTN if the tensor and the bias are set to zero.

Besides improving the score function, NTN also attempts to utilize the latent textual information located in entity names and successfully achieves significant improvements. Differing from previous methods that provide each entity with a vector, NTN represents each entity as the average of its entity name's word embeddings. For example, the entity *Bengal tiger* will be represented as the average word embeddings of *Bengal* and *tiger*. It is apparent that the entity name will provide valuable information for understanding an entity, since *Bengal tiger* may come from Bengal and be related to other tigers.

NTN utilizes tensor-based neural networks to model triplet facts and achieves excellent success. However, the overcomplicated method leads to high computational complexity compared to other methods, and the vast number of parameters limits the performance on sparse and large-scale KGs.

Neural Association Model (NAM) NAM [94] adopts multilayer nonlinear activation to model relations. More specifically, two structures are used by NAM to represent KGs: deep neural network (DNN) and relation modulated neural network (RMNN).

NAM-DNN adopts a MLP with L layers to operate knowledge embeddings:

$$\mathbf{z}^k = \text{Sigmoid}(\mathbf{M}^k \mathbf{z}^{k-1} + \mathbf{b}^k), \quad k = 1, \cdots, L, \tag{9.38}$$

where $\mathbf{z}^0 = [\mathbf{h}; \mathbf{r}]$ is the concatenation of $\mathbf{h}$ and $\mathbf{r}$, $\mathbf{M}^k$ is the weight matrix of the k-th layer, and $\mathbf{b}^k$ is the bias vector of the k-th layer. Finally, NAM-DNN defines the score function as

$$f(h, r, t) = \text{Sigmoid}(\mathbf{t}^\top \mathbf{z}^L). \tag{9.39}$$

As compared with NAM-DNN, NAM-RMNN additionally feeds the relation embedding $\mathbf{r}$ into the model:

$$\mathbf{z}^k = \text{Sigmoid}(\mathbf{M}^k \mathbf{z}^{k-1} + \mathbf{B}^k \mathbf{r}), \quad k = 1, \cdots, L, \tag{9.40}$$

where $\mathbf{M}^k$ and $\mathbf{B}^k$ indicate the weight and bias matrices. Finally, NAM-RMNN defines the score function as

$$f(h, r, t) = \text{Sigmoid}(\mathbf{t}^\top \mathbf{z}^L + \mathbf{B}^{L+1} \mathbf{r}). \tag{9.41}$$

Convolutional 2D Embeddings (ConvE) ConvE [32] uses 2D convolutional operations over embeddings to model KGs. Specifically, ConvE uses convolutional and fully connected layers to model interactions between entities and relations. After that, the obtained features are flattened and transformed by a fully connected layer,

and the inner product between the final feature and the tail entity embeddings is used to build the score function:

$$f(h, r, t) = N\left(\text{vec}(N([\bar{\mathbf{h}}; \bar{\mathbf{r}}] * \omega))\mathbf{W}\right) \cdot \mathbf{t}, \tag{9.42}$$

where $[\bar{\mathbf{h}}; \bar{\mathbf{r}}]$ is the concatenation of $\bar{\mathbf{h}}$ and $\bar{\mathbf{r}}$, $N(\cdot)$ is a neural layer, $*$ denotes the convolution operator, and vec$(\cdot)$ means compressing a matrix into a vector. $\bar{\mathbf{h}}$ and $\bar{\mathbf{r}}$ denote the 2D-reshaping versions of $\mathbf{h}$ and $\mathbf{r}$, respectively: if $\mathbf{h}, \mathbf{r} \in \mathbb{R}^d$, then $\bar{\mathbf{h}}, \bar{\mathbf{r}} \in \mathbb{R}^{d_a \times d_b}$, where $d = d_a d_b$.

To some extent, ConvE can be seen as an improvement model based on HolE. Compared with HolE, ConvE adopts multiple neural layers to learn nonlinear features and is thus more expressive than HolE.

Relational Graph Convolutional Networks (RGCN) RGCN [136] is an extension of GCNs to model KGs. The core idea of RGCN is to formalize modeling KGs as message passing. Therefore, in RGCN, the representations of entities and relations are the results of information propagation and fusion at multiple layers. Specifically, given an entity h, its embedding at the $(k+1)$-th layer is

$$\mathbf{h}^{k+1} = \text{Sigmoid}\left(\sum_{r \in \mathcal{R}} \sum_{t \in \mathcal{N}_h^r} \frac{1}{c_h^r} \mathbf{W}_r^k \mathbf{t}^k + \tilde{\mathbf{W}}^k \mathbf{t}^k\right), \tag{9.43}$$

where $\mathcal{N}_h^r$ denotes the neighbor set of h under the relation r and c_h^r is the normalization factor. c_h^r can be either learned or preset, and normally $c_h^r = |\mathcal{N}_h^r|$.

Note that RGCN only aims to obtain more expressive features for entities and relations. Therefore, based on the output features of RGCN, any score function mentioned above can be used here, such as combining the features of RGCN and the score function of TransE to learn knowledge representations.

9.3.4 Manifold Representation

So far, we have introduced linear methods, translation methods, and neural methods for knowledge representation. All these methods project entities and relations into low-dimensional embedding spaces, and seek to improve the flexibility and variety of entity and relation representations. Although these methods have achieved promising results, they assume that the geometry of the embedding spaces for entities and relations are all Euclidean. However, the basic Euclidean geometry may not be the optimal geometry to model the complex structure of KGs. Next, we will introduce several typical manifold methods that aim to use more flexible and powerful geometric spaces to carry representations.

ManifoldE ManifoldE [173] considers the possible positions of golden candidates for representations in spaces as a manifold rather than a point. The overall score function of ManifoldE is

$$f(h, r, t) = -\|M(h, r, t) - D_r^2\|, \tag{9.44}$$

in which D_r^2 is a relation-specific manifold parameter. Two kinds of manifolds are then proposed in ManifoldE. ManifoldE-Sphere is a straightforward manifold that supposes $\mathbf{t}$ should be located in the sphere which has $\mathbf{h} + \mathbf{r}$ to be the center and D_r to be the radius. We have:

$$M(h, r, t) = \|\mathbf{h} + \mathbf{r} - \mathbf{t}\|. \tag{9.45}$$

A tail may correspond to many different head-relation pairs, and the manifold assumption requires that the tail lays in all the manifolds of these head-relation pairs, i.e., lays in the intersection of these manifolds. However, two spheres can only intersect only under some strict conditions. Therefore, the hyperplane is utilized because it is easier for two hyperplanes to intersect. The function of ManifoldE-Hyperplane is

$$M(h, r, t) = (\mathbf{h} + \mathbf{r}_h)^\top (\mathbf{t} + \mathbf{r}_t), \tag{9.46}$$

in which $\mathbf{r}_h$ and $\mathbf{r}_t$ represent the two entity-specific embeddings of the relation r. This indicates that for a triplet $\langle h, r, t\rangle$, the tail entity $\mathbf{t}$ should locate in the hyperplane whose normal vector is $\mathbf{h} + \mathbf{r}_h$ and intercept is D_r^2. Furthermore, ManifoldE-Hyperplane considers absolute values in $M(h, r, t)$ as $|\mathbf{h}+\mathbf{r}_h|^\top |\mathbf{t}+\mathbf{r}_t|$ to double the solution number of possible tail entities. For both manifolds, ManifoldE applies a kernel form on the reproducing kernel Hilbert space.

ComplEx ComplEx [153] employs an eigenvalue decomposition model which makes use of complex-valued embeddings, i.e., $\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{C}^d$. Complex embeddings can well handle binary relations, such as the symmetric and antisymmetric relations. The score function of ComplEx is

$$\begin{aligned} f(h, r, t) &= \mathrm{Re}(\langle \mathbf{r}, \mathbf{h}, \mathbf{t}\rangle) \\ &= \langle \mathrm{Re}(\mathbf{r}), \mathrm{Re}(\mathbf{h}), \mathrm{Re}(\mathbf{t})\rangle + \langle \mathrm{Re}(\mathbf{r}), \mathrm{Im}(\mathbf{h}), \mathrm{Im}(\mathbf{t})\rangle \\ &\quad - \langle \mathrm{Im}(\mathbf{r}), \mathrm{Re}(\mathbf{h}), \mathrm{Im}(\mathbf{t})\rangle - \langle \mathrm{Im}(\mathbf{r}), \mathrm{Im}(\mathbf{h}), \mathrm{Re}(\mathbf{t})\rangle, \end{aligned}$$

where $\langle \mathbf{x}, \mathbf{y}, \mathbf{z}\rangle = \sum_i x_i y_i z_i$ denotes the trilinear dot product, $\mathrm{Re}(x)$ is the real part of x, and $\mathrm{Im}(x)$ is the imaginary part of x. In fact, ComplEx can be viewed as a generalization of RESCAL that uses complex embeddings to model KGs.

RotatE Similar to ComplEx, RotatE [149] also represents KGs with complex-valued embeddings. RotatE defines relations as rotations from head entities to tail entities, which makes it easier to learn various relation patterns such as symmetry,

antisymmetry, inversion, and composition. The element-wise (Hadamard) product can naturally represent the rotation process in the complex-valued space. Therefore, the score function of RotatE is

$$f(h, r, t) = -\|\mathbf{h} \odot \mathbf{r} - \mathbf{t}\|, \tag{9.47}$$

where $\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{C}^d$ and $\odot$ denotes the element-wise (Hadamard) product. RotatE is simple and achieves quite good performance. Compared with previous methods, it is the first model that is theoretically able to model all the above four patterns (symmetry, antisymmetry, inversion, and composition). On the basis of RotatE, Zhang et al. [203] further introduce hypercomplex spaces to represent entities and relations, and achieves better performance.

MuRP MuRP [4] proposes to embed the entities in the hyperbolic space since hyperbolic space is shown to be more suitable to represent hierarchical data than Euclidean space. Specifically, they embed the entities to the Poincaré model [130] (a typical geometric model in hyperbolic space), and exploit the Mobiüs transformations in the Poincaré model as the alternatives to vector-matrix multiplication and vector addition in Euclidean space. The score function of MuRP is

$$\begin{aligned} f(h, r, t) &= d_{\mathbb{P}}(\mathbf{h}^{(r)}, \mathbf{t}^{(r)})^2 - b_h - b_t \\ &= d_{\mathbb{P}}(\exp_{\mathbf{0}}^c(\mathbf{M}_r \log_{\mathbf{0}}^c(\mathbf{h})), \mathbf{t} \oplus \mathbf{r}) - b_h - b_t, \end{aligned} \tag{9.48}$$

where $d_{\mathbb{P}}(\cdot, \cdot)$ calculates the distance between two points in the Poincaré model, $\mathbf{M}_r$ is the transform matrix for the relation r, $\mathbf{r}$ is the translation vector of the relation r, and b_h and b_t are biases for the head and tail entities respectively. $\exp_{\mathbf{0}}^c$ is the exponential mapping at $\mathbf{0}$ in the Poincaré model of the curvature c, and it maps points in the tangent space at $\mathbf{0}$ (an Euclidean subspace) to the Poincaré model. $\log_{\mathbf{0}}^c$ is the logarithmic mapping at $\mathbf{0}$ in the Poincaré model of the curvature c, and is the inverse mapping for $\exp_{\mathbf{0}}^c$. MuRP with a dimension as low as 40 achieves comparable results to the Euclidean models with dimension greater than 100, showing the effectiveness of hyperbolic space in encoding relational knowledge.

HyboNet HyboNet [23] argues that previous hyperbolic methods such as MuRP only introduce the hyperbolic geometric for embeddings, but still perform linear transformations in tangent spaces (Euclidean subspaces), significantly limiting the capability of hyperbolic models. Inspired by the Lorentz transformation in Physics, HyboNet proposes a linear transformation in the Lorentz model [130] (another typical geometric model to build hyperbolic spaces) to avoid the introduction of exponential mapping and logarithmic mapping when transforming embeddings, significantly speeding up the network and stabilizing the computation. The score function of HyboNet is

$$f(h, r, t) = d_{\mathbb{L}}^2(g_r(\mathbf{h}), \mathbf{t}) - b_h - b_t - \delta, \tag{9.49}$$

where $d_{\mathbb{L}}^2$ is the squared Lorentzian distance between two points in Lorentz model, g_r is the relation-specific Lorentz linear transformation, b_h, b_t are the biases for the head and tail entities, respectively, and δ is a hyper-parameter used to make the training process more stable.

9.3.5 Contextualized Representation

We live in a complicated pluralistic real world where we can get information from different senses. Due to this, we can learn knowledge not only from structured KGs but also from text, schemas, images, and rules. Despite the massive size of existing KGs, there is a large amount of knowledge in the real world that may not be included in the KGs. Integrating multisource information provides a novel approach for learning knowledge representations not only from the internal structured information of KGs but also from other external information. Moreover, exploring multisource information can help further understand human cognition with different senses in the real world. Next, we will introduce typical methods that utilize multisource information to enhance knowledge representations.

Knowledge Representation with Text Textual information is one of the most common and widely used information for knowledge representation. Wang et al. [164] attempt to utilize textual information by jointly learning representations of entities, relations, and words within the same low-dimensional embedding space. The method contains three parts: the knowledge model, the text model, and the alignment model. The knowledge model is learned on the triplets of KGs using TransE, while the text model is learned on the text using skip-gram. As for the alignment model, two methods are proposed to align entity and word representations by utilizing Wikipedia anchors and entity names, respectively.

Modeling entities and words into the same embedding space has the merit of encoding the information in both KGs and plain text in a unified semantic space. However, Wang's joint model mainly depends on the completeness of Wikipedia anchors and suffers from the ambiguities of many entity names. To address these issues, Zhong et al. [207] further improve the alignment model with entity descriptions, assuming that entities should have similar semantics to their corresponding descriptions.

Different from the above joint models that merely consider the alignments between KGs and textual information, description-embodied knowledge representation learning (DKRL) [176] can directly build knowledge representations from entity descriptions. Specifically, DKRL provides two kinds of knowledge representations: for each entity h, the first is the structure-based representation $\mathbf{h}_S$, which can be learned based on the structure of KGs, and the second is the

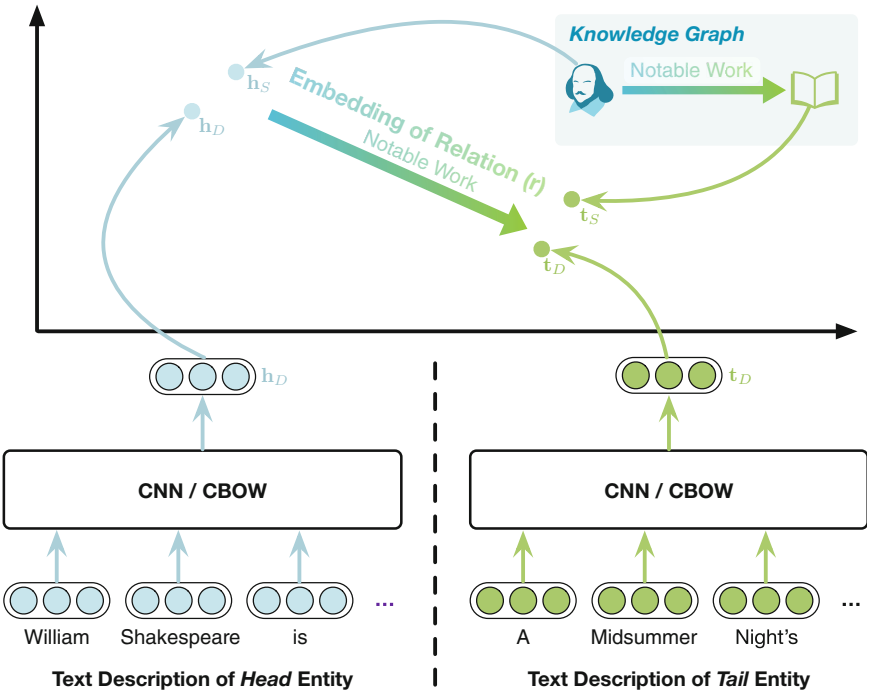

Fig. 9.11 The architecture of DKRL [176]. The figure is redrawn according to Fig. 3 from DKRL paper [176]

description-based representation $\mathbf{h}_D$ that derives from its description. The score function of DKRL derives from translation methods, and we have:

$$f(h,r,t) = -(\|\mathbf{h}_S + \mathbf{r} - \mathbf{t}_S\| + \|\mathbf{h}_S + \mathbf{r} - \mathbf{t}_D\| + \|\mathbf{h}_D + \mathbf{r} - \mathbf{t}_S\| + \|\mathbf{h}_D + \mathbf{r} - \mathbf{t}_D\|). \tag{9.50}$$

As shown in Fig. 9.11, the description-based representations are constructed via CBOW or CNNs that can encode rich textual information from plain text into representations.

Compared with conventional non-contextualized methods, the representations learned by DKRL are built with both structured and textual information and thus could perform better. Besides, DKRL can represent an entity even if it is not in the training set as long as there are a few sentences to describe this entity. Therefore, with millions of new entities emerging every day, DKRL can handle these new entities based on the setting of zero-shot learning.

Knowledge Representation with Types Entity types, as hierarchical schemas, can provide rich structured information to understand entities. Generally, there are

two paths for using entity types for knowledge representations: type-constrained methods and type-augmented methods.

Type-Constrained Methods Krompaß et al. [78] take type information as constraints to improve existing methods like RESCAL and TransE via type constraints. It is intuitive that for a particular relation, the head and tail entities associated with this relation can only be of some specific types. For example, the head entity of the relation `Writes Books` should be a person (more precisely, an author), and the tail entity should be a book.

With type constraints, in RESCAL, the original factorization $\mathbf{X}_n \approx \mathbf{E}\mathbf{M}_{r_n}\mathbf{E}^\top$ in Eq.(9.8) can be modified to

$$\mathbf{X'}_n \approx \mathbf{E}_{[\mathcal{H}_{r_n},:]}\mathbf{M}_{r_n}\mathbf{E}^\top_{[\mathcal{T}_{r_n},:]}, \tag{9.51}$$

where $\mathcal{H}_{r_n}$, $\mathcal{T}_{r_n}$ are the entity sets fitting the type constraints of the n-th relation r_n in $\mathcal{R}$, and $\mathbf{X'}_n$ is a sparse adjacency matrix of the shape $|\mathcal{H}_{r_n}| \times |\mathcal{T}_{r_n}|$. Intuitively, only the entities that fit type constraints will be considered during the factorization process.

With type constraints, in TransE, negative samples with higher quality can be generated. Learning knowledge representations need negative samples, and negative samples are often generated by randomly replacing triplets' head or tail entities. Given a triplet $\langle h, r, t\rangle$, with type constraints, its negative samples $\langle \tilde{h}, \tilde{r}, \tilde{t}\rangle$ need to satisfy

$$\tilde{h} \in \mathcal{H}_r \subseteq \mathcal{E}, \quad \tilde{t} \in \mathcal{T}_r \subseteq \mathcal{E}. \tag{9.52}$$

Intuitively, for an entity whose type does not match the relation r, it will not be used to construct negative samples. The negative samples constructed with type constraints are more confusing, which is beneficial for learning more robust and effective representations.

Type-Augmented Methods In addition to the simplicity and effectiveness of using the type information as constraints, the representation can be further enhanced by using the type information directly as additional information in the learning. Instead of merely viewing type information as type constraints, type-embodied knowledge representation learning (TKRL) is proposed [177], utilizing hierarchical type structures to instruct the construction of projection matrices. Inspired by TransR that every entity should have multiple representations in different relation spaces, the score function of TKRL is

$$f(h, r, t) = -\|\mathbf{M}_{rh}\mathbf{h} + \mathbf{r} - \mathbf{M}_{rt}\mathbf{t}\|, \tag{9.53}$$

in which $\mathbf{M}_{rh}$ and $\mathbf{M}_{rt}$ are two projection matrices for h and t that depend on their corresponding hierarchical types in this triplet. Two hierarchical encoders are proposed to learn the above projection matrices, regarding all sub-types in the hierarchy as projection matrices, where the recursive hierarchical encoder (RHE) is

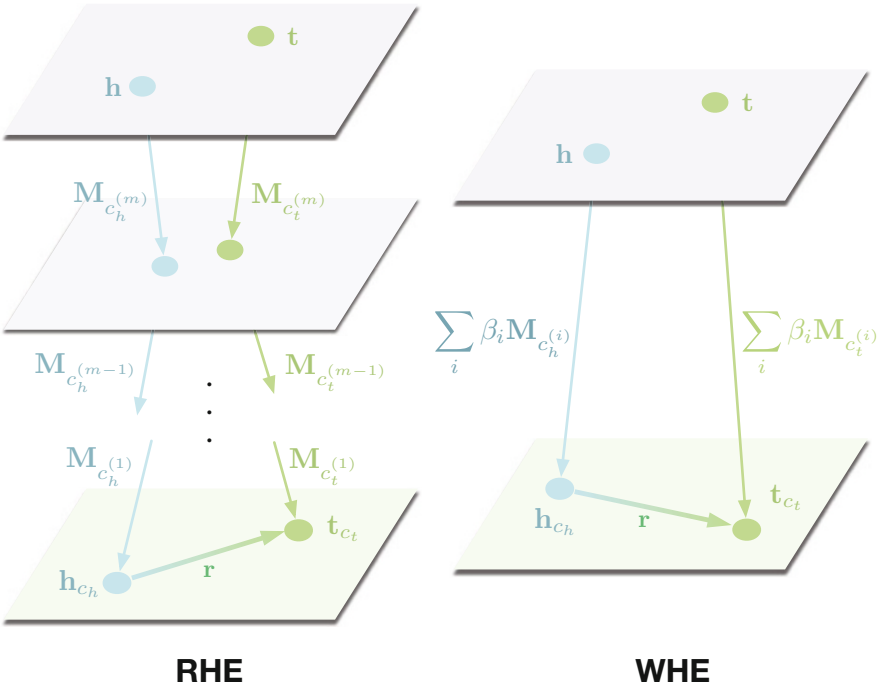

Fig. 9.12 The architecture of TKRL [177]. The figure is redrawn according to Fig. 2 from TKDL paper [177]

based on the matrix multiplication operation, and the weighted hierarchical encoder (WHE) is based on the matrix summation operation.

Figure 9.12 shows a simple illustration of TKRL. Taking a type hierarchy c with m layers for instance, $c^{(i)}$ is the i-th sub-type. Considering the sub-type at the first layer is the most precise and the sub-type at the last layer is the most general, TKRL can get type-specific entity representations at different granularities following the hierarchical structure, and the projection matrices can be formalized as

$$\begin{aligned} \mathbf{M}_{RHE_c} &= \prod_{i=1}^{m} \mathbf{M}_{c^{(i)}} = \mathbf{M}_{c^{(1)}} \mathbf{M}_{c^{(2)}} \dots \mathbf{M}_{c^{(m)}}, \\ \mathbf{M}_{WHE_c} &= \sum_{i=1}^{m} \beta_i \mathbf{M}_{c^{(i)}} = \beta_1 \mathbf{M}_{c^{(1)}} + \dots + \beta_m \mathbf{M}_{c^{(m)}}, \end{aligned} \tag{9.54}$$

where $\mathbf{M}_{c^{(i)}}$ stands for the projection matrix of the i-th sub-type of the hierarchical type c and β_i is the corresponding weight of the sub-type. Taking RHE as an example, given the entity *William Shakespeare*, it is first projected to a general sub-

Fig. 9.13 Examples of entity images. These examples come from the original paper of TKRL [175]. All these images come from ImageNet [31]

type space like *human* and then sequentially projected to a more precise sub-type like *author* or *English author*.

Knowledge Representation with Images Human cognition is highly related to the visual information of objects in the real world. For entities in KGs, their corresponding images can provide intuitive visual information about their appearance, which may give important hints about some attributes of the entities. For instance, Fig. 9.13 shows the images of *Suit of armour* and *Armet*. From these images, we can easily infer the fact $\langle$*Suit of armour*, `Has a Part`, *Armet*$\rangle$ directly.

Image-embodied knowledge representation learning (IKRL) [175] is proposed to consider visual information when learning knowledge representations. Inspired by the abovementioned DKRL, for each entity h, IKRL also proposes the image-based representation $\mathbf{h}_I$ besides the original structure-based representation $\mathbf{h}_S$, and jointly learns these entity representations simultaneously within the translation-based framework:

$$f(h,r,t) = -(\|\mathbf{h}_S+\mathbf{r}-\mathbf{t}_S\|+\|\mathbf{h}_S+\mathbf{r}-\mathbf{t}_I\|+\|\mathbf{h}_I+\mathbf{r}-\mathbf{t}_S\|+\|\mathbf{h}_I+\mathbf{r}-\mathbf{t}_I\|). \quad (9.55)$$

More specifically, IKRL uses CNNs to obtain the representations of all entity images, and then uses a matrix to project image representations from the image embedding space to the entity embedding space. Since one entity may have multiple images, IKRL uses an attention-based method to highlight those most informative images. IKRL not only shows the importance of visual information for representing entities but also shows the possibility of finding a unified space to represent heterogeneous and multimodal information.

Knowledge Representation with Logic Rules Typical KGs store knowledge in the form of triplets with one relation linking two entities. Most existing knowledge representation methods only consider the information of triplets independently, ignoring the possible interactions and relations between different triplets. Logic rules, which are certain kinds of summaries derived from human prior knowledge, could help us with knowledge reasoning. For instance, given the triplet ⟨ *Beijing*, `Capital of`, *China*⟩, we can easily infer the triplet ⟨*Beijing*, `Located in`, *China*⟩ with high confidence, since we know the logic rule `Capital of` ⇒ `Located in`. To this end, various efforts have been devoted to exploring logic rules for KGs [5, 127, 162]. Here we introduce a typical translation method that jointly learns knowledge representations and logic rules – KALE [52]. KALE can rank all possible logic rules based on the results pre-trained by TransE, and then manually filter useful rules to improve knowledge representations.

The joint learning of KALE consists of two parts: triplet modeling and rule modeling. For the triplet modeling, KALE defines its score function following the translation assumption as

$$f(h,r,t) = 1 - \frac{1}{3\sqrt{d}}\|\mathbf{h}+\mathbf{r}-\mathbf{t}\|, \tag{9.56}$$

in which d stands for the dimension of knowledge embeddings. $f(h,r,t)$ takes a value in $[0,1]$, aiming to map discrete Boolean values (false or true) into a continuous space ($[0,1]$). For the rule modeling, KALE uses the t-norm fuzzy logics [55] that compute the truth value of a complex formula from the truth values of its constituents. Especially, KALE focuses on two typical types of logic rules. The first rule is $\forall h,t : \langle h,r_1,t\rangle \Rightarrow \langle h,r_2,t\rangle$ (e.g., given ⟨*Beijing*, `Capital of`, *China*⟩, we can infer that ⟨*Beijing*, `Located in`, *China*⟩). KALE represents the score function of this logic rule l_1 via specific t-norm logical connectives as

$$f(l_1) = f(h,r_1,t)f(h,r_2,t) - f(h,r_1,t) + 1. \tag{9.57}$$

The second rule is $\forall h,e,t : \langle h,r_1,e\rangle \wedge \langle e,r_2,t\rangle \Rightarrow \langle h,r_3,t\rangle$ (e.g., given ⟨*Tsinghua*, `Located in`, *Beijing*⟩ and ⟨*Beijing*, `Located in`, *China*⟩, we can infer that ⟨*Tsinghua*, `Located in`, *China*⟩). And the second score function is defined as

$$f(l_2) = f(h,r_1,e)f(e,r_2,t)f(h,r_3,t) - f(h,r_1,e)f(e,r_2,t) + 1. \tag{9.58}$$

The joint training contains all positive formulas, including triplet facts and logic rules. Note that for the consideration of rule qualities, KALE ranks all possible logic rules by their truth values with pre-trained TransE and manually filters some rules.

9.3.6 Summary

Knowledge representation learning is the cornerstone of applying knowledge for NLP tasks. Knowledge can be incorporated into NLP tasks in a high-quality manner only with good knowledge representations. In this section, we introduce five directions of existing efforts to obtain distributed knowledge representations: (1) *linear methods*, where relations are represented as linear transformations between entities, (2) *translation methods*, where relations are represented as additive translations between entities, (3) *neural methods*, where neural networks parameterize the interactions between entities and relations, (4) *manifold methods*, where representations are learned in more flexible and powerful geometric spaces instead of the basic Euclidean geometry, and (5) *contextualized methods*, where representations are learned under complex contexts.

In summary, from simple methods like SE and TransE, to more sophisticated methods that use neural networks (e.g., ConvE), the hyperbolic geometry (e.g., HyboNet), and textual information (e.g., DKRL), all these methods can provide effective knowledge representations. These methods lay a solid foundation for further knowledge-guided NLP and knowledge acquisition, which will be introduced in later sections. Note that more sophisticated methods do not necessarily lead to a better application in NLP tasks. Researchers still need to choose the appropriate knowledge representation learning method according to the characteristics of specific tasks and the balance between computational efficiency and representation quality.

9.4 Knowledge-Guided NLP

An effective NLP agent is expected to accurately and deeply understand user demands, and appropriately and flexibly give responses and solutions. Such kind of work can only be done supported by certain forms of knowledge. To this end, knowledge-guided NLP has been widely explored in recent years. Figure 9.14 shows a brief pipeline of utilizing knowledge for NLP tasks. In this pipeline, we first need to extract knowledge from heterogeneous data sources and store the extracted knowledge with knowledge systems (e.g., KGs). Next, we need to project knowledge systems into low-dimensional continuous spaces with knowledge representation learning methods to manipulate the knowledge in a machine-friendly way. Finally, informative knowledge representations can be applied to handle various NLP tasks. After introducing how to learn knowledge representations, we will detailedly show in this section how to use knowledge representations for specific NLP tasks.

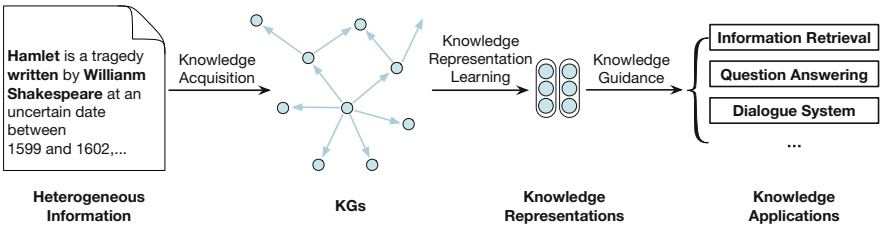

Fig. 9.14 The pipeline of utilizing knowledge for NLP tasks

The performance of NLP models (more generally, machine learning models) depends on four critical factors: input data, model architecture, learning objective, and hypothesis space. And the whole goal is to minimize the structural risk

$$\min_{f \in \mathcal{F}} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(y_i, f(x_i)) + \lambda \mathcal{J}(f), \tag{9.59}$$

where x_i is the input data, f is the model function, $\mathcal{L}$ is the learning objective, $\mathcal{F}$ is the hypothesis space, and $\mathcal{J}(f)$ is the regularization term. By applying knowledge to each of these four factors, we can form four directions to perform knowledge-guided NLP: (1) knowledge augmentation, which aims to augment the input data x_i with knowledge; (2) knowledge reformulation, which aims to reformulate the model function f with knowledge; (3) knowledge regularization, which aims to regularize or modify the learning objectives $\mathcal{L}$ with knowledge; (4) knowledge transfer, which aims to transfer the pre-trained parameters as prior knowledge to constrain the hypothesis space $\mathcal{F}$.

Some works [60, 61] have briefly introduced this knowledge-guided NLP framework, while in this section, we will further present more details around the four knowledge-guided directions. In addition, to make this section clearer and more intuitive, we will also introduce some specific application cases of knowledge-guided NLP.

9.4.1 Knowledge Augmentation

Knowledge augmentation aims at using knowledge to augment the input features of models. Formally, after using knowledge k to augment the input, the original risk function is changed to

$$\min_{f \in \mathcal{F}} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(y_i, f(x_i, k)) + \lambda \mathcal{J}(f). \tag{9.60}$$

In order to achieve this kind of knowledge augmentation at the input level, existing efforts focus on adopting two mainstream approaches.

Augmentation with Knowledge Context One approach is to directly add knowledge to the input as additional context. Augmenting language modeling with retrieval is a representative method, such as REALM [53] and RAG [86]. These methods retrieve background knowledge from additional corpora and then use the retrieved knowledge to provide more information for language modeling. Since the retrieved knowledge can significantly improve the performance of language understanding and generation, this approach to achieving knowledge augmentation is widely applied by question answering [76, 111] and dialogue systems [139, 168]. Next, we will take RAG as an example to show how to perform knowledge augmentation with knowledge context.

Example: Knowledge Augmentation for the Generation of PTMs In recent years, PTMs have achieved state-of-the-art results on a variety of NLP tasks, but these PTMs still face challenges in precisely accessing and manipulating knowledge and cannot well handle various knowledge-intensive tasks, especially for various text generation tasks that require extensive knowledge. To help PTMs utilize more knowledge for text generation, retrieval-augmented generation (RAG) [86] has been proposed with the aim of using the retrieved external knowledge as additional context to generate text with higher quality.

Given the input sequence x to generate the output sequence y, the overall process of the typical autoregressive generation method can be formalized as $P(y|x) = \prod_{i=1}^{N} P_\theta(y_i|x, y_{1:i-1})$, where θ is the parameters of the generator, N is the length of y, and y_i is the i-th token of y. To use more knowledge to generate y, RAG first retrieves the external information z according to the input x and then generates the output sequence y based on both x and z. To ensure that the retrieved contents can cover the crucial knowledge required to generate y, the top-K contents retrieved by the retriever are all used to help generate the output sequence y, and thus the overall generation process is

$$\begin{aligned} P_{\text{RAG-Sequence}}(y|x) &\approx \sum_{z \in \text{top}-K[P_\eta(\cdot|x)]} P_\eta(z|x) P_\theta(y|x, z) \\ &= \sum_{z \in \text{top-}K[P_\eta(\cdot|x)]} P_\eta(z|x) \prod_{i=1}^{N} P_\theta(y_i|x, z, y_{1:i-1}), \end{aligned} \tag{9.61}$$

where η is the parameters of the retriever.

In addition to applying knowledge augmentation at the sequence level, token-level RAG is also introduced to provide finer-grained augmentation. Specifically, token-level RAG first retrieves the top K external information according to the input x, which is the same as RAG-Sequence. When generating text, token-level RAG considers all the retrieved information together to generate the distribution for the next output token, instead of sequence-level RAG which separately generates

sequences based on the retrieved content and then merges the generated sequences. Formally, the token-level RAG is

$$P_{\text{RAG-Token}}(y|x) \approx \prod_{i=1}^{N} \sum_{z \in \text{top-}K[P(\cdot|x)]} P_\eta(z|x) P_\theta(y_i|x, z, y_{1:i-1}). \tag{9.62}$$

To sum up, RAG adds the retrieved knowledge to the input as additional context, which is a typical example of knowledge augmentation with knowledge context.

Augmentation with Knowledge Embeddings Another approach is to design special modules to fuse the original input features and knowledge embeddings and then use the knowledgeable features as the input to solve NLP tasks. Since this approach can help to fully utilize heterogeneous knowledge from multiple sources, many works follow this approach to integrate unstructured text and structured symbolic knowledge in KGs, leading to knowledge-guided information retrieval [87, 100] and knowledge-guided PTMs [96, 124, 128, 163, 185, 205]. Next, we will first introduce word-entity duet, an effective information retrieval method, and then take a typical knowledge-guided information retrieval method EDRM as an example to show how to perform knowledge augmentation with knowledge embeddings.

Example: Knowledge Augmentation for Information Retrieval Information retrieval focuses on obtaining informative representations of queries and documents, and then designing effective metrics to compute the similarities between queries and documents. The emergence of large-scale KGs has motivated the development of entity-oriented information retrieval, which aims to leverage KGs to improve the retrieval process. Word-entity duet [179] is a typical method for entity-oriented information retrieval. Specifically, given a query q and a document d, word-entity duet first constructs bag-of-words q^w and d^w. By annotating the entities mentioned by the query q and the document d, word-entity duet then constructs bag-of-entities q^e and d^e. Based on bag-of-words and bag-of-entities, word-entity duet utilizes the duet representations of bag-of-words and bag-of-entities to match the query q and the document d. The word-entity duet method consists of a four-way interaction: query words to document words (q^w-d^w), query words to document entities (q^w-d^e), query entities to document words (q^e-d^w), and query entities to document entities (q^e-d^e).

On the basis of the word-entity duet method, EDRM [100] further uses distributed representations instead of bag-of-words and bag-of-entities to represent queries and documents for ranking. As shown in Fig. 9.15, EDRM first learns the distributed representations of entities according to entity-related information in KGs, such as entity descriptions and entity types. Then, EDRM uses interaction-based neural models [28] to match the query and documents with word-entity duet distributed representations. More specifically, EDRM uses a translation layer that calculates the similarity between query-document terms: ($\mathbf{v}^i_{w^q}$ or $\mathbf{v}^i_{e^q}$) and ($\mathbf{v}^j_{w^d}$ or $\mathbf{v}^j_{e^d}$). It constructs the interaction matrix $\mathbf{M} = \{\mathbf{M}_{ww}, \mathbf{M}_{we}, \mathbf{M}_{ew}, \mathbf{M}_{ee}\}$, by

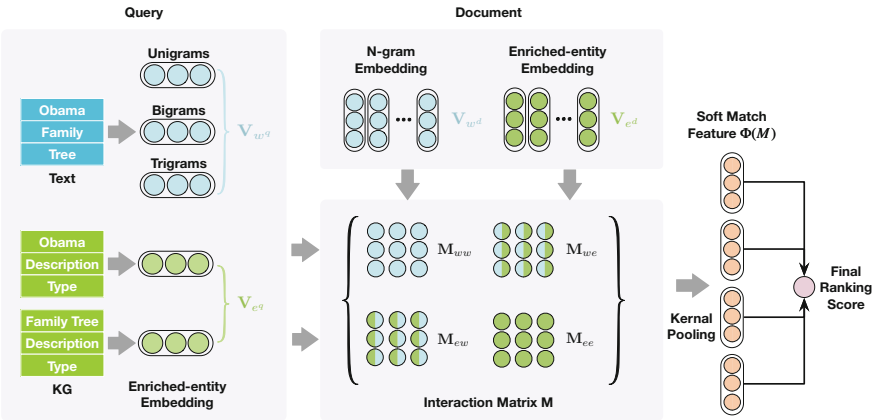

Fig. 9.15 The architecture of EDRM [100]. The figure is redrawn according to Fig. 1 from EDRM paper [100]

denoting $\mathbf{M}_{ww}, \mathbf{M}_{we}, \mathbf{M}_{ew}, \mathbf{M}_{ee}$ as the interactions of q^w-d^w, q^w-d^e, q^e-d^w, q^e-d^e, respectively. And the elements in these matrices are the cosine similarities of corresponding terms:

$$\begin{aligned} \mathbf{M}_{ww}^{ij} &= \cos(\mathbf{v}_{w^q}^i, \mathbf{v}_{w^d}^j); \mathbf{M}_{we}^{ij} = \cos(\mathbf{v}_{w^q}^i, \mathbf{v}_{e^d}^j), \\ \mathbf{M}_{ew}^{ij} &= \cos(\mathbf{v}_{e^q}^i, \mathbf{v}_{w^d}^j); \mathbf{M}_{ee}^{ij} = \cos(\mathbf{v}_{e^q}^i, \mathbf{v}_{e^d}^j). \end{aligned} \tag{9.63}$$

The final ranking feature $\Phi(\mathbf{M})$ is a concatenation of four cross matches:

$$\Phi(\mathbf{M}) = [\phi(\mathbf{M}_{ww}); \phi(\mathbf{M}_{we}); \phi(\mathbf{M}_{ew}); \phi(\mathbf{M}_{ee})], \tag{9.64}$$

where $\phi(\cdot)$ can be any function used in interaction-based neural ranking models, such as using Gaussian kernels to extract the matching feature from the matrix $\mathbf{M}$ and then pool into a feature vector $\phi(\mathbf{M})$. For more details of designing $\phi(\cdot)$ and using $\Phi(\mathbf{M})$ to compute ranking scores, we suggest referring to some typical interaction-based information retrieval models [28, 180].

To sum up, EDRM introduces distributed knowledge representations to improve the representations of queries and documents for information retrieval, which is a typical example of knowledge augmentation with knowledge embeddings.

9.4.2 Knowledge Reformulation

Knowledge reformulation aims at using knowledge to enhance the model processing procedure. Formally, after using knowledge to reformulate the model function, the

original risk function is changed to

$$\min_{f_k \in \mathcal{F}} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(y_i, f_k(x_i)) + \lambda \mathcal{J}(f_k), \tag{9.65}$$

where $f_k(\cdot)$ is the model function reformulated by knowledge. Considering the complexity of the model function $f(\cdot)$, it is difficult for us to comprehensively discuss the construction process of f_k. To introduce this section more clearly and give readers a more intuitive understanding of knowledge reformulation, we here focus on introducing two relatively simple knowledge reformulation scenarios: knowledgeable preprocessing and post-processing.

Knowledgeable Preprocessing On the one hand, we can use the underlying knowledge-guided model layer for preprocessing to make features more informative [160, 167, 194]. Formally, x_i is first input to the function k and then input to the function f as

$$f_k(x_i) = f(k(x_i)), \tag{9.66}$$

where $k(\cdot)$ is the knowledge-guided model function used for preprocessing and $f(\cdot)$ is the original model function. The knowledge-guided attention mechanism is a representative approach that usually leverages informative knowledge representations to enhance model feature processing. Next, we will take two typical knowledge-guided attention mechanisms [58, 178] as examples to show how to use knowledge for model preprocessing.

Example: Knowledge Reformulation for Knowledge Acquisition Knowledge acquisition includes two main approaches. One is knowledge graph completion (KGC), which aims to perform link prediction on KGs. The other is relation extraction (RE) to predict relations between entity pairs based on the sentences containing entity pairs. Formally, given sentences $s_1, s_2, \cdots$ containing the entity pair h, t, RE aims to evaluate the likelihood that a relation r and h, t can form a triplet based on the semantics of these sentences. Different from RE, KGC only uses the representations of h, r, t learned by knowledge representation learning methods to compute the score function $f(h, r, t)$, and the score function serves knowledge acquisition.

Generally, RE and KGC models are learned separately, and these models cannot fully integrate text and knowledge to acquire more knowledge. To this end, Han et al. [58] propose a joint learning framework for knowledge acquisition, which can jointly learn knowledge and text representations within a unified semantic space via KG-text alignments. Figure 9.16 shows the brief framework of the joint model. For the text part, the sentence with two entities (e.g., *Mark Twain* and *Florida*) is regarded as the input to the encoder, and the output is considered to potentially describe specific relations (e.g., `Place of Birth`). For the KG part, entity and relation representations are learned via a knowledge representation learning method

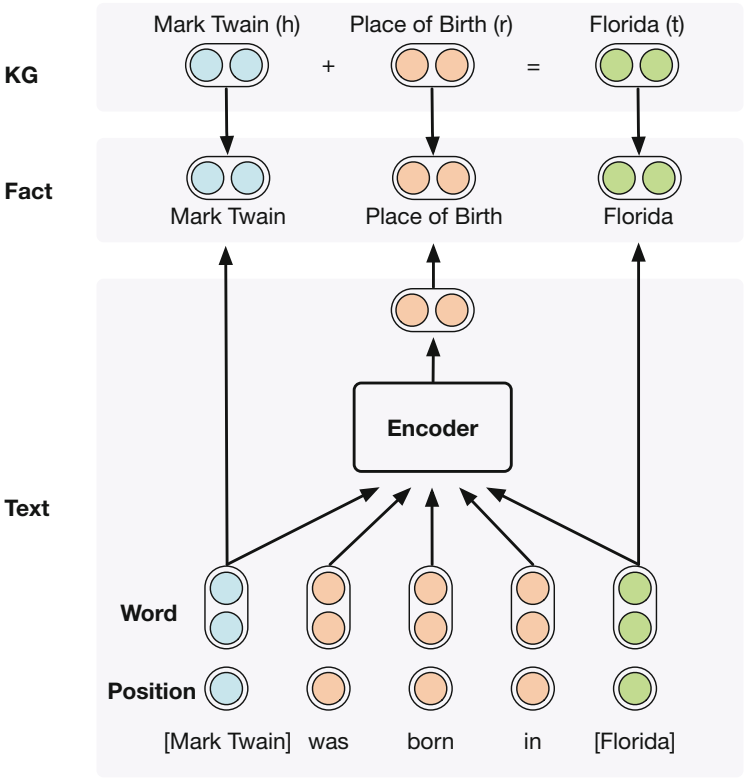

Fig. 9.16 The joint learning framework for knowledge acquisition [58]. The figure is redrawn according to Fig. 1 from the paper of Han et al. [58]

such as TransE. The learned representations of the KG and text parts are aligned during the training process.

Given sentences $\{s_1, s_2, \cdots\}$ containing the same entity pair h, t, not all of these sentences can help predict the relation between h and t. For a given relation r, there are many triplets $\{(h_1, r, t_1), (h_2, r, t_2), \cdots\}$ containing the relation, but not all triplets are important enough for learning the representation of r. Therefore, as shown in Fig. 9.17, Han et al. further adopt mutual attention to reformulate the preprocessing of both the text and knowledge models, to select more useful sentences for RE and more important triplets for KGC. Specifically, we use knowledge representations to highlight the more valuable sentences for predicting the relation between h and t. This process can be formalized as

$$\alpha = \text{Softmax}(\mathbf{r}_{ht}^{\top}\mathbf{W}_{\text{KA}}\mathbf{S}), \quad \hat{\mathbf{s}} = \mathbf{S}\alpha^{\top}, \tag{9.67}$$

where $\mathbf{W}_{\text{KA}}$ is a bilinear matrix of the knowledge-guided attention, $\mathbf{S} = [\mathbf{s}_1, \mathbf{s}_2, \cdots]$ are the hidden states of the sentences $s_1, s_2, \cdots$. $\mathbf{r}_{ht}^{\top}$ is a representation that

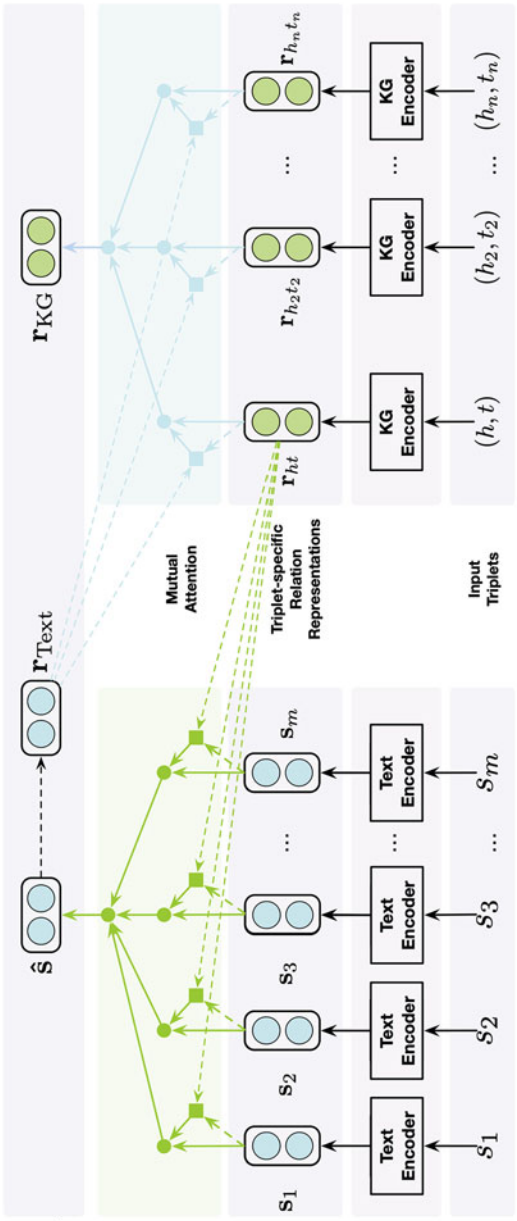

Fig. 9.17 The mutual attention to reformulate both the text and knowledge models [58]. The figure is redrawn according to Fig. 1 from the paper of Han et al. [58]

can indicate the latent relation between h and t, computed based on knowledge representations. $\hat{\mathbf{s}}$ is the feature after synthesizing the information of all sentences, which is used to predict the relation between h and t finally.

Similar to using knowledge representations to select high-quality sentences, we can also use semantic information to select triples conducive to learning relations. This process can be formalized as

$$\alpha = \text{Softmax}(\mathbf{r}_{\text{Text}}^{\top}\mathbf{W}_{\text{SA}}\mathbf{R}), \quad \mathbf{r}_{\text{KG}} = \mathbf{R}\alpha^{\top}, \tag{9.68}$$

where $\mathbf{W}_{\text{SA}}$ is a bilinear matrix of the semantics-guided attention, $\mathbf{R} = [\mathbf{r}_{h_1t_1}, \mathbf{r}_{h_2h_2}, \cdots]$ are the triplet-specific relation representations of the triplets $\{(h_1, r, t_1), (h_2, r, t_2), \cdots\}$. $\mathbf{r}_{\text{Text}}$ is the semantic representation of the relation r used by the RE model. $\mathbf{r}_{\text{KG}}$ is the final relation representation enhanced with semantic information.

This work is a typical attempt to apply knowledge representations of existing KGs to reformulate knowledge acquisition models. In Sect. 9.5, we will introduce knowledge acquisition in more detail.

Example: Knowledge Reformulation for Entity Typing Entity typing is the task of detecting semantic types for a named entity (or entity mention) in plain text. For example, given a sentence *Jordan played 15 seasons in the NBA*, entity typing aims to infer that *Jordan* in this sentence is a *person*, an *athlete*, and even a *basketball player*. Entity typing is important for named entity disambiguation since it can narrow down the range of candidates for an entity mention [21]. Moreover, entity typing also benefits massive NLP tasks such as relation extraction [98], question answering [184], and knowledge base population [20].

Neural models [36, 138] have achieved state-of-the-art performance for fine-grained entity typing. However, these methods only consider the textual information of named entity mentions for entity typing while ignoring the rich information that KGs can provide for determining entity types. For example, in the sentence *In 1975, Gates ... Microsoft ... company*, even though we have no type information of *Microsoft* in KGs, other entities similar to *Microsoft* (e.g., *IBM*) in KGs can also provide supplementary information to help us determine the type of *Microsoft*. To take advantage of KGs for entity typing, knowledge-guided attention for neural entity typing (KNET) has been proposed [178].

As illustrated in Fig. 9.18, KNET mainly consists of two parts. Firstly, KNET builds a neural network, including a bidirectional LSTM and a fully connected layer, to generate context and named entity mention representations. Secondly, KNET introduces a knowledge-guided attention mechanism to emphasize those critical words and improve the quality of context representations. Here, we introduce the knowledge-guided attention in detail. KNET employs the translation method TransE to obtain entity embedding $\mathbf{e}$ for each entity e in KGs. During the training process, given the context words $c = \{w_i, \cdots, w_j\}$, a named entity mention m

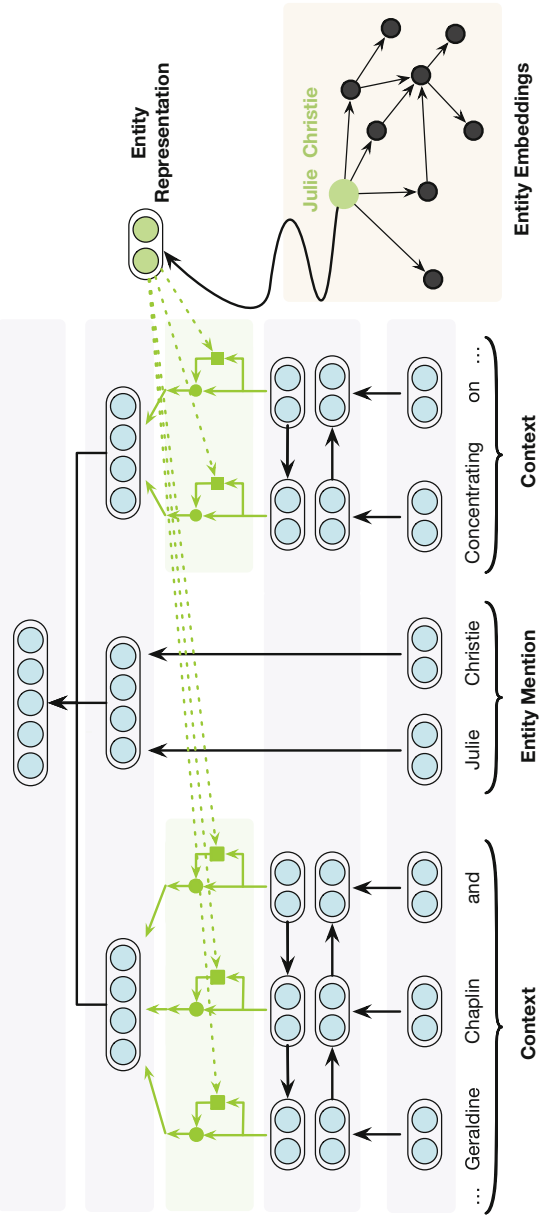

Fig. 9.18 The architecture of KNET [178]. The figure is redrawn according to Fig. 1 from KNET paper [178]

and its corresponding entity embedding $\mathbf{e}$, KNET computes the knowledge-guided attention as

$$\alpha = \text{Softmax}(\mathbf{e}^\top \mathbf{W}_{\text{KA}} \mathbf{H}), \quad \mathbf{c} = \mathbf{H}\alpha^\top, \tag{9.69}$$

where $\mathbf{W}_{\text{KA}}$ is a bilinear matrix of the knowledge-guided attention and $\mathbf{H} = [\mathbf{h}_i, \cdots, \mathbf{h}_j]$ are the bidirectional LSTM states of $\{w_i, \cdots, w_j\}$. The context representation $\mathbf{c}$ is used as an important feature for the subsequent process of type classification.

Through the above two examples of knowledge acquisition and entity typing, we introduce how to highlight important features based on knowledge in the model preprocessing stage, so as to output better features to help improve model performance.

Knowledgeable Post-Processing Apart from reformulating model functions for pre-processing, on the other hand, knowledge can be used as an expert at the end of models for post-processing, guiding models to obtain more accurate and effective results [1, 51, 124]. Formally, x_i is first input to the function f and then input to the function k as

$$f_k(x_i) = k(f(x_i)), \tag{9.70}$$

where $k(\cdot)$ is the knowledge-guided model function used for post-processing and $f(\cdot)$ is the original model function. Knowledgeable post-processing is widely used by knowledge-guided language modeling to improve the word prediction process [1, 51]. Next, we will take a typical knowledge-guided language modeling method NKLM [1] as an example to show how to use knowledge representations to improve model post-processing (Fig. 9.19).

Example: Knowledge Post-Processing on Language Modeling NKLM [1] aims to perform language modeling by considering both semantics and knowledge to generate text. Specifically, NKLM designs two ways to generate each word in the text. The first is the same as conventional auto-regressive models that generate a vocabulary word according to the probabilities over the vocabulary. The second is to generate a knowledge word according to external KGs. Specifically, NKLM uses the LSTM architecture as the backbone to generate words. For external KGs, NKLM stores knowledge representations to build a knowledgeable module $\mathcal{K} = \{(\mathbf{a}_1, O_1), (\mathbf{a}_2, O_2), \cdots, (\mathbf{a}_n, O_n)\}$, in which O_i denotes the description of the i-th fact, $\mathbf{a}_i$ denotes the concatenation of the representations of the head entity, relation and tail entity of the i-th fact.

Given the context $\{w_1, w_2, \cdots, w_{t-1}\}$, NKLM takes both the vocabulary word representation $\mathbf{w}^v_{t-1}$, the knowledge word representation $\mathbf{w}^o_{t-1}$, and the knowledge-guided representation $\mathbf{a}_{t-1}$ at the step $t-1$ as LSTM's input $\mathbf{x}_t = \{\mathbf{w}^v_{t-1}, \mathbf{w}^o_{t-1}, \mathbf{a}_{t-1}\}$. $\mathbf{x}_t$ is then fed to LSTM together with the hidden state $\mathbf{h}_{t-1}$ to get the output state $\mathbf{h}_t$. Next, a two-layer multilayer perceptron $f(\cdot)$ is applied to the concatenation of $\mathbf{h}_t$ and $\mathbf{x}_t$ to get the fact key $\mathbf{k}_t = f(\mathbf{h}_t, \mathbf{x}_t)$. $\mathbf{k}_t$ is

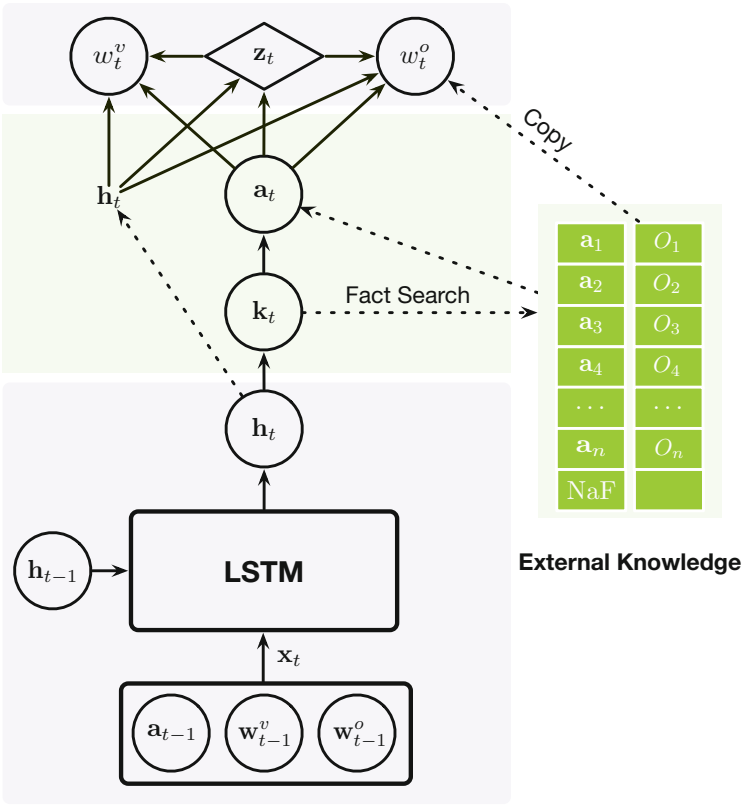

Fig. 9.19 The architecture of NKLM [1]. A special entry (NaF,) is included in the knowledgable module to allow the absence of knowledge when the currently generated word is not included in the knowledgable module. NaF is short for *not a fact*. The figure is redrawn according to Fig. 1 from NKLM paper [1]

then used to extract the most relevant fact representation $\mathbf{a}_t$ from the knowledgeable module. Finally, the selected fact $\mathbf{a}_t$ is combined with the hidden state $\mathbf{h}_t$ to output a vocabulary word w_t^v and knowledge word w_t^o (which is copied from the entity name in the t-th fact), and then determine which word to generate at the step t.

Overall, by using KGs to enhance the post-processing of language modeling, NKLM can generate sentences that are highly related to world knowledge, which are often difficult to model without considering external knowledge.

9.4.3 Knowledge Regularization

Knowledge regularization aims to use knowledge to modify the objective functions of models:

$$\min_{f \in \mathcal{F}} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(y_i, f(x_i)) + \lambda_k \mathcal{L}_k(k, f(x_i)) + \lambda \mathcal{J}(f), \tag{9.71}$$

where $\mathcal{L}_k(k, f(x_i))$ is the additional predictive targets and learning objectives constructed based on knowledge and λ_k is a hyper-parameter to control the knowledgeable loss term.

Distant supervision [109] is a representative method that uses external knowledge to heuristically annotate corpora as additional supervision signals. For many vital information extraction tasks, such as RE [58, 72, 91, 196] and entity typing [36, 138, 178], distant supervision is widely applied for model training. As we will introduce distant supervision in Sect. 9.5 to show how to build additional supervision signals with knowledge, we do not introduce concrete examples here.

Knowledge regularization is also widely used by knowledge-guided PTMs [124, 163, 205]. To fully integrate knowledge into language modeling, these knowledge-guided PTMs design knowledge-specific tasks as their pre-training objectives and use knowledge representations to build additional prediction objectives. Next, we will take the typical knowledge-guided PTM ERNIE [205] as an example to show how knowledge regularization can help the learning process of models.

Example: Knowledge Regularization for PTMs PTMs like BERT [33] have great abilities to extract features from text. With informative language representations, PTMs obtain state-of-the-art results on various NLP tasks. However, the existing PTMs rarely consider incorporating external knowledge, which is essential in providing related background information for better language understanding. For example, given a sentence *Bob Dylan wrote Blowin' in the Wind and Chronicles: Volume One*, without knowing *Blowin' in the Wind* is a *song* and *Chronicles: Volume One* is a *book*, it is not easy to know the occupations of *Bob Dylan*, i.e., *songwriter* and *writer*.

To this end, an enhanced language representation model with informative entities (ERNIE) is proposed [205]. Figure 9.20 is the overall architecture of ERNIE. ERNIE first augments the input data using knowledge augmentation as we have mentioned in Sect. 9.4.1. Specifically, ERNIE recognizes named entity mentions and then aligns these mentions to their corresponding entities in KGs. Based on the alignments between text and KGs, ERNIE takes the informative entity representations as additional input features.

Similar to conventional PTMs, ERNIE adopts masked language modeling and next sentence prediction as the pre-training objectives. To better fuse textual and knowledge features, ERNIE proposes *denoising entity auto-encoding (DAE)* by randomly masking some mention-entity alignments in the text and requiring models

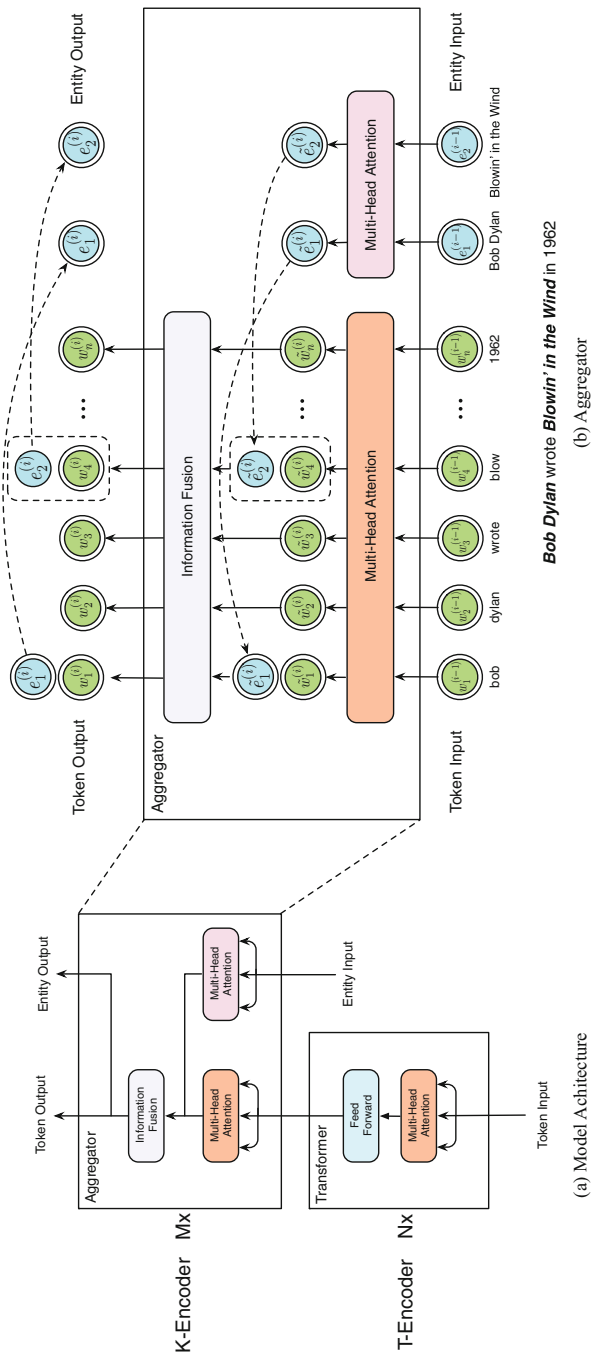

Fig. 9.20 The architecture of ERNIE [205]. The figure is redrawn according to Fig. 2 from ERNIE paper [205]

to select appropriate entities to complete the alignments. Different from the existing PTMs that predict tokens with only using local context, DAE requires ERNIE to aggregate both text and knowledge to predict both tokens and entities, leading to knowledge-guided language modeling. DAE is clearly a knowledge-guided objective function.

In addition to ERNIE, there are other representative works on knowledge regularization. For example, KEPLER [163] incorporates structured knowledge into its pre-training. Specifically, KEPLER encodes the textual description of entities as entity representations and predicts the relation between entities based on these description-based representations. In this way, KEPLER can learn the structured information of entities and relations in KGs in a language-modeling manner. WKLM [181] proposes a pre-training objective type-constrained entity replacement. Specifically, WKLM randomly replaces the named entity mentions in the text with other entities of the same type and requires the model to identify whether an entity mention is replaced or not. Based on the new pre-training objective, WKLM can accurately learn text-related knowledge and capture the type information of entities.

From Fig. 9.20, we can find that ERNIE also adopts knowledge reformulation by adding the new aggregator layers designed for knowledge integration to the original Transformer architecture. To a large extent, the success of knowledge-guided PTMs comes from the fact that these models use knowledge to enhance important factors of model learning. Up to now, we have introduced knowledge augmentation, knowledge reformulation, and knowledge regularization. Next, we will further introduce knowledge transfer.

9.4.4 Knowledge Transfer

Knowledge transfer aims to use knowledge to obtain a knowledgeable hypothesis space, reducing the cost of searching optimal parameters and making it easier to train an effective model. There are two typical approaches to transferring knowledge: (1) transfer learning [120] that focuses on transferring model knowledge learned from *labeled* data to downstream task-specific models and (2) self-supervised learning [97] that focuses on transferring model knowledge learned from *unlabeled* data to downstream task-specific models. More generally, the essence of knowledge transfer is to use prior knowledge to constrain the hypothesis space:

$$\min_{f \in \mathcal{F}_k} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(y_i, f(x_i)) + \lambda \mathcal{J}(f), \tag{9.72}$$

where $\mathcal{F}_k$ is the knowledge-guided hypothesis space.

Knowledge transfer is widely used in NLP. The fine-tuning stage of PTMs is a typical scenario of knowledge transfer, which aims to transfer the versatile knowledge acquired in the pre-training stage to specific tasks. Intuitively, after pre-training

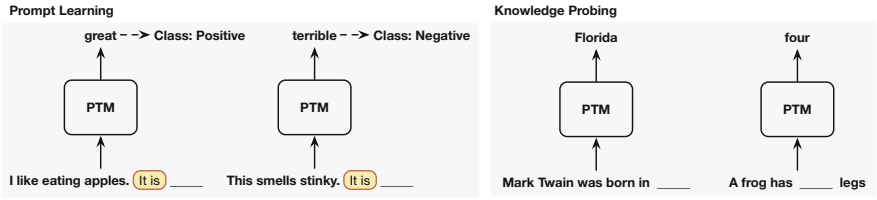

Fig. 9.21 By using prompts, we can stimulate the knowledge of PTMs to handle specific tasks such as sentiment classification and predicting symbolic knowledge

a PTM, fine-tuning this PTM can be seen as narrowing down searching task-specific parameters to a local hypothesis space around the pre-trained parameters rather than the global hypothesis space.

As we mentioned in Chap. 5, in addition to fine-tuning PTMs, prompt learning has also been widely explored. Despite the success of fine-tuning PTMs, it still faces two challenges. On the one hand, there is a gap between the objectives of pre-training and fine-tuning, since most PTMs are learned with language modeling objectives, yet downstream tasks may have quite different objective forms such as classification, regression, and labeling. On the other hand, as the parameter size of PTMs increases rapidly, fine-tuning PTMs has become resource-intensive. In order to alleviate these issues, prompts have been introduced to utilize the knowledge of PTMs in an effective and efficient manner [93].

As shown in Fig. 9.21, prompt learning aims at converting downstream tasks into a cloze-style task similar to pre-training objectives so that we can better transfer the knowledge of PTMs to downstream tasks. Taking prompt learning for sentiment classification as an example, a typical prompt consists of a template (e.g., . . . *It was* `[MASK]`.) and a label word set (e.g., *great* and *terrible*) as candidates for predicting `[MASK]`. By changing the input using the template to predict `[MASK]` and mapping the prediction to corresponding labels, we can apply masked language modeling for sentiment classification. For example, given the sentence *I like eating apples.*, we first use the prompt template to get the new input sentence *I like eating apples. It was* `[MASK]`. According to PTMs predicting *great* or *terrible* at the masked position, we can determine whether this sentence is positive or negative.

The recently proposed large-scale PTM GPT-3 [17] shows the excellent performance of prompt learning in various language understanding and generation tasks. In prompt learning, all downstream tasks are transformed to be the same as the pre-training tasks. And since the parameters of PTMs are frozen during prompt learning, the size of hypothesis space is much smaller compared to fine-tuning, making more efficient knowledge transfer possible.

Overall, PTMs play an important role in driving the use of model knowledge. And to some extent, PTMs also influence the paradigm of using symbolic knowledge in NLP. As shown in Fig. 9.21, many knowledge probing works [74, 125, 126] show that by designing prompt, PTMs can even complete structured knowledge information. These studies show that PTMs, as good carriers of symbolic

knowledge, can memorize symbolic knowledge well. Moreover, these studies also indicate one factor that may contribute to the power of PTMs: knowledge can be spontaneously abstracted by PTMs from large-scale unstructured data and then used to solve concrete problems, and the abstracted knowledge matches well with the knowledge formed by human beings. Inspired by this, we can further delve into how PTMs abstract knowledge and how PTMs store knowledge in their parameters, which is very meaningful for further advancing the integration of symbolic knowledge and model knowledge. On the other hand, all these studies also show the importance of knowledge-guided NLP. Compared with letting models slowly abstract knowledge from large-scale data, directly injecting symbolic knowledge into models is a more effective solution.

The success of PTMs demonstrates the clear advantages of fully transferring existing model knowledge in terms of computing efficiency and effectiveness, as compared to learning a model from scratch. Since we have introduced the details of PTMs in Chap. 5, in this section, we mainly discuss the valuable properties of knowledge transfer owned by PTMs.

9.4.5 Summary

In this section, we present several ways in which knowledge is used to guide NLP models. Depending on the location of model learning where knowledge steps in, we group the guidance from knowledge into four categories: (1) *knowledge augmentation*, where knowledge is introduced to augment the input data, (2) *knowledge reformulation*, where special model modules are designed to interact with knowledge, (3) *knowledge regularization*, where knowledge does not directly intervene the forward pass of the model but acts as a regularizer, and (4) *knowledge transfer*, where knowledge helps narrow down the hypothesis space to achieve more efficient and effective model learning.

These approaches enable effective integration of knowledge into deep models, allowing models to leverage sufficient knowledge (especially symbolic knowledge) to better perform NLP tasks. Since knowledge is essential for models to understand and complete the NLP tasks, knowledge-guided NLP is a worthwhile area for researchers to continue to explore.

9.5 Knowledge Acquisition

The KBs used in early expert systems and the KGs built in recent years both have long relied on manual construction. Manually organizing knowledge ensures that knowledge systems are constructed with high quality but suffers from inefficiency, incompleteness, and inconsistency in the annotation process. As shown in Fig. 9.22,

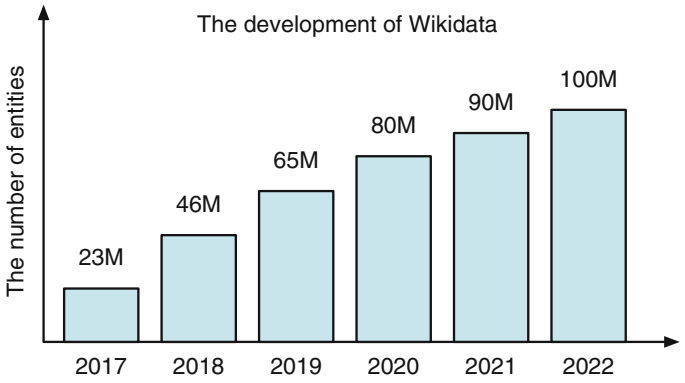

Fig. 9.22 The development trend of Wikidata from 2017 to 2022

the number of entities in the popular open-source KG Wikidata[4] grew at a rate of over 15 million per year from 2017 to 2022. At this rate of growth, it is unrealistic to rely solely on human annotation to organize large-scale human knowledge. Therefore, it is crucial to explore automatic knowledge acquisition, which can significantly better support knowledge representation learning and knowledge-guided NLP. In this section, taking KGs that store rich world knowledge as an example, we describe how to perform automatic knowledge acquisition to enrich the amount of knowledge for KGs.

Generally, we have several approaches to acquiring knowledge. Knowledge graph completion (KGC) and RE are two typical approaches. As shown in Fig. 9.23, KGC aims to obtain new knowledge by reasoning over the internal structure of KGs. For example, given the triplet ⟨*Mark Twain*, `Place of Birth`, *Florida*⟩ and the triplet ⟨*Florida*, `City of`, *U.S.A*⟩, we can easily infer the fact ⟨*Mark Twain*, `Citizenship`, *U.S.A*⟩. Different from KGC that infers new knowledge based on the internal information of KGs, RE focuses on detecting relations between entities from external plain text. For example, given the sentence *Mark Twain was an American author and humorist*, we can get the triplet ⟨*Mark Twain*, `Citizenship`, *U.S.A*⟩ from the semantic information of the sentence. Since the text is the core carrier of human knowledge, RE can obtain more and broader knowledge than KGC. Moreover, KGC highly relies on the knowledge representation learning methods that we have introduced in the previous Sect. 9.3. Therefore, in this section, we only introduce knowledge acquisition by using RE as an example.

As RE is an important way to acquire knowledge, many researchers have devoted extensive efforts to this field in the past decades. Various statistical RE methods based on feature engineering [75, 208], kernel models [18, 27], and probabilistic

[4] https://www.wikidata.org.

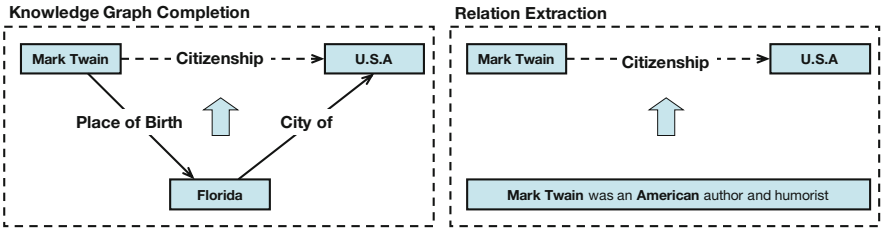

Fig. 9.23 An example of knowledge graph completion and relation extraction

graphical models [133, 134] have been proposed and achieved promising results. With the development of deep learning, neural networks as a powerful tool for encoding semantics have further advanced the development of RE, including recursive neural networks [110, 144], convolutional neural networks [92, 197], recurrent neural networks [115, 201], and graph neural networks [204, 210]. Considering that neural networks have become the backbone of NLP research in recent years, we focus on introducing knowledge acquisition with neural RE models in this section. For those statistical methods, some surveys [121, 195] can provide sufficient details about them. Next, we present how to acquire knowledge in various complex textual scenarios around neural RE, including sentence-level methods, bag-level methods, document-level methods, few-shot methods, and contextualized methods.

9.5.1 Sentence-Level Relation Extraction

Sentence-level RE is the basis for acquiring knowledge from text to enrich KGs. As shown in Fig. 9.24, sentence-level RE is based on the sentence-level semantics to extract relations between entities. Formally, given an input sentence $s = \{w_1, w_2, \cdots, w_n\}$ consisting of n words and an entity pair (e_1, e_2) in the sentence, sentence-level RE aims to obtain the probability distribution $P(r|s, e_1, e_2)$ over the relation set $\mathcal{R}$ ($r \in \mathcal{R}$). Based on $P(r|s, e_1, e_2)$, we can infer all relations between e_1 and e_2.

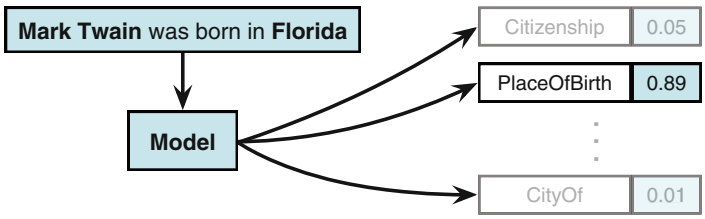

Fig. 9.24 An example of sentence-level relation extraction

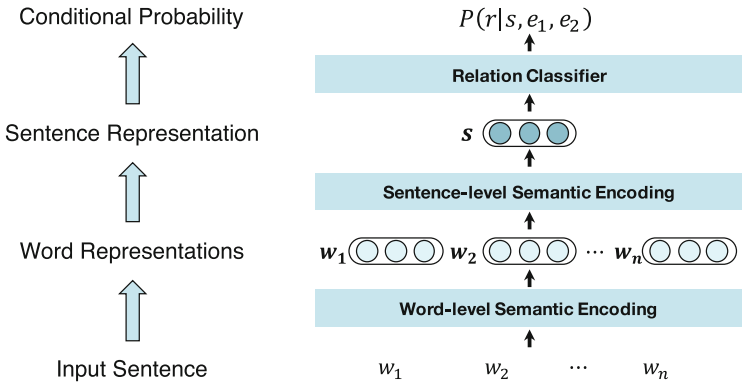

Fig. 9.25 The process of encoding sentence-level semantic information to detect the relations expressed by a given sentence

Learning an effective model to measure $P(r|s, e_1, e_2)$ requires efforts of three different aspects. As shown in Fig. 9.25, the first is to encode the input words into informative word-level features $\{\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_n\}$ that can well serve the relation classification process. The second is to train a sentence encoder, which can well encode the word-level features $\{\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_n\}$ into the sentence-level feature $\mathbf{s}$ with respect to the entity pair (e_1, e_2). The third is to train a classifier that can well compute the conditional probability distribution $P(r|s, e_1, e_2)$ over all relations in $\mathcal{R}$ based on the sentence-level feature $\mathbf{s}$. Next, we will present some typical works in each of these three aspects.

Word-Level Semantics Encoding Given the sentence $s = \{w_1, w_2, \cdots, w_n\}$ and the entity pair (e_1, e_2), before encoding sentence semantics and further classifying relations, we have to project the discrete words of the source sentence s into a continuous vector space to get the input representation $\mathbf{w} = \{\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_n\}$. In general, widely used word-level features include the following components:

Word Embeddings Word embeddings aim to encode the syntactic and semantic information of words into distributed representations, i.e., each word is represented by a vector. Word embeddings are the basis for encoding word-level semantics, and word2vec [108] and GloVe [123] are the most common ways to obtain word embeddings.

Position Embeddings Position embeddings aim to encode which input words belong to the target entities and how close each word is to the target entities. Specifically, for each word w_i, its position embedding is formalized as the combination of the relative distances from w_i to e_1 and e_2. For instance, given the sentence *Mark Twain was an American author and humorist*, the relative distance from the word *was* to the entity *Mark Twain* is -1, and the distance to the entity *American* is 2. The relative distances -1 and 2 are then encoded into the position embedding to provide a positional representation for the word *was*. Since RE highly relies on word-level

positional information to capture entity-specific semantics, position embeddings are widely used in RE [135, 197, 201].

Part-of-Speech (POS) Tag Embeddings POS Tag Embeddings aim to encode the word-level lexical information (e.g., nouns, verbs, etc.) of the sentence. Formally, all words in the sentence are encoded into embeddings according to their POS tags, and these POS tag embeddings can serve as lexical complements for word embeddings and position embeddings [19, 183, 209].

Hypernym Embeddings Hypernym embeddings aim to leverage the prior knowledge of hypernyms in WordNet [43]. Compared to POS tags, hypernyms are finer-grained. WordNet is a typical linguistic KG. In WordNet, all words are grouped into sets of cognitive synonyms (synsets), and each synset can express a distinct concept. Hypernyms are defined among these synsets. Here is just a brief introduction to WordNet, and we will introduce linguistic knowledge in detail in Chap. 10. When given the hypernym information of each word in WordNet (e.g., noun.food, verb.motion, etc), it is easy to connect this word with other words that are different but conceptually similar. Similar to POS tag embeddings, each hypernym tag in WordNet has a tag-specific embedding, and each word in a sentence is encoded into a hypernym embedding based on the word-specific hypernym tag.

The above embeddings are usually concatenated together to obtain the final input features $\mathbf{w} = \{\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_n\}$, and $\mathbf{w}$ is used to support further encoding sentence-level semantics.

Sentence-Level Semantics Encoding Based on word-level features, we introduce different sentence encoders to encode sentence-level semantic information for RE:

Convolutional Neural Network Encoders CNN encoders [135, 197] use convolutional layers to extract local features and then use pooling operations to encode all local features into a fixed-sized vector.

Here we take an encoder with only one convolutional layer and one max-pooling operation as an example. Given the word-level features $\{\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_n\}$, the convolutional layer can be formalized as

$$\{\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_n\} = \text{CNN}(\{\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_n\}), \tag{9.73}$$

where $\text{CNN}(\cdot)$ indicates the convolution operation inside the convolutional layer, $\mathbf{h}_i$ is the hidden state of the i-th word, and we have introduced this part in Chap. 4. Then, the sentence representation $\mathbf{s}$ is obtained by using the max-pooling operation, where the i-th element of $\mathbf{s}$ is given as

$$[\mathbf{s}]_i = \max_{1 \le j \le n} [\mathbf{h}_j]_i, \tag{9.74}$$

where $[\cdot]_i$ is the i-th element of the vector.

Further, PCNN [196], which is a variant of CNN, adopts a piecewise max-pooling operation. All hidden states $\{\mathbf{p}_1, \mathbf{p}_2, \cdots \mathbf{p}_n\}$ are divided into three parts by

the positions of e_1 and e_2. The max-pooling operation is performed on the three segments respectively, and $\mathbf{s}$ is the concatenation of the three pooling results.

Recurrent Neural Network Encoders RNN encoders [201] use recurrent layers to learn temporal features on the input sequence. Given the word-level features $\{\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_n\}$, each input word feature is fed into recurrent layers step by step. For the i-th step, the network takes $\mathbf{w}_i$ and the hidden state of the last step $\mathbf{h}_{i-1}$ as input, and the whole process is given as

$$\mathbf{h}_i = \mathrm{RNN}(\mathbf{w}_i, \mathbf{h}_{i-1}), \tag{9.75}$$

where $\mathrm{RNN}(\cdot)$ indicates the RNN function, which can be a LSTM unit or a GRU unit mentioned in Chap. 4.

The conventional recurrent models typically encode sequences from start to end and build the hidden state of each step only considering its preceding steps. Besides unidirectional RNNs, bidirectional RNNs [137] are also adopted to encode sentence-level semantics, and the whole process is given as

$$\overleftarrow{\mathbf{h}}_i = \overleftarrow{\mathrm{RNN}}(\mathbf{w}_i, \overleftarrow{\mathbf{h}}_{i+1}), \ \overrightarrow{\mathbf{h}}_i = \overrightarrow{\mathrm{RNN}}(\mathbf{w}_i, \overrightarrow{\mathbf{h}}_{i-1}), \mathbf{h}_i = [\overleftarrow{\mathbf{h}}_i; \overrightarrow{\mathbf{h}}_i], \tag{9.76}$$

where $[\cdot;\cdot]$ is the concatenation of two vectors.

Similar to the abovementioned convolutional models, the recurrent models also use pooling operations to extract the global sentence feature $\mathbf{s}$, which forms the representation of the whole input sentence. For example, we can use a max-pooling operation to obtain $\mathbf{s}$:

$$[\mathbf{s}]_i = \max_{1 \le j \le n} [\mathbf{h}_j]_i. \tag{9.77}$$

Besides pooling operations, attention operations [3] can also combine all local features. Specifically, given the output states $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_n]$ produced by a recurrent models, $\mathbf{s}$ can be formalized as

$$\alpha = \mathrm{Softmax}(\mathbf{q}^\top f(\mathbf{H})), \quad \mathbf{s} = \mathbf{H}\alpha^\top, \tag{9.78}$$

where $\mathbf{q}$ is a learnable query vector and $f(\cdot)$ is an attention transformation function.

Moreover, some works [110] propose to encode semantics from both the word sequence and tree-structured dependency of a sentence by stacking bidirectional path-based recurrent neural networks. More specifically, these path-based methods mainly consider the shortest path between entities in the dependency tree, and utilize stacked layers to encode the path as the sentence representation. Some preliminary works [182] have shown that these paths are informative in RE and proposed various recursive neural models for this. Next, we will introduce these recursive models in detail.

Recursive Neural Network Encoders Recursive encoders aim to extract features based on syntactic parsing trees, considering that the syntactic information between target entities in a sentence can benefit classifying their relations. Generally, these encoders utilize the parsing tree structure as the composition direction to integrate word-level features into sentence-level features. Socher et al. [144] introduce a recursive matrix-vector model that can capture the structure information by assigning a matrix-vector representation for each constituent in parsing trees. In Socher's model, the vector can represent the constituent, and the matrix can represent how the constituent modifies the word meaning it is combined with.

Tai et al. [151] further propose two tree-structured models, the Child-Sum Tree-LSTM and the N-ary Tree-LSTM. Given the parsing tree of a sentence, the transition equations of the Child-Sum Tree-LSTM are defined as

$$\mathbf{h}_t = \sum_{k \in C(t)} \text{TLSTM}(\mathbf{h}_k), \tag{9.79}$$

where $C(t)$ is the children set of the node t, TLSTM$(\cdot)$ indicates a Tree-LSTM cell, which is simply modified from the LSTM cell, and the hidden states of the leaf nodes are the input features. The transition equations of the N-ary Tree-LSTM are similar to the transition equations of Child-Sum Tree-LSTM. The main difference is that the N-ary Tree-LSTM limits the tree structures to have at most N branches. More details of recursive neural networks can be found in Chap. 4.

Sentence-Level Relation Classification After obtaining the representation $\mathbf{s}$ of the input sentence s, we require a relation classifier to compute the conditional probability $P(r|s, e_1, e_2)$. Generally, $P(r|s, e_1, e_2)$ can be obtained with

$$P(r|s, e_1, e_2) = \text{Softmax}(\mathbf{M}\mathbf{s} + \mathbf{b}), \tag{9.80}$$

where $\mathbf{M}$ is the relation matrix consisting of relation embeddings and $\mathbf{b}$ is a bias vector. Intuitively, using a Softmax layer to compute the conditional probability means that an entity pair has only one corresponding relation. However, sometimes multiple relations may exist between an entity pair. To this end, for each relation $r \in \mathcal{R}$, some works perform relation-specific binary classification:

$$P(r|s, e_1, e_2) = \text{Sigmoid}(\mathbf{r}^\top \mathbf{s} + b_r), \tag{9.81}$$

where $\mathbf{r}$ is the relation embedding of r and b_r is a relation-specific bias value.

9.5.2 Bag-Level Relation Extraction

Although existing neural methods have achieved promising results in sentence-level RE, these neural methods still suffer from the problem of data scarcity since manu-

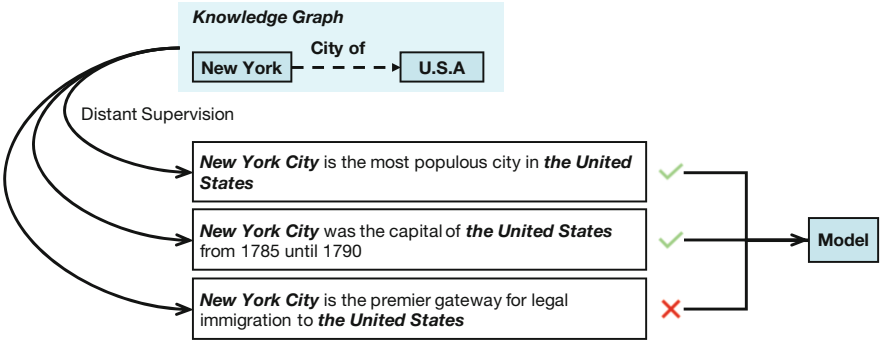

Fig. 9.26 An example of bag-level relation extraction

ally annotating training data is time-consuming and labor-intensive. To alleviate this problem, distant supervision [109] has been introduced to automatically annotate training data by aligning existing KGs and plain text. The main idea of distant supervision is that sentences containing two entities may describe the relations of the two entities recorded in KGs. As shown in Fig. 9.26, given (*New York*, `City of`, *U.S.A*), the distant supervision assumption regards all sentences that contain *New York* and *U.S.A* as positive instances for the relation `City of`. Besides providing massive training data, distant supervision also naturally provides a way to detect the relations between two given entities based on multiple sentences (bag-level) rather than a single sentence (sentence-level).

Therefore, bag-level RE aims to predict the relations between two given entities by considering all sentences containing these entities, by highlighting those informative examples and filtering out noisy ones. As shown in Fig. 9.26, given the input sentence set $\mathcal{S} = \{s_1, s_2, \cdots, s_m\}$ and an entity pair (e_1, e_2) contained by these sentences, bag-level RE methods aim to obtain the probability $P(r|\mathcal{S}, e_1, e_2)$ over the relation set.

As shown in Fig. 9.27, learning an effective model to measure $P(r|\mathcal{S}, e_1, e_2)$ requires efforts from three different aspects: encoding sentence-level semantics (including encoding word-level semantics), encoding bag-level semantics, and finally classifying relations. Since encoding word-level semantics and sentence-level semantics have been already introduced in sentence-level RE, we mainly focus on introducing how to encode bag-level semantics here.

Bag-Level Semantics Encoding For bag-level RE, we need to encode bag-level semantics based on sentence-level representations. Formally, given a sentence bag $\mathcal{S} = \{s_1, s_2, \cdots, s_m\}$, each sentence s_i has its own sentence representation $\mathbf{s}_i$; a bag-level encoder encodes all sentence representations into a single bag representation $\hat{\mathbf{s}}$. Next, we will introduce some typical bag-level encoders as follows:

Max Encoders Max encoders aim to select the most confident sentence in the bag $\mathcal{S}$ and use the representation of the selected sentence as the bag representation,

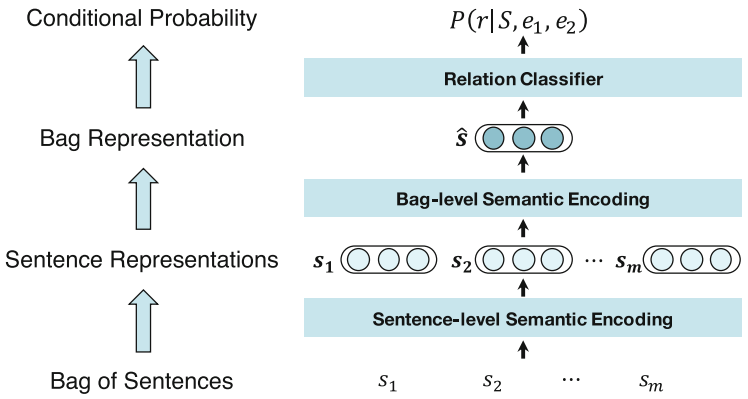

Fig. 9.27 The process of obtaining bag-level semantic information to detect the relations described by a given sentence bag

considering not all sentences containing an entity pair can express the relations between the entity pair. For instance, given *New York City is the premier gateway for legal immigration to the United States*, the sentence does not highly express the relation `City of`. To this end, the at-least-one assumption [196] has been proposed, assuming that at least one sentence containing target entities can express their relations. With the at-least-one assumption, the sentence with the highest probability for a specific relation is selected to represent the bag $\mathcal{S}$. Formally, the bag representation is given as

$$\hat{\mathbf{s}} = \mathbf{s}_{i^*}, \quad i^* = \arg\max_i P(r|s_i, e_1, e_2). \tag{9.82}$$

Average Encoders Average encoders use the average of all sentence vectors to represent the bag. Max encoders use only one sentence in the bag as the bag representation, ignoring the rich information and correlation among different sentences in the bag. To take advantage of all sentences, Lin et al. [91] make the bag representation $\hat{\mathbf{s}}$ depends on the representations of all sentences in the bag. The average encoder assumes all sentences contribute equally to the bag representation $\hat{\mathbf{s}}$:

$$\hat{\mathbf{s}} = \sum_i \frac{1}{m} \mathbf{s}_i. \tag{9.83}$$

Attentive Encoders Attentive encoders use attention operations to aggregate all sentence vectors. Considering the inevitable mislabeling problem introduced by distant supervision, average encoders may be affected by those mislabeled sentences. To address this problem, Lin et al. [91] further propose a sentence-level selective

attention to reduce the side effect of mislabeled data. Formally, with the attention operation, the bag representation $\hat{\mathbf{s}}$ is defined as

$$\alpha = \text{Softmax}(\mathbf{q}_r^\top f(\mathbf{S})), \quad \hat{\mathbf{s}} = \mathbf{S}\alpha^\top, \tag{9.84}$$

where $\mathbf{S} = \{\mathbf{s_1}, \mathbf{s_2}, \cdots, \mathbf{s_m}\}$, $f(\cdot)$ is an attention transformation function and and $\mathbf{q}_r$ is the query vector of the relation r used for the attention operation. To further improve the attention operation, more sophisticated mechanisms, such as knowledge-enhanced strategies [58, 72, 199], soft-labeling strategies [95], reinforcement learning [44, 200], and adversarial training [171], have been explored to build more effective attention operations.

Bag-Level Relation Classification Similar to sentence-level methods, when obtaining the bag representation $\hat{\mathbf{s}}$, the probability $P(r|\mathcal{S}, e_1, e_2)$ is computed as

$$P(r|\mathcal{S}, e_1, e_2) = \text{Softmax}(\mathbf{M}\hat{\mathbf{s}} + \mathbf{b}), \tag{9.85}$$

where $\mathbf{M}$ is the relation matrix consisting of relation embeddings and $\mathbf{b}$ is a bias vector. For those methods performing relation-specific binary classification, the relation-specific conditional probability is given by

$$P(r|\mathcal{S}, e_1, e_2) = \text{Sigmoid}(\mathbf{r}^\top \hat{\mathbf{s}} + b_r), \tag{9.86}$$

where $\mathbf{r}$ is the relation embedding of r and b_r is a relation-specific bias value.

9.5.3 Document-Level Relation Extraction

For RE, not all relational facts can be acquired by sentence-level or bag-level methods. Many facts are expressed across multiple sentences in a document. As shown in Fig. 9.28, a document may contain multiple entities that interact with each other in a complex way. If we want to get the fact that ⟨*Riddarhuset*, `City of`, *Sweden*⟩, we first have to detect *Riddarhuset* is located in *Stockholm* from the fourth sentence, and then detect *Stockholm* is the capital of *Sweden* and *Sweden* is a country from the first sentence. From these three facts, we can finally infer that the sovereign state of *Riddarhuset* is *Sweden*.

Performing reading and reasoning over multiple sentences becomes important, which is intuitively hard to reach for both sentence-level and bag-level methods. According to the statistics of a human-annotated dataset sampled from Wikipedia [188], more than 40% facts require considering the semantics of multiple sentences for their extraction, which is not negligible. Some works [150, 155] that focus on document-level RE also report similar observations. Hence, it is crucial to advance RE from the sentence level to the document level.

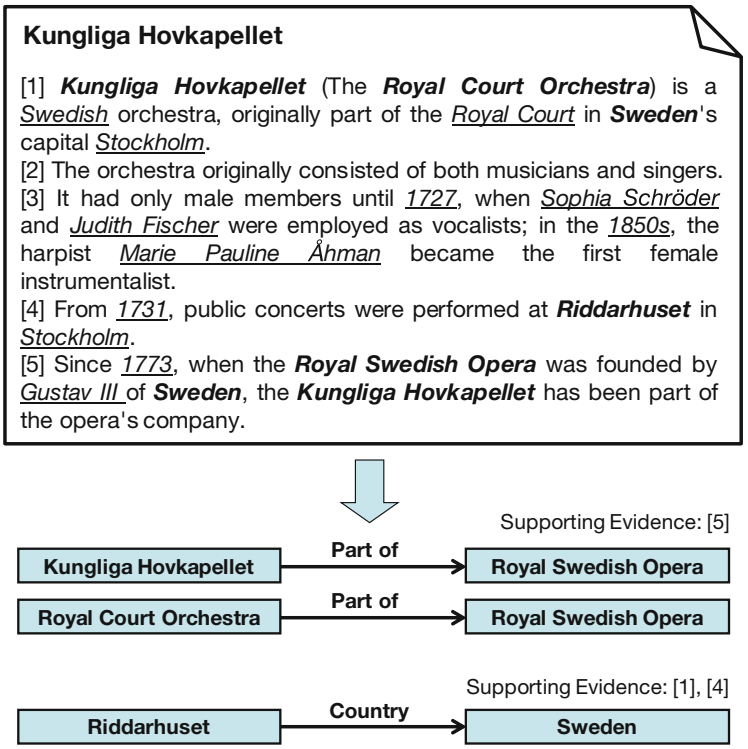

Fig. 9.28 An example of document-level relation extraction from the dataset DocRED [188]

Due to being much more complex than sentence-level and bag-level RE, document-level RE remains an open problem in terms of benchmarking and methodology. For benchmarks that can evaluate the performance of document-level RE, existing datasets either only have a few manually annotated examples [88], or have noisy distantly supervised annotations [122, 129], or serve only a specific domain [85]. To address this issue, Yao et al. [188] manually annotate a large-scale and general-purpose dataset to support the evaluation of document-level RE methods, named DocRED. DocRED is built based on Wikipedia and Wikidata, and it has two main features. First, DocRED is the largest human-annotated document-level RE dataset, containing 132, 375 entities and 56, 354 facts. Second, nearly half of the facts in DocRED can only be extracted from multiple sentences. This makes it necessary for models to read multiple sentences in a document to identify entities and infer their relations by considering the holistic semantic information of the document.

Document-Level RE Methods The preliminary results on DocRED show that existing sentence-level methods cannot work well on DocRED, indicating that

document-level RE is more challenging than sentence-level RE. Many efforts have been devoted to document-level RE based on DocRED.

PTM-Based Methods Wang et al. [158] use PTMs as a backbone to build an effective document-level RE model, considering that recently proposed PTMs show the ability to encode long sequences. Although PTMs can effectively capture contextual semantic information from plain text for document-level RE, PTMs still cannot explicitly handle coreference, which is critical for modeling interactions between entities. Ye et al. [190] introduce a PTM that captures coreference relations between entities to improve document-level RE.

Graph-Based Methods Nan et al. [112] construct document-level graphs based on syntactic trees, coreferences, and some human-designed heuristics to model dependencies in documents. To better model document-level graphs, Zeng et al. [198] construct a path reasoning mechanism using graph neural networks to infer relations between entities.

Document-Level Distant Supervision Some works [128, 172] also propose to leverage document-level distant supervision to learn entity and relation representations. Based on well-designed heuristic rules, these distantly supervised methods perform effective data augmentation for document-level RE.

Cross-Document RE In addition to abovementioned methods for RE within one document, Yao et al. [187] further propose CodRED which aims to acquire knowledge from multiple documents. Cross-document RE presents two main challenges: (1) Given an entity pair, models need to retrieve relevant documents to establish multiple reasoning paths. (2) Since the head and tail entities come from different documents, models need to perform cross-document reasoning via bridging entities to resolve the relations. To support the research, CodRED provides a large-scale human-annotated dataset, which contains 30,504 relational facts, as well as the associated reasoning text paths and evidence. Experiments show that CodRED is challenging for existing RE methods. In summary, acquiring knowledge from documents has drawn increasing attention from the community, and is still a promising direction worth further exploration.

9.5.4 Few-Shot Relation Extraction

As we mentioned before, the performance of conventional RE methods heavily relies on annotated data. Annotating large-scale data is time-consuming and labor-intensive. Although distant supervision can alleviate this issue, the distantly supervised data also exhibits a long-tail distribution, i.e., most relations have very limited instances. In addition, distant supervision suffers from mislabeling, which makes the classification of long-tail relations more difficult. Hence, it is necessary to study training RE models with few-shot training instances. Figure 9.29 is an example of few-shot RE.

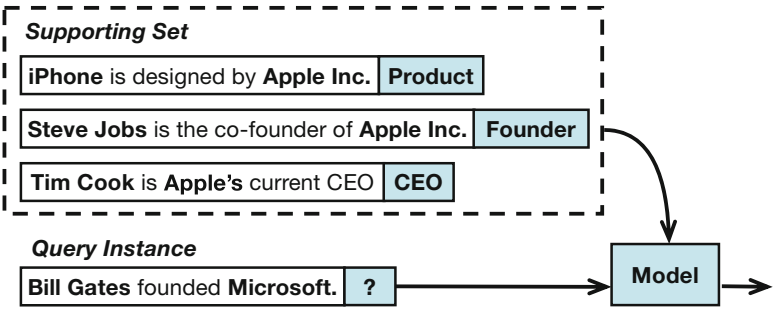

Fig. 9.29 An example of few-shot relation extraction

Few-Shot RE Methods FewRel [62] is a large-scale supervised dataset for few-shot RE, which requires models to handle relation classification with limited training instances. Based on FewRel, some works explore few-shot RE and achieve promising results.

Meta Learning and Metric Learning Methods Han et al. [62] demonstrate that meta learning and metric learning can be well used for few-shot RE. Following the direction of meta learning, Dong et al. [35] propose to leverage meta-information of relations (e.g., relation names and aliases) to guide the initialization and fast adaptation of meta learning for few-shot RE. Following the direction of metric learning, based on the prototypical neural network [141], which is a typical metric learning approach, many more effective metric learning methods [45, 191] are proposed for few-shot RE.

PTM-Based Methods Soares et al. [142] utilize PTMs to handle few-shot RE and show surprising results. On the one hand, the use of PTMs can transfer the knowledge captured from unlabeled data to help solve the problem of data scarcity. On the other hand, Soares et al. introduce contrastive learning based on PTMs, which can be seen as a more effective metric learning method. On FewRel, Soares' model can achieve comparable results to human performance.

Domain Adaptation and Out-of-Distribution Detection After FewRel, Gao et al. [46] propose some more challenging few-shot scenarios, including domain adaptation and out-of-distribution detection. Gao et al. build a more challenging dataset FewRel 2.0 and use sufficient experimental results on FewRel 2.0 to show that the state-of-the-art few-shot methods struggle in these two scenarios, and the commonly used techniques for domain adaptation and out-of-distribution detection cannot handle the two challenges well. These findings call for more attention and further efforts to few-shot RE, which is still a challenging open problem.

9.5.5 Open-Domain Relation Extraction

Most RE systems regard the task as relation classification and can only deal with pre-defined relation types. However, relation types in the real-world corpora are typically in rapid growth. For example, the number of relation types in Wikidata grows to over 10,000 in 6 years from 2017 to 2022 [202]. Therefore, handling emerging relations in the open-domain scenario is a challenging problem for RE. Existing methods for open-domain RE can be divided into three categories: extracting open relation phrases, clustering open relation types, and learning with increasing relation types.

Extracting Open Relation Phrases Open information extraction (OpenIE) aims to extract semi-structured relation phrases [39, 40]. Since the relations are treated as free-form text from the sentences, OpenIE can deal with relations that are not pre-defined. For OpenIE, the traditional statistical methods typically design heuristic rules (e.g., syntactic and lexical constraints) to identify relation phrase candidates and filter out noisy ones via a relation discriminator [41, 169, 189]. Neural OpenIE methods typically learn to generate relation phrases in an encoder-decoder architecture [26, 77], or identify relation phrases in the sentence via sequence labeling [145]. The supervision for neural OpenIE models typically comes from the high-confidence results from the statistical methods. The advantage of OpenIE is that minimal human efforts are required in both relation type design and relation instance annotation. The relational phrases also exhibit good readability to humans. However, due to the diversity of natural language, the same relation type can have different surface forms in different sentences. Therefore, linking various surface forms of relation phrases to the standardized relation types could be difficult.

Clustering Open Relation Types Open relation extraction (OpenRE) aims to discover new relation types by clustering relational instances into groups. OpenRE methods typically learn discriminative representations for relational instances and cluster these open-domain instances into groups. Compared with OpenIE, OpenRE aims at clustering new types that are out of existing relation types, yet OpenIE only focuses on representing relations with language phrases to get rid of pre-defined types. Generally, the results of OpenIE can be used to support the clustering of OpenRE. Elsahar et al. [38] make an initial attempt to obtain relational instance representations through rich features, including entity types and re-weighted word embeddings, and cluster these handcrafted representations to discover new relation types. Then, some works [67, 103] propose to improve the learning and clustering of relational instance representations by using effective self-supervised signals. Notably, Wu et al. [170] propose to transfer relational knowledge from the supervised data of existing relations to the unsupervised data of open relations. Given labeled relational instances of existing relations, Wu et al. use a relational Siamese network to learn a metric space. Then, the metric space is transferred to measure the similarities of unlabeled sentences, based on which the clustering is performed. Inspired by Wu et al. [170], Zhang et al. [202] further leverage relation

hierarchies to learn more discriminative metric space for the clustering of relational instance representations, where the instances of the nearby relations on hierarchies are encouraged to share similar representations. Moreover, since relational instance representations contain rich hierarchical information, the newly discovered relations can be directly appended to the existing relation hierarchy. OpenRE can deal with the diversity of relation surface forms by clustering. However, the specific semantics of relation clusters still needs to be summarized through human efforts.

Learning with Increasing Relation Types After discovering novel relations from the open corpora, the relation classifier needs to be updated to deal with both existing and new relations. A straightforward approach is to re-train the relation classifier using all the instances of existing and new relations together from scratch whenever new relations emerge. However, the approach is not feasible due to the high computational cost. Continual relation learning aims to utilize the instances of novel relations to update a relation classifier continually. A significant challenge of continual relation learning is the catastrophic forgetting [159], where the performance on the existing relations can degrade significantly after training with new relations. To address this problem, some works propose saving several instances for existing classes and re-training the classifier with these memorized instances and new data together [30, 159]. This learning process based on memorized instances is named *memory replay*. However, repeatedly updating the classifier with several memorized instances may cause overfitting of existing relations. Drawing inspirations from the study of mammalian memory in neuroscience [10], EMAR [56] proposes episodic memory activation and reconsolidation mechanism to prevent the overfitting problem. The key idea is that the prototypes of the existing relations should remain discriminative after each time of replaying and activating memorized relation instances. In this way, EMAR can flexibly handle new relations without forgetting or overfitting existing relations.

9.5.6 Contextualized Relation Extraction

As mentioned above, RE systems have been significantly improved in supervised and distantly supervised scenarios. To further improve RE performance, many researchers are working on the contextualized RE by integrating multisource information. In this section, we will describe some typical approaches to contextualized RE in detail.

Utilizing External Information Most existing RE systems stated above only concentrate on the text, regardless of the rich external text-related heterogeneous information, like world knowledge in KGs, visual knowledge in images, and structured or semi-structured knowledge on the Web. Text-related heterogeneous information could provide rich additional context. As mentioned in Sect. 9.4.2, Han et al. [58] propose a joint learning framework for RE, the key idea of which is to jointly learn knowledge and text representations within a unified semantic

space via KG-text alignments. In Han's work, for the text part, word and sentence representations are learned via a CNN encoder. For the KG part, entity and relation representations are learned via translation-based methods mentioned in Sect. 9.3.2. The learned representations of the KG and text parts are aligned during the training process, by using entity anchors to share word and entity representations as well as adopting mutual attention to make sentence representations and knowledge representations enhance each other. Apart from this preliminary attempt, many efforts have been devoted to this direction [72, 132, 164, 166].

Incorporating Relational Paths Although existing RE systems have achieved promising results, they still suffer from a major problem: the models can only directly learn from sentences containing both target entities. However, those sentences containing only one of the target entities could also provide helpful information and help build inference chains. For example, if we know that *Alexandre Dumas fils* is the son of *Alexandre Dumas* and *Alexandre Dumas* is the son of *Thomas-Alexandre Dumas*, we can infer that *Alexandre Dumas fils* is the grandson of *Thomas-Alexandre Dumas*. Zeng et al. [199] introduce a path-based RE model incorporating textual relational paths so as to utilize the information of both direct and indirect sentences. The model employs an encoder to represent the semantics of multiple sentences and then builds a relation path encoder to measure the probability distribution of relations given the inference path in text. Finally, the model combines information from both sentences and relational paths and predicts each relation's confidence. This work is the preliminary effort to consider the knowledge of relation paths in text RE. There are also several methods later to consider the reasoning paths of sentence semantic meanings for RE, such as using effective neural models like RNNs to learn relation paths [29], and using distant supervision to annotate implicit relation paths [49] automatically.

9.5.7 Summary

In this section, we elaborate on the approaches to acquiring knowledge. Typically, we focus on RE, and classify RE methods into six groups according to their application scenarios: (1) *sentence-level RE*, which focuses on extracting relations from sentences, (2) *bag-level RE*, which focuses on extracting relations from the bags of sentences annotated by distant supervision, (3) *document-level RE*, which focuses on extracting relations from documents, (4) *few-shot RE*, which focuses on low-resource scenarios, (5) *open-domain RE*, which focuses on continually extracting open-domain relations that are not pre-defined, and (6) *contextualized RE*, which focuses on integrating multisource information for RE.

Note that knowledge acquisition does not just mean RE and includes many other methods, such as KGC, event extraction, etc. Moreover, not all human knowledge is represented in a textual form, and there is also a large amount of knowledge in images, audio, and other knowledge carriers. How to obtain knowledge from these

carriers to empower models is also a problem worthy of further consideration by researchers.

9.6 Summary and Further Readings

We have now overviewed the current progress of using knowledge for NLP tasks, including knowledge representation learning, knowledge-guided NLP, and knowledge acquisition. In this last section, we will summarize the contents of this chapter and then provide more readings for reference.

Knowledge representation learning is a critical component of using knowledge since it bridges the gap between knowledge systems that store knowledge and applications that require knowledge. We systemically describe existing methods for knowledge representation learning. Further, we discuss several advanced approaches that deal with the current challenges of knowledge representation learning. For further understanding of knowledge graph representation learning, more related papers can be found in this paper list.[5] There are also some recommended surveys and books [6, 73, 102, 116, 161].

After introducing knowledge representation learning, we introduce the framework of knowledge-guided learning, aiming to improve NLP models with knowledge representations. The framework includes four important directions: knowledge augmentation, knowledge reformulation, knowledge regularization, and knowledge transfer. All these four directions of knowledge-guided learning have been widely advanced in the past few years. Following these four directions, we review typical cases to clarify this knowledge-guided framework, covering information extraction, information retrieval, language modeling, and text generation. Considering the breakthroughs of PTMs, we also use prompts as an example to show the recent trend of knowledge transfer in the era of PTMs. We suggest readers to find further insights from the recent surveys and books about knowledge and PTMs [9, 60, 193].

Based on knowledge representation learning and knowledge-guided learning, we introduce how to acquire more knowledge from plain text to enrich existing knowledge systems. We systematically review knowledge acquisition methods in various textual scenarios, including sentence-level, bag-level, document-level, few-shot, and open-domain acquisition. In this field, we refer further readings to the paper list[6] and the typical surveys [57, 73].

Acknowledgments The contributions of all authors for the second edition are the following: Zhiyuan Liu, Yankai Lin, and Maosong Sun designed the overall architecture of this chapter; Xu Han and Weize Chen drafted the chapter; Zhengyan Zhang and Yuan Yao participated in and proofread some parts of Sects. 9.4 and 9.5. Zhiyuan Liu and Yankai Lin proofread and revised this chapter.

[5] https://github.com/thunlp/KRLPapers.

[6] https://github.com/thunlp/NREPapers.

We thank Ruobing Xie for providing initial materials in the first edition and also thank Ning Ding, Chaojun Xiao, Shengding Hu, Xinrong Zhang, Qimin Zhan, Bowen Li, and Shihao Liang for proofreading this chapter.

This chapter is the knowledge representation learning chapter of the second edition of the book *Representation Learning for Natural Language Processing*, with its first edition published in 2020 [101]. As compared to the first edition of this chapter, the main changes include the following: (1) we added the section Knowledge-guided NLP and the section Knowledge Acquisition, and (2) we comprehensively supplemented and updated the information, discussions, examples, and figures in other existing sections.

References

1. Sungjin Ahn, Heeyoul Choi, Tanel Pärnamaa, and Yoshua Bengio. A neural knowledge language model. *arXiv preprint arXiv:1608.00318*, 2016.
2. Barr Avron and Edward A Feigenbaum. *The handbook of artificial intelligence*. Addison-Wesley, 1981.
3. Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*, 2015.
4. Ivana Balažević, Carl Allen, and Timothy Hospedales. Multi-relational poincaré graph embeddings. In *Proceedings of NeurIPS*, 2019.
5. Islam Beltagy and Raymond J Mooney. Efficient markov logic inference for natural language semantics. In *Proceedings of AAAI Workshop*, 2014.
6. Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013.
7. Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific american*, 284(5):34–43, 2001.
8. Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of KDD*, 2008.
9. Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.
10. Bruno Bontempi, Catherine Laurent-Demir, Claude Destrade, and Robert Jaffard. Time-dependent reorganization of brain circuitry underlying long-term memory storage. *Nature*, 400(6745):671–675, 1999.
11. Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. Joint learning of words and meaning representations for open-text semantic parsing. In *Proceedings of AISTATS*, 2012.
12. Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Proceedings of NeurIPS*, 2013.
13. Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. Learning structured embeddings of knowledge bases. In *Proceedings of AAAI*, 2011.
14. Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of COLT*, 1992.
15. David Bostock. *Plato's theaetetus*. Oxford University Press, 1988.
16. Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC Press, 1984.
17. Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Proceedings of NeurIPS*, 2020.

18. Razvan C Bunescu and Raymond J Mooney. A shortest path dependency kernel for relation extraction. In *Proceedings of EMNLP*, 2005.
19. Rui Cai, Xiaodong Zhang, and Houfeng Wang. Bidirectional recurrent convolutional neural network for relation classification. In *Proceedings of ACL*, 2016.
20. Andrew Carlson, Justin Betteridge, Richard C Wang, Estevam R Hruschka Jr, and Tom M Mitchell. Coupled semi-supervised learning for information extraction. In *Proceedings of WSDM*, 2010.
21. Mohamed Chabchoub, Michel Gagnon, and Amal Zouaq. Collective disambiguation and semantic annotation for entity linking and typing. In *Proceedings of SWEC*, 2016.
22. Gang Chen, Maosong Sun, and Yang Liu. Towards a universal continuous knowledge base. *AI Open*, 2:197–204, 2021.
23. Weize Chen, Xu Han, Yankai Lin, Hexu Zhao, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. Fully hyperbolic neural networks. In *Proceedings of ACL*, 2022.
24. Noam Chomsky. *Syntactic structures*. De Gruyter, 1957.
25. Noam Chomsky. *Aspects of the Theory of Syntax*. MIT Press, 1965.
26. Lei Cui, Furu Wei, and Ming Zhou. Neural open information extraction. In *Proceedings of ACL*, 2018.
27. Aron Culotta and Jeffrey Sorensen. Dependency tree kernels for relation extraction. In *Proceedings of ACL*, 2004.
28. Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. Convolutional neural networks for soft-matching n-grams in ad-hoc search. In *Proceedings of WSDM*, 2018.
29. 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*, 2017.
30. Cyprien de Masson D'Autume, Sebastian Ruder, Lingpeng Kong, and Dani Yogatama. Episodic memory in lifelong language learning. In *Proceedings of NeurIPS*, 2019.
31. Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proceedings of CVPR*, 2009.
32. Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *Proceedings of AAAI*, 2018.
33. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, 2019.
34. Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models. *arXiv preprint arXiv:2203.06904*, 2022.
35. Bowen Dong, Yuan Yao, Ruobing Xie, Tianyu Gao, Xu Han, Zhiyuan Liu, Fen Lin, Leyu Lin, and Maosong Sun. Meta-information guided meta-learning for few-shot relation classification. In *Proceedings of COLING*, 2020.
36. Li Dong, Furu Wei, Hong Sun, Ming Zhou, and Ke Xu. A hybrid neural model for type classification of entity mentions. In *Proceedings of IJCAI*, 2015.
37. Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of KDD*, 2014.
38. Hady Elsahar, Elena Demidova, Simon Gottschalk, Christophe Gravier, and Frederique Laforest. Unsupervised open relation extraction. In *Proceedings of ESWC*, 2017.
39. Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S Weld. Open information extraction from the web. *Communications of the ACM*, 51(12):68–74, 2008.
40. Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, et al. Open information extraction: The second generation. In *Proceedings of IJCAI*, 2011.
41. Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In *Proceedings of EMNLP*, 2011.

42. Edward A Feigenbaum. Knowledge engineering: The applied side of artificial intelligence. Technical report, Computer Science Department of Stanford University, 1980.
43. Christiane Fellbaum. Wordnet. *The encyclopedia of applied linguistics*, 2012.
44. Jun Feng, Minlie Huang, Li Zhao, Yang Yang, and Xiaoyan Zhu. Reinforcement learning for relation classification from noisy data. In *Proceedings of AAAI*, 2018.
45. Tianyu Gao, Xu Han, Zhiyuan Liu, and Maosong Sun. Hybrid attention-based prototypical networks for noisy few-shot relation classification. In *Proceedings of AAAI*, 2019.
46. 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*, 2019.
47. Alberto García-Durán, Antoine Bordes, and Nicolas Usunier. Composing relationships with translations. In *Proceedings of EMNLP*, 2015.
48. Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020.
49. Michael Glass, Alfio Gliozzo, Oktie Hassanzadeh, Nandana Mihindukulasooriya, and Gaetano Rossiello. Inducing implicit relations from text using distantly supervised deep nets. In *Proceedings of ISWC*, 2018.
50. Kelvin Gu, John Miller, and Percy Liang. Traversing knowledge graphs in vector space. In *Proceedings of EMNLP*, 2015.
51. Yihong Gu, Jun Yan, Hao Zhu, Zhiyuan Liu, Ruobing Xie, Maosong Sun, Fen Lin, and Leyu Lin. Language modeling with sparse product of sememe experts. In *Proceedings of EMNLP*, 2018.
52. Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. Jointly embedding knowledge graphs and logical rules. In *Proceedings of EMNLP*, 2016.
53. Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. Retrieval augmented language model pre-training. In *Proceedings of ICML*, 2020.
54. Michael Haenlein and Andreas Kaplan. A brief history of artificial intelligence: On the past, present, and future of artificial intelligence. *California management review*, 61(4):5–14, 2019.
55. Petr Hájek. *Metamathematics of fuzzy logic*. Springer Science & Business Media, 1998.
56. Xu Han, Yi Dai, Tianyu Gao, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. Continual relation learning via episodic memory activation and reconsolidation. In *Proceedings of ACL*, 2020.
57. Xu Han, Tianyu Gao, Yankai Lin, Hao Peng, Yaoliang Yang, Chaojun Xiao, Zhiyuan Liu, Peng Li, Jie Zhou, and Maosong Sun. More data, more relations, more context and more openness: A review and outlook for relation extraction. In *Proceedings of AACL-IJCNLP*, 2020.
58. Xu Han, Zhiyuan Liu, and Maosong Sun. Neural knowledge acquisition via mutual attention between knowledge graph and text. In *Proceedings of AAAI*, 2018.
59. Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Yuan Yao, Ao Zhang, Liang Zhang, et al. Pre-trained models: Past, present and future. *AI Open*, 2021.
60. Xu Han, Zhengyan Zhang, and Zhiyuan Liu. Knowledgeable machine learning for natural language processing. *Communications of the ACM*, 64(11):50–51, 2021.
61. Xu Han, Zhengyan Zhang, and Zhiyuan Liu. Knowledge-guided pre-trained language models. *ZTE CommunicationsM*, 28(2):10–15, 2022.
62. 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.
63. Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.
64. Frederick Hayes-Roth, Donald A Waterman, and Douglas B Lenat. *Building expert system*. Addison-Wesley, 1983.

65. Shizhu He, Kang Liu, Guoliang Ji, and Jun Zhao. Learning to represent knowledge graphs with gaussian embedding. In *Proceedings of CIKM*, 2015.
66. John Hewitt and Christopher D. Manning. A structural probe for finding syntax in word representations. In *Proceedings of NAACL-HLT*, 2019.
67. Xuming Hu, Lijie Wen, Yusong Xu, Chenwei Zhang, and S Yu Philip. Selfore: Self-supervised relational feature learning for open relation extraction. In *Proceedings of EMNLP*, 2020.
68. Peter Hylton. *Russell, idealism, and the emergence of analytic philosophy*. Oxford University Press, 1990.
69. Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. What does BERT learn about the structure of language? In *Proceedings of ACL*, 2019.
70. Rodolphe Jenatton, Nicolas L Roux, Antoine Bordes, and Guillaume R Obozinski. A latent factor model for highly multi-relational data. In *Proceedings of NeurIPS*, 2012.
71. Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. Knowledge graph completion with adaptive sparse transfer matrix. In *Proceedings of AAAI*, 2016.
72. 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*, 2017.
73. Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and S Yu Philip. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 33(2):494–514, 2021.
74. Zhengbao Jiang, Frank F Xu, Jun Araki, and Graham Neubig. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438, 2020.
75. Nanda Kambhatla. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *Proceedings of ACL*, 2004.
76. Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of EMNLP*, 2020.
77. Keshav Kolluru, Samarth Aggarwal, Vipul Rathore, Soumen Chakrabarti, et al. Imojie: Iterative memory-based joint open information extraction. In *Proceedings of ACL*, 2020.
78. Denis Krompaß, Stephan Baier, and Volker Tresp. Type-constrained representation learning in knowledge graphs. In *Proceedings of ISWC*, 2015.
79. John Lafferty, Andrew McCallum, Fernando Pereira, et al. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*, 2001.
80. Thomas K Landauer and Susan T Dumais. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 1997.
81. Ora Lassila, Ralph R. Swick, World Wide, and Web Consortium. Resource description framework (rdf) model and syntax specification, 1998.
82. Quoc V Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of ICML*, 2014.
83. Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436, 2015.
84. Douglas B Lenat. CYC: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38, 1995.
85. Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. Zero-shot relation extraction via reading comprehension. In *Proceedings of CoNLL*, 2017.
86. Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of NeurIPS*, 2020.
87. Diya Li, Lifu Huang, Heng Ji, and Jiawei Han. Biomedical event extraction based on knowledge-driven tree-LSTM. In *Proceedings of NAACL-HLT*, 2019.

88. 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*, 2016, 05 2016.
89. Yankai Lin, Zhiyuan Liu, Huanbo Luan, Maosong Sun, Siwei Rao, and Song Liu. Modeling relation paths for representation learning of knowledge bases. In *Proceedings of EMNLP*, 2015.
90. Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of AAAI*, 2015.
91. Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. Neural relation extraction with selective attention over instances. In *Proceedings of ACL*, 2016.
92. Chunyang Liu, Wenbo Sun, Wenhan Chao, and Wanxiang Che. Convolution neural network for relation extraction. In *Proceedings of ICDM*, 2013.
93. Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*, 2021.
94. Quan Liu, Hui Jiang, Andrew Evdokimov, Zhen-Hua Ling, Xiaodan Zhu, Si Wei, and Yu Hu. Probabilistic reasoning via deep learning: Neural association models. *arXiv preprint arXiv:1603.07704*, 2016.
95. Tianyu Liu, Kexiang Wang, Baobao Chang, and Zhifang Sui. A soft-label method for noise-tolerant distantly supervised relation extraction. In *Proceedings of EMNLP*, 2017.
96. Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. K-BERT: Enabling language representation with knowledge graph. In *Proceedings of AAAI*, 2020.
97. Xiao Liu, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang. Self-supervised learning: Generative or contrastive. *IEEE Transactions on Knowledge and Data Engineering*, 2021.
98. Yang Liu, Kang Liu, Liheng Xu, Jun Zhao, et al. Exploring fine-grained entity type constraints for distantly supervised relation extraction. In *Proceedings of COLING*, 2014.
99. 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.
100. Zhenghao Liu, Chenyan Xiong, Maosong Sun, and Zhiyuan Liu. Entity-duet neural ranking: Understanding the role of knowledge graph semantics in neural information retrieval. In *Proceedings of ACL*, 2018.
101. Zhiyuan Liu, Yankai Lin, and Maosong Sun. *Representation Learning for Natural Language Processing*. Springer, 2020.
102. Zhiyuan Liu, Maosong Sun, Yankai Lin, and Ruobing Xie. Knowledge representation learning: A review. *JCRD*, 53(2):247–261, 2016.
103. Diego Marcheggiani and Ivan Titov. Discrete-state variational autoencoders for joint discovery and factorization of relations. *Transactions of the Association for Computational Linguistics*, 4:231–244, 2016.
104. J McCarthy, ML Minsky, and N Rochester. A proposal for the dartmouth summer research project on artificial intelligence. 1955.
105. John McCarthy. History of lisp. *ACM SIGPLAN Notices*, 13(8):217–223, 1978.
106. Pablo N Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. DBpedia spotlight: shedding light on the web of documents. In *Proceedings of ICSS*, 2011.
107. T Mikolov and J Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of NeurIPS*, 2013.
108. Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Proceedings of ICLR*, 2013.
109. Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of ACL-IJCNLP*, 2009.

110. Makoto Miwa and Mohit Bansal. End-to-end relation extraction using LSTMs on sequences and tree structures. In *Proceedings of ACL*, 2016.
111. Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. WebGPT: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.
112. Guoshun Nan, Zhijiang Guo, Ivan Sekulić, and Wei Lu. Reasoning with latent structure refinement for document-level relation extraction. In *Proceedings of ACL*, 2020.
113. Allen Newell, John Clifford Shaw, and Herbert A Simon. Empirical explorations of the logic theory machine: a case study in heuristic. In *Proceedings of Western Computer*, 1957.
114. Allen Newell and Fred M Tonge. An introduction to information processing language v. *Communications of the ACM*, 3(4):205–211, 1960.
115. Thien Huu Nguyen and Ralph Grishman. Combining neural networks and log-linear models to improve relation extraction. *arXiv preprint arXiv:1511.05926*, 2015.
116. Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. In *Proceedings of the IEEE*, 2015.
117. Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. Holographic embeddings of knowledge graphs. In *Proceedings of AAAI*, 2016.
118. Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *Proceedings of ICML*, 2011.
119. Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. Factorizing YAGO: scalable machine learning for linked data. In *Proceedings of WWW*, 2012.
120. Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *TKDE*, 22(10):1345–1359, 2009.
121. Sachin Pawar, Girish K Palshikar, and Pushpak Bhattacharyya. Relation extraction: A survey. *arXiv preprint arXiv:1712.05191*, 2017.
122. 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.
123. Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of EMNLP*, 2014.
124. 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.
125. Fabio Petroni, Patrick Lewis, Aleksandra Piktus, Tim Rocktäschel, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. How context affects language models' factual predictions. In *Proceedings of AKBC*, 2020.
126. Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. Language models as knowledge bases? In *Proceedings of EMNLP-IJCNLP*, 2019.
127. Jay Pujara, Hui Miao, Lise Getoor, and William W Cohen. Knowledge graph identification. In *Proceedings of ISWC*, 2013.
128. Yujia Qin, Yankai Lin, Ryuichi Takanobu, Zhiyuan Liu, Peng Li, Heng Ji, Minlie Huang, Maosong Sun, and Jie Zhou. Erica: Improving entity and relation understanding for pre-trained language models via contrastive learning. In *Proceedings of ACL-IJCNLP*, 2021.
129. Chris Quirk and Hoifung Poon. Distant supervision for relation extraction beyond the sentence boundary. In *Proceedings of EACL*, 2017.
130. Arlan Ramsay and Robert D Richtmyer. *Introduction to hyperbolic geometry*. Springer Science & Business Media, 1995.
131. Erich H Reck. *From Frege to Wittgenstein: Perspectives on Early Analytic Philosophy*. Oxford University Press, 2001.
132. Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. Relation extraction with matrix factorization and universal schemas. In *Proceedings of NAACL-HLT*, 2013.

133. Dan Roth and Wen-tau Yih. Probabilistic reasoning for entity & relation recognition. In *Proceedings of COLING*, 2002.
134. Dan Roth and Wen-tau Yih. A linear programming formulation for global inference in natural language tasks. In *Proceedings of CoNLL*, 2004.
135. Cicero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. Classifying relations by ranking with convolutional neural networks. In *Proceedings of ACL-IJCNLP*, 2015.
136. Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *Proceedings of ESWC*, 2018.
137. Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.
138. Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. An attentive neural architecture for fine-grained entity type classification. In *Proceedings of AKBC Workshop*, 2016.
139. Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. Retrieval augmentation reduces hallucination in conversation. In *Findings of EMNLP*, 2021.
140. Barry Smith. Ontology. In *The furniture of the world*, pages 47–68. Brill, 2012.
141. Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Proceedings of NeurIPS*, 2017.
142. 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.
143. Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. In *Proceedings of NeurIPS*, 2013.
144. Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of EMNLP*, 2012.
145. Gabriel Stanovsky, Julian Michael, Luke Zettlemoyer, and Ido Dagan. Supervised open information extraction. In *Proceedings of NAACL-HLT*, 2018.
146. Matthias Steup and Ram Neta. Epistemology. 2005.
147. Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. YAGO: a core of semantic knowledge. In *Proceedings of WWW*, 2007.
148. Tony Sun, Andrew Gaut, Shirlyn Tang, Yuxin Huang, Mai ElSherief, Jieyu Zhao, Diba Mirza, Elizabeth Belding, Kai-Wei Chang, and William Yang Wang. Mitigating gender bias in natural language processing: Literature review. In *Proceedings of ACL*, 2019.
149. Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. RotatE: Knowledge graph embedding by relational rotation in complex space. In *Proceedings of ICLR*, 2019.
150. Kumutha Swampillai and Mark Stevenson. Inter-sentential relations in information extraction corpora. In *Proceedings of LREC*, 2010.
151. 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.
152. Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. LINE: Large-scale information network embedding. In *Proceedings of WWW*, 2015.
153. Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *Proceedings of ICML*, 2016.
154. 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.
155. 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.
156. Denny Vrandečić and Markus Krötzsch. Wikidata: A free collaborative knowledge base. *Communications of the ACM*, 57(10):78–85, 2014.
157. Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. Universal adversarial triggers for attacking and analyzing nlp. In *Proceedings of EMNLP-IJCNLP*, 2019.

158. Hong Wang, Christfried Focke, Rob Sylvester, Nilesh Mishra, and William Wang. Fine-tune BERT for DocRED with two-step process. *arXiv preprint arXiv:1909.11898*, 2019.
159. Hong Wang, Wenhan Xiong, Mo Yu, Xiaoxiao Guo, Shiyu Chang, and William Yang Wang. Sentence embedding alignment for lifelong relation extraction. In *Proceedings of NAACL-HLT*, 2019.
160. Qingyun Wang, Lifu Huang, Zhiying Jiang, Kevin Knight, Heng Ji, Mohit Bansal, and Yi Luan. PaperRobot: Incremental draft generation of scientific ideas. In *Proceedings of ACL*, 2019.
161. Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, 2017.
162. Quan Wang, Bin Wang, and Li Guo. Knowledge base completion using embeddings and rules. In *Proceedings of IJCAI*, 2015.
163. Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. KEPLER: A unified model for knowledge embedding and pre-trained language representation. *Transactions of the Association for Computational Linguistics*, 9:176–194, 2021.
164. Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph and text jointly embedding. In *Proceedings of EMNLP*, 2014.
165. Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of AAAI*, 2014.
166. Zhigang Wang and Juan-Zi Li. Text-enhanced representation learning for knowledge graph. In *Proceedings of IJCAI*, 2016.
167. Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.
168. Jason Weston, Emily Dinan, and Alexander H Miller. Retrieve and refine: Improved sequence generation models for dialogue. In *Proceedings of EMNLP*, 2018.
169. Fei Wu and Daniel S Weld. Open information extraction using wikipedia. In *Proceedings of ACL*, 2010.
170. Ruidong Wu, Yuan Yao, Xu Han, Ruobing Xie, Zhiyuan Liu, Fen Lin, Leyu Lin, and Maosong Sun. Open relation extraction: Relational knowledge transfer from supervised data to unsupervised data. In *Proceedings of EMNLP-IJCNLP*, 2019.
171. Yi Wu, David Bamman, and Stuart Russell. Adversarial training for relation extraction. In *Proceedings of EMNLP*, 2017.
172. Chaojun Xiao, Yuan Yao, Ruobing Xie, Xu Han, Zhiyuan Liu, Maosong Sun, Fen Lin, and Leyu Lin. Denoising relation extraction from document-level distant supervision. In *Proceedings of EMNLP*, 2020.
173. Han Xiao, Minlie Huang, and Xiaoyan Zhu. From one point to a manifold: Knowledge graph embedding for precise link prediction. In *Proceedings of IJCAI*, 2016.
174. Han Xiao, Minlie Huang, and Xiaoyan Zhu. TransG: A generative model for knowledge graph embedding. In *Proceedings of ACL*, 2016.
175. Ruobing Xie, Zhiyuan Liu, Tat-seng Chua, Huanbo Luan, and Maosong Sun. Image-embodied knowledge representation learning. In *Proceedings of IJCAI*, 2016.
176. Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. Representation learning of knowledge graphs with entity descriptions. In *Proceedings of AAAI*, 2016.
177. Ruobing Xie, Zhiyuan Liu, and Maosong Sun. Representation learning of knowledge graphs with hierarchical types. In *Proceedings of IJCAI*, 2016.
178. Ji Xin, Yankai Lin, Zhiyuan Liu, and Maosong Sun. Improving neural fine-grained entity typing with knowledge attention. In *Proceedings of AAAI*, 2018.
179. Chenyan Xiong, Jamie Callan, and Tie-Yan Liu. Word-entity duet representations for document ranking. In *Proceedings of SIGIR*, 2017.
180. Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. End-to-end neural ad-hoc ranking with kernel pooling. In *Proceedings of SIGIR*, 2017.

181. Wenhan Xiong, Jingfei Du, William Yang Wang, and Veselin Stoyanov. Pretrained encyclopedia: Weakly supervised knowledge-pretrained language model. In *Proceedings of ICLR*, 2020.
182. 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.
183. Yan Xu, Ran Jia, Lili Mou, Ge Li, Yunchuan Chen, Yangyang Lu, and Zhi Jin. Improved relation classification by deep recurrent neural networks with data augmentation. In *Proceedings of COLING*, 2016.
184. Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, and Gerhard Weikum. Robust question answering over the web of linked data. In *Proceedings of CIKM*, 2013.
185. Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. Luke: Deep contextualized entity representations with entity-aware self-attention. In *Proceedings of EMNLP*, 2020.
186. Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In *Proceedings of ICLR*, 2015.
187. Yuan Yao, Jiaju Du, Yankai Lin, Peng Li, Zhiyuan Liu, Jie Zhou, and Maosong Sun. Codred: A cross-document relation extraction dataset for acquiring knowledge in the wild. In *Proceedings of EMNLP*, 2021.
188. 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.
189. Alexander Yates, Michele Banko, Matthew Broadhead, Michael J Cafarella, Oren Etzioni, and Stephen Soderland. Textrunner: open information extraction on the web. In *Proceedings of NAACL-HLT*, 2007.
190. Deming Ye, Yankai Lin, Jiaju Du, Zhenghao Liu, Peng Li, Maosong Sun, and Zhiyuan Liu. Coreferential reasoning learning for language representation. In *Proceedings of EMNLP*, 2020.
191. Zhi-Xiu Ye and Zhen-Hua Ling. Multi-level matching and aggregation network for few-shot relation classification. In *Proceedings of ACL*, 2019.
192. David Yenicelik, Florian Schmidt, and Yannic Kilcher. How does BERT capture semantics? a closer look at polysemous words. In *Proceedings of BlackboxNLP*, 2020.
193. Wenhao Yu, Chenguang Zhu, Zaitang Li, Zhiting Hu, Qingyun Wang, Heng Ji, and Meng Jiang. A survey of knowledge-enhanced text generation. *ACM Computing Surveys (CSUR)*, 54(11):1–38, 2022.
194. Yuan Zang, Fanchao Qi, Chenghao Yang, Zhiyuan Liu, Meng Zhang, Qun Liu, and Maosong Sun. Word-level textual adversarial attacking as combinatorial optimization. In *Proceedings of ACL*, 2020.
195. Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. Kernel methods for relation extraction. *JMLR*, 3:1083–1106, 2003.
196. Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of EMNLP*, 2015.
197. Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. Relation classification via convolutional deep neural network. In *Proceedings of COLING*, 2014.
198. Shuang Zeng, Runxin Xu, Baobao Chang, and Lei Li. Double graph based reasoning for document-level relation extraction. In *Proceedings of EMNLP*, 2020.
199. Wenyuan Zeng, Yankai Lin, Zhiyuan Liu, and Maosong Sun. Incorporating relation paths in neural relation extraction. In *Proceedings of EMNLP*, 2017.
200. Xiangrong Zeng, Shizhu He, Kang Liu, and Jun Zhao. Large scaled relation extraction with reinforcement learning. In *Proceedings of AAAI*, 2018.
201. Dongxu Zhang and Dong Wang. Relation classification via recurrent neural network. *arXiv preprint arXiv:1508.01006*, 2015.
202. Kai Zhang, Yuan Yao, Ruobing Xie, Xu Han, Zhiyuan Liu, Fen Lin, Leyu Lin, and Maosong Sun. Open hierarchical relation extraction. In *Proceedings of NAACL-HLT*, 2021.

203. Shuai Zhang, Yi Tay, Lina Yao, and Qi Liu. Quaternion knowledge graph embeddings. In *Proceedings of NeurIPS*, 2019.
204. Yuhao Zhang, Peng Qi, and Christopher D Manning. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of EMNLP*, 2018.
205. 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.
206. Zhengyan Zhang, Xu Han, Hao Zhou, Pei Ke, Yuxian Gu, Deming Ye, Yujia Qin, Yusheng Su, Haozhe Ji, Jian Guan, et al. CPM: A large-scale generative chinese pre-trained language model. *AI Open*, 2:93–99, 2021.
207. Huaping Zhong, Jianwen Zhang, Zhen Wang, Hai Wan, and Zheng Chen. Aligning knowledge and text embeddings by entity descriptions. In *Proceedings of EMNLP*, 2015.
208. Guodong Zhou, Jian Su, Jie Zhang, and Min Zhang. Exploring various knowledge in relation extraction. In *Proceedings of ACL*, 2005.
209. Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of ACL*, 2016.
210. Hao Zhu, Yankai Lin, Zhiyuan Liu, Jie Fu, Tat-Seng Chua, and Maosong Sun. Graph neural networks with generated parameters for relation extraction. In *Proceedings of ACL*, 2019.