Chapter 7
World Knowledge Representation

Abstract  World knowledge representation aims to represent entities and relations in the knowledge graph in low-dimensional semantic space, which have been widely used in large knowledge-driven tasks. In this chapter, we first introduce the concept of the knowledge graph. Next, we introduce the motivations and give an overview of the existing approaches for knowledge graph representation. Further, we discuss several advanced approaches that aim to deal with the current challenges of knowledge graph representation. We also review the real-world applications of knowledge graph representation, such as language modeling, question answering, information retrieval, and recommender systems.

7.1 Introduction

Knowledge Graph (KG), which is also named as Knowledge Base (KB), is a significant multi-relational dataset for modeling concrete entities and abstract concepts in the real world. It provides useful structured information and plays a crucial role in lots of real-world applications such as web search and question answering. It is not exaggerated to say that knowledge graphs teach us how to model the entities as well as the relationships among them in this complicated real world.

To encode knowledge into a real-world application, knowledge graph representation, which represents entities and relations in knowledge graphs with distributed representations, has been proposed and applied to various real-world artificial intelligence fields including question answering, information retrieval, and dialogue system. That is, knowledge graph representation learning plays a vital role as a bridge between knowledge graphs and knowledge-driven tasks.

In this section, we will introduce the concept of knowledge graph, several typical knowledge graphs, knowledge graph representation learning, and several typical knowledge-driven tasks.
7.1.1 World Knowledge Graphs

In ancient times, knowledge was stored and inherited through books and letters written on parchment or bamboo slip. With the Internet thriving in the twenty-first century, millions of thousands of messages have flooded into the World Wide Web, and knowledge was transferred to the semi-structured textual information on the web. However, due to the information explosion, it is not easy to extract knowledge we want from the significant, noisy plain text on the Internet. To obtain knowledge effectively, people notice that the world is not only made of strings but also made of entities and relations. Knowledge Graph, which arranges structured multi-relational data of concrete entities and abstract concepts in the real world, is blooming in recent years and attracts wide attention in both academia and industry.

KGs are usually constructed from existing Semantic Web datasets in Resource Description Framework (RDF) with the help of manual annotation, while it can also be automatically enriched by extracting knowledge from large plain texts on the Internet. A typical KG usually contains two elements, including entities (i.e., concrete entities and abstract concepts in the real world) and relations between entities. It usually represents knowledge with large quantities of triple facts in the triple form of \( \langle \text{head entity}, \text{relation}, \text{tail entity} \rangle \) abridged as \( \langle h, r, t \rangle \). For example, William Shakespeare is a famous English poet and playwright, who is widely regarded as the greatest writer in the English language, and Romeo and Juliet is one of his masterpieces. In knowledge graph, we will represent this knowledge as (William Shakespeare, works_written, Romeo and Juliet). Note that in the real world, the same head entity and relation may have multiple tail entities (e.g., William Shakespeare also wrote Hamlet and A Midsummer Night’s Dream), and conversely the same situation will happen when tail entity and relation are fixed. Even it is possible when both the head entity and tail entity are multiple (e.g., in relations like actor_in_movie). However, in KG, all knowledge can be represented in triple facts regardless of the types of entities and relations. Through these triples, we can generate a huge directed graph whose nodes correspond to entities and edges correspond to relations to model the real world. With the well-structured united knowledge representation, KGs are widely used in a variety of applications to enhance their system performance.

There are several KGs widely utilized nowadays in applications of information retrieval and question answering. In this subsection, we will introduce some famous KGs such as Freebase, DBpedia, Yago, and WordNet. In fact, there are also lots of comparatively smaller KGs in specific fields of knowledge functioned in vertical search.

7.1.1.1 Freebase

Freebase is one of the most popular knowledge graphs in the world. It is a large community-curated database consisting of well-known people, places, and things,
which is composed of existing databases and its community members. Freebase was first developed by Metaweb, an American software company, and ran since March 2007. In July 2010, Metaweb was acquired by Google, and Freebase was combined to power up Google’s Knowledge Graph. In December 2014, the Freebase team officially announced that the website, as well as the API of Freebase, would be shut down by June 30, 2015. While the data in Freebase would be transferred to Wikidata, which is another collaboratively edited knowledge base operated by Wikimedia Foundation. Up to March 24, 2016, Freebase arranged 58,726,427 topics and 3,197,653,841 facts.

Freebase contains well-structured data representing relationships between entities as well as the attributes of entities in the form of triple facts (Fig. 7.1). Data in Freebase was mainly harvested from various sources, including Wikipedia, Fashion Model Directory, NNDB, MusicBrainz, and so on. Moreover, the community members also contributed a lot to Freebase. Freebase is an open and shared database that aims to construct a global database which encodes the world’s knowledge. It announced an open API, RDF endpoint, and a database dump for its users for both commercial and noncommercial use. As described by Tim O’Reilly, Freebase is the bridge between the bottom-up vision of Web 2.0 collective intelligence and the more structured world of the Semantic Web.

7.1.1.2 DBpedia

DBpedia is a crowd-sourced community effort aiming to extract structured content from Wikipedia and make this information accessible on the web. It was started by researchers at Free University of Berlin, Leipzig University and OpenLink Software,
initially released to the public in January 2007. DBpedia allows users to ask semantic queries associated with Wikipedia resources, even including links to other related datasets, which makes it easier for us to fully utilize the massive amount of information in Wikipedia in a novel and effective way. DBpedia is also an essential part of the Linked Data effort described by Tim Berners-Lee.

The English version of DBpedia describes 4.58 million entities, out of which 4.22 million are classified in a consistent ontology, including 1,445,000 persons, 735,000 places, 411,000 creative works, 251,000 species, 241,000 organizations, and 6,000 diseases. There are also localized versions of DBpedia in 125 languages, all of which contain 38.3 million entities. Besides, DBpedia also contains a great number of internal and external links, including 80.9 million links to Wikipedia categories, 41.2 million links to YAGO categories, 25.2 million links to images, and 29.8 million links to external web pages. Moreover, DBpedia maintains a hierarchical, cross-domain ontology covering overall 685 classes, which has been manually created based on the commonly used infoboxes in Wikipedia.

DBpedia has several advantages over other KGs. First, DBpedia has a close connection to Wikipedia and can automatically evolve as Wikipedia changes. It makes the update process of DBpedia more efficient. Second, DBpedia is multilingual that is convenient for users over the world with their native languages.

7.1.1.3 YAGO

YAGO, which is short for Yet Another Great Ontology, is a high-quality KG developed by Max Planck Institute for Computer Science in Saarbrücken initially released in 2008. Knowledge in YAGO is automatically extracted from Wikipedia, WordNet, and GeoNames, whose accuracy has been manually evaluated and proves a confirmed accuracy of 95%. YAGO is special not only because of the confidence value every fact possesses depending on the manual evaluation but also because that YAGO is anchored in space and time, which can provide a spatial dimension or temporal dimension to part of its entities.

Currently, YAGO has more than 10 million entities, including persons, organizations, and locations, with over 120 million facts about these entities. YAGO also combines knowledge extracted from Wikipedias of 10 different languages and classifies them into approximately 350,000 classes according to the Wikipedia category system and the taxonomy of WordNet. YAGO has also joined the linked data project and been linked to the DBpedia ontology and the SUMO ontology (Fig. 7.2).

7.2 Knowledge Graph Representation

Knowledge Graphs provide us with a novel aspect to describe the world with entities and triple facts, which attract growing attention from researchers. Large KGs such as Freebase, DBpedia, and YAGO have been constructed and widely used in an enormous amount of applications such as question answering and Web search.
However, with KG size increasing, we are facing two main challenges: data sparsity and computational inefficiency. Data sparsity is a general problem in lots of fields like social network analysis or interest mining. It is because that there are too many nodes (e.g., users, products, or entities) in a large graph, while too few edges (e.g., relationships) between these nodes, since the number of relations of a node is limited in the real world. Computational efficiency is another challenge we need to overcome with the increasing size of knowledge graphs.

To tackle these problems, representation learning is introduced to knowledge representation. Representation learning in KGs aims to project both entities and relations into a low-dimensional continuous vector space to get their distributed representations, whose performance has been confirmed in word representation and social representation. Compared with the traditional one-hot representation, distributed representation has much fewer dimensions, and thus lowers the computational complexity. What is more, distributed representation can explicitly show the similarity between entities through some distance calculated by the low-dimensional embeddings, while all embeddings in one-hot representation are orthogonal, making it difficult to tell the potential relations between entities.

With the advantages above, knowledge graph representation learning is blooming in knowledge applications, significantly improving the ability of KGs on the task of knowledge completion, knowledge fusion, and reasoning. It is considered as the bridge between knowledge construction, knowledge graphs, and knowledge-driven applications. Up till now, a high number of methods have been proposed using a distributed representation for modeling knowledge graphs, with the learned knowledge representations widely utilized in various knowledge-driven tasks like question answering, information retrieval, and dialogue system.
In summary, Knowledge graph Representation Learning (KRL) aims to construct distributed knowledge representations for entities and relations, projecting knowledge into low-dimensional semantic vector spaces. Recent years have witnessed significant advances in knowledge graph representation learning with a large amount of KRL methods proposed to construct knowledge representations, among which the translation-based methods achieve state-of-the-art performance in many KG tasks, with a right balance in both effectiveness and efficiency.

In this section, we will first describe the notations that we will use in KRL. Then, we will introduce TransE, which is the fundamental version of translation-based methods. Next, we will explore the various extension methods of TransE in detail. At last, we will take a brief look over other representation learning methods utilized in modeling knowledge graphs.

### 7.2.1 Notations

First, we introduce the general notations used in the rest of this section. We use $G = (E, R, T)$ to denote the whole KG, in which $E = \{e_1, e_2, \ldots, e_{|E|}\}$ stands for the entity set, $R = \{r_1, r_2, \ldots, r_{|R|}\}$ stands for the relation set, and $T$ stands for the triple set. $|E|$ and $|R|$ are the corresponding entity and relation numbers in their overall sets. As stated above, we represent knowledge in the form of triple fact $\langle h, r, t \rangle$, where $h \in E$ means the head entity, $t \in E$ means the tail entity, and $r \in R$ means the relation between $h$ and $t$.

### 7.2.2 TransE

TransE [7] is a translation-based model for learning low-dimensional embeddings of entities and relations. It projects entities as well as relations into the same semantic embedding space, and then considers relations as translations in the embedding space. First, we will start with the motivations of this method, and then discuss the details in how knowledge representations are trained under TransE. Finally, we will explore the advantages and disadvantages of TransE for a deeper understanding.

#### 7.2.2.1 Motivation

There are three main motivations behind the translation-based knowledge graph representation learning method. The primary motivation is that it is natural to consider relationships between entities as translating operations. Through distributed representations, entities are projected to a low-dimensional vector space. Intuitively, we agree that a reasonable projection should map entities with similar semantic meanings to the same field, while entities with different meanings should belong to
distinct clusters in the vector space. 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 works_written, and the translations between writers and books in the vector space are similar.

The secondary motivation of TransE derives from the breakthrough in word representation by Word2vec [49]. Word2vec proposes two simple models, Skip-gram and CBOW, to learn word embeddings from large-scale corpora, significantly improving the performance in word similarity and analogy. The word embeddings learned by Word2vec have some interesting phenomena: if two word-pairs share the same semantic or syntactic relationships, their subtraction embeddings in each word pair will be similar. For instance, we have

\[ w(\text{king}) - w(\text{man}) \approx w(\text{queen}) - w(\text{woman}), \] (7.1)

which indicates that the latent semantic relation between king and man, which is similar to the relation between queen and woman, is successfully embedded in the word representation. This approximate relation could be found not only with the semantic relations but also with the syntactic relations. We have

\[ w(\text{bigger}) - w(\text{big}) \approx w(\text{smaller}) - w(\text{small}). \] (7.2)

The phenomenon found in word representation strongly implies that there may exist an explicit method to represent relationships between entities as translating operations in vector space.

The last motivation comes from the consideration of computational complexity. On the one hand, a substantial increase in model complexity will result in high computational costs and obscure model interpretability. Moreover, a complex model may lead to overfitting. On the other hand, experimental results on model complexity demonstrate that the simpler models perform almost as good as more expressive models in most KG applications, in the condition that there are sizeable multi-relational dataset and a relatively large amount of relations. As KG size increases, computational complexity becomes the primary challenge in the knowledge graph representation. The intuitive assumption of translation leads to a better trade-off between accuracy and efficiency.

7.2.2.2 Methodology

As illustrated in Fig. 7.3, TransE projects entities and relations into the same low-dimensional space. All embeddings take values in \( \mathbb{R}^d \), where \( d \) is a hyperparameter indicating the dimension of embeddings. With the translation assumption, for each triple \( \langle h, r, t \rangle \) in \( T \), we want the summation embedding \( h + r \) to be the nearest neighbor of tail embedding \( t \). The score function of TransE is then defined as follows:
More specifically, to learn such embeddings of entities and relations, TransE formalizes a margin-based loss function with negative sampling as objective for training. The pair-wise function is defined as follows:

$$\mathcal{L} = \sum_{(h,r,t) \in T} \sum_{(h',r',t') \in T^-} \max(\gamma + \mathcal{E}(h, r, t)) - \mathcal{E}(h', r', t'), 0),$$

(7.4)

in which $\mathcal{E}(h, r, t)$ is the score of energy function for a positive triple (i.e., triple in $T$) and $\mathcal{E}(h', r', t')$ is that of a negative triple. The energy function $\mathcal{E}$ can be either measured by $L_1$ or $L_2$ distance. $\gamma > 0$ is a hyperparameter of margin and a bigger $\gamma$ means a wider gap between positive and the corresponding negative scores. $T^-$ is the negative triple set with respect to $T$.

Since there are no explicit negative triples in knowledge graphs, we define $T^-$ as follows:

$$T^- = \{(h', r, t)|h' \in E\} \cup \{(h, r', t)|r' \in R\} \cup \{(h, r, t')|t' \in E\}, \quad (h, r, t) \in T,$$

(7.5)

which means the negative triple set $T^-$ is composed of the positive triple $(h, r, t)$ with head entity, relation, or tail entity randomly replaced by any other entities or relations in KG. Note that the new triple generated after replacement will not be considered as a negative sample if it has already been in $T$.

TransE is optimized using mini-batch stochastic gradient descent (SGD), with entities and relations randomly initialized. Knowledge completion, which is a link prediction task aiming to predict the third element in a triple (could be either entity or relation) with the given rest two elements, is designed to evaluate the learned knowledge representations.
7.2.2.3 Disadvantages and Challenges

TransE is effective and efficient and has shown its power on link prediction. However, it still has several disadvantages and challenges to be further explored.

First, in knowledge completion, we may have multiple correct answers with the given two elements in a triple. For instance, with the given head entity William Shakespeare and the relation works_written, we will get a list of masterpieces including Romeo and Juliet, Hamlet and A Midsummer Night’s Dream. These books share the same information in the writer while differing in many other fields such as theme, background, and famous roles in the book. However, with the translation assumption in TransE, every entity has only one embedding in all triples, which significantly limits the ability of TransE in knowledge graph representations. In [7], the authors categorize all relations into four classes, 1-to-1, 1-to-Many, Many-to-1, Many-to-Many, according to the cardinalities of their head and tail arguments. A relation is considered as 1-to-1 if most heads appear with one tail, 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, Many-to-Many relations occupy a large proportion. TransE does well in 1-to-1, but it has issues when dealing with 1-to-Many, Many-to-1, Many-to-Many relations. Similarly, TransE may also struggle with reflexive relations.

Second, the translating operation is intuitive and effective, only considering the simple one-step translation, which may limit the ability to model KGs. Taking entities as nodes and relations as edges, we can construct a huge knowledge graph with the triple facts. However, TransE focuses on minimizing the energy function \( E(h, r, t) = \| h + r - t \| \), which only utilize the one-step relation information in knowledge graphs, regardless of the latent relationships located in long-distance paths. For example, if we know the triple fact that (The forbidden city, locate_in, Beijing) and (Beijing, capital_of, China), 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 dissimilarity function in TransE are oversimplified for the consideration of efficiency. Therefore, TransE may not be capable enough of modeling those complicated entities and relations in knowledge graphs. There still exist challenges on how to balance the effectiveness and efficiency, avoiding both overfitting and underfitting.

Besides the disadvantages and challenges stated above, multisource information such as textual information and hierarchical type/label information is of great significance, which will be further discussed in the following.
7.2.3 Extensions of TransE

There are lots of extension methods following TransE to address the challenges above. Specifically, TransH, TransR, TransD, and TranSparse are proposed to solve the challenges in modeling 1-to-Many, Many-to-1, and Many-to-Many relations, PTransE is proposed to encode long-distance information located in multistep paths, and CTransR, TransA, TransG, and KG2E further extend the oversimplified model of TransE. We will discuss these extension methods in detail.

7.2.3.1 TransH

With distributed representation, entities are projected to the semantic vector 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 next to Mark Twain when talking about occupation. To accomplish this, we want entities to show different preferences in different situations, that is, to have multiple representations in different triples.

To address the issue when modeling 1-to-Many, Many-to-1, Many-to-Many, and reflexive relations, TransH [77] enables an entity to have multiple representations when involved in different relations. As illustrated in Fig. 7.4, TransH proposes a relation-specific hyperplane $w_r$ for each relation, and judge dissimilarities on the hyperplane instead of the original vector space of entities. Given a triple $\langle h, r, t \rangle$, TransH first projects $h$ and $t$ to the corresponding hyperplane $w_r$ to get the projection $h_\perp$ and $t_\perp$, and the translation vector $r$ is used to connect $h_\perp$ and $t_\perp$ on the hyperplane.

The score function is defined as follows:

$$E(h, r, t) = \|h_\perp + r - t_\perp\|,$$

(7.6)

in which we have

$$h_\perp = h - w_r^\top hw_r, \quad t_\perp = t - w_r^\top tw_r,$$

(7.7)

where $w_r$ is a vector and $\|w_r\|_2$ is restricted to 1. As for training, TransH also minimizes the margin-based loss function with negative sampling which is similar to TransE, and use mini-batch SGD to learn representations.

7.2.3.2 TransR/CTransR

TransH enables entities to have multiple representations in different relations with the favor of hyperplanes, while entities and relations are still restricted in the same semantic vector space, which may limit the ability for modeling entities and relations. TransR [39] assumes that entities and relations should be arranged in distinct spaces, that is, entity space for all entities and relation space for each relation.
As illustrated in Fig. 7.5, for a triple \( \langle h, r, t \rangle \), \( h, t \in \mathbb{R}^k \) and \( r \in \mathbb{R}^d \), TransR first projects \( h \) and \( t \) from entity space to the corresponding relation space of \( r \). That is to say, every entity has a relation-specific representation for each relation, and the translating operation is processed in the specific relation space. The energy function of TransR is defined as follows:

\[
E(h, r, t) = \| h_r + r - t_r \|, \tag{7.8}
\]

where \( h_r \) and \( t_r \) stand for the relation-specific representation for \( h \) and \( t \) in the corresponding relation space of \( r \). The projection from entity space to relation space is

\[
h_r = hM_r, \quad t_r = tM_r, \tag{7.9}
\]

where \( M_r \in \mathbb{R}^{k \times d} \) is a projection matrix mapping entities from the entity space to the relation space of \( r \). TransR also constrains the norms of the embeddings and has \( \| h \|_2 \leq 1, \| t \|_2 \leq 1, \| r \|_2 \leq 1, \| h_r \|_2 \leq 1, \| t_r \|_2 \leq 1 \). As for training, TransR shares the same margin-based score function as TransE.

Furthermore, the author found that some relations in knowledge graphs could be divided into a few sub-relations that give more precise information. The differences between those sub-relations can be learned from corresponding entity pairs. For instance, the relation location contains has head-tail patterns like city-street, country-city, and even country-university, showing different attributes in cognition. With the sub-relations being considered, entities may be projected to more precise positions in the semantic vector space.

Cluster-based TransR (CTransR), which is an enhanced version of TransR with the sub-relations into consideration, is then proposed. More specifically, for each relation \( r \), all entity pairs \( \langle h, t \rangle \) are first clustered into several groups. The clustering of entity pairs depends on the subtraction result of \( t - h \), in which \( h \) and \( t \) are pretrained by TransE. Next, we learn a distinct sub-relation vector \( r_c \) for each cluster according to the corresponding entity pairs, and the original energy function is modified as
7.2.3.3 TransD

TransH and TransR focus on the multiple representations of entities in different relations, improving the performance on knowledge completion and triple classification. However, both models only project entities according to the relations in triples, ignoring the diversity of entities. Moreover, the projection operation with matrix-vector multiplication leads to a higher computational complexity compared to TransE, which is time consuming when applied on large-scale graphs. To address this problem, TransD [32] proposes a novel projection method with a dynamic mapping matrix depending on both entity and relation, which takes the diversity of entities as well as relations into consideration.

TransD defines two vectors for each entity and relation, i.e., the original vector that is also used in TransE, TransH, and TransR for distributed representation of entities and relations, and the projection vector that is used in constructing projection matrices for mapping entities from entity space to relation space. As illustrated in Fig. 7.6, TransD uses \( h, t, r \) to represent the original vectors, while \( h_p, t_p, r_p \) are used to represent the projection vectors. There are two projection matrices \( M_{rh}, M_{rt} \in \mathbb{R}^{m \times n} \) used to project from entity space to relation space, and the projection matrices are dynamically constructed as follows:

\[
M_{rh} = r_p h_p^\top + I_{m \times n}, \quad M_{rt} = r_p t_p^\top + I_{m \times n},
\]
which means the projection vectors of entity and relation are combined to determine the dynamic projection matrix. The score function is then defined as

$$E(h, r, t) = \|M_{rh}h + r - M_{rt}t\|. \quad (7.12)$$

The projection matrices are initialized with identity matrices, and there are also some normalization constraints as in TransR.

TransD proposes a dynamic method to construct projection matrices with the consideration of diversity in both entities and relations, achieving better performance compared to existing methods in link prediction and triple classification. Moreover, it lowers both computational and spatial complexity compared to TransR.

### 7.2.3.4 TranSparse

The extension methods of TransE stated above focus on the multiple representations for entities in different relations and entity pairs. However, there are still two challenges ignored: (1) The heterogeneity. Relations in knowledge graphs differ in granularity. Some complex relations may link to many entity pairs, while some relatively simple relations not. (2) The unbalance. Some relations may have more links to head entities and fewer links to tail entities, and vice versa. The performance will be further improved if we consider these rather than merely treat all relations equally.

Existing methods like TransR build projection matrices for each relation, while these projection matrices have the same number of parameters, regardless of the variety in the complexity of relations. TranSparse [33] is then proposed to address the issues. The underlying assumption of TranSparse is that complex relations should have more parameters to learn while simple relations have fewer, where the complexity of a relation is judged from the number of triples or entities linked by the relation.
To accomplish this, two models, i.e., TranSparse(share) and TranSparse(separate), are proposed for avoiding overfitting and underfitting.

Inspired by TransR, TranSparse(share) builds a projection matrix $M_r(\theta_r)$ for each relation $r$. This projection matrix is sparse, and the sparse degree $\theta_r$ 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 matrix $M_r$ that $0 \leq \theta_{min} \leq 1$. The sparse degree of relation $r$ is defined as follows:

$$\theta_r = 1 - (1 - \theta_{min})N_r/N_r^*.$$  \hfill (7.13)

Both head and tail entities share the same sparse projection matrix $M_r(\theta_r)$ in translation. The score function is

$$\mathcal{E}(h, r, t) = \|M_r(\theta_r)h + r - M_r(\theta_r)t\|.$$ \hfill (7.14)

Differing from TranSparse(share), TranSparse(separate) builds two different sparse matrices $M_{rh}(\theta_{rh})$ and $M_{rt}(\theta_{rt})$ for head and tail entities. The sparse degree $\theta_{rh}$ (or $\theta_{rt}$) then depends on the number of head (or tail) entities linked by relation $r$. 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

$$\theta_{rh} = 1 - (1 - \theta_{min})N_{rh}/N_{rh}^*, \quad \theta_{rt} = 1 - (1 - \theta_{min})N_{rt}/N_{rt}^*.$$  \hfill (7.15)

The score function of TranSparse(separate) is

$$\mathcal{E}(h, r, t) = \|M_{rh}(\theta_{rh})h + r - M_{rt}(\theta_{rt})t\|.$$ \hfill (7.16)

Through the sparse projection matrix, TranSparse solves the heterogeneity and the unbalance simultaneously.

### 7.2.3.5 PTransE

The extension models of TransE stated above are mainly focused on the challenge of multiple representations of entities in different scenarios. However, those extension models only consider the simple one-step paths (i.e., relation) in translating operation, ignoring the rich global information located in the whole knowledge graphs. Considering the multistep relational path is a potential method to utilize the global information. For instance, if we notice the multistep relational path that $\langle$The forbidden city, locate_in, Beijing$\rangle \rightarrow \langle$Beijing, capital_of, China$\rangle$, we can inference with confidence that the triple $\langle$The forbidden city, locate_in, China$\rangle$ may exist. The relational path provides us with a powerful way to con-
struct better knowledge graph representations and even get a better understanding of 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 relation sequence patterns which do not indicate reasonable relationships. Let us just consider the relational path \(<\text{The forbidden city}, \text{locate_in}, \text{Beijing}> \rightarrow <\text{Beijing}, \text{held}, \text{2008 Summer Olympics}>\), it is hard to describe the relationship between The forbidden city and 2008 Summer Olympics. Second, how to model those meaningful relational paths once we get them since it is difficult to solve this composition semantic problem in relational paths.

PTransE [38] is then proposed to model the multistep relational paths. To select meaningful relational paths, the authors propose a Path-Constraint Resource Allocation (PCRA) algorithm to judge the relation path reliability. Suppose there is information (or resource) in head entity \(h\) which will flow to tail entity \(t\) through some certain relational paths. The basic assumption of PCRA is that: the reliability of path \(\ell\) depends on the resource amount that finally flows from head to tail. Formally, we set \(\ell = (r_1, \ldots, r_l)\) for a certain path between \(h\) and \(t\). The resource travels from \(h\) to \(t\) and 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 resource amount of \(m\) is defined as follows:

\[
R_\ell(m) = \sum_{n \in S_{i-1}(\cdot, m)} \frac{1}{|S_i(n, \cdot)|} R_\ell(n), \tag{7.17}
\]

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

Once we have learned the reliability and select those meaningful relational path candidates, the next challenge is how to model the meaning of those multistep paths. PTransE proposes three types of composition operation, namely, Addition, Multiplication, and recurrent neural networks, to get the representation \(l\) of \(\ell = (r_1, \ldots, r_l)\) through those relations. The score function of the path triple \(\langle h, \ell, t \rangle\) is defined as follows:

\[
E(h, \ell, t) = \|l - (t - h)\| \approx \|l - r\| = E(\ell, r), \tag{7.18}
\]

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

\[
\mathcal{L} = \sum_{(h, r, t) \in S} [\mathcal{L}(h, r, t) + \frac{1}{Z} \sum_{\ell \in \mathcal{P}(h, t)} R(\ell|h, t) \mathcal{L}(\ell, r)], \tag{7.19}
\]
where $\mathcal{L}(h, r, t)$ is the margin-based score function with $\mathcal{E}(h, r, t)$ and $\mathcal{L}(\ell, r)$ is the margin-based score function with $\mathcal{E}(\ell, r)$. The reliability $R(\ell|h, t)$ of $\ell$ in $(h, \ell, t)$ is well considered in the overall loss function.

Besides PTransE, similar ideas such as [21, 22] also consider the multistep relational paths on different tasks such as knowledge completion and question answering successfully. These works demonstrate that there is plentiful information located in multi-step relational paths, which could significantly improve the performance of knowledge graph representation, and further explorations on more sophisticated models for relational paths are still promising.

### 7.2.3.6 TransA

TransA [78] is proposed to solve the following problems in TransE and other extensions: (1) TransE and its extensions only consider the Euclidean distance in their energy functions, which seems to be less flexible. (2) Existing methods regard each dimension in the semantic vector space identically whatever the triple is, which may bring in errors when calculating dissimilarities. To solve these problems, as illustrated in Fig. 7.7, TransA replaces the inflexible Euclidean distance with adaptive Mahalanobis distance, which is more adaptive and flexible. The energy function of TransA is as follows:

$$\mathcal{E}(h, r, t) = (|h + r - t|)^\top W_r(|h + r - t|), \quad (7.20)$$

where $W_r$ is a relation-specific nonnegative symmetric matrix corresponding to the adaptive matrix. Note that the $|h + r - t|$ stands for a nonnegative vector that each dimension is the absolute value of the translating operation. We have

$$(|h + r - t|) \triangleq (|h_1 + r_1 - t_1|, |h_2 + r_2 - t_2|, \ldots |h_n + r_n - t_n|). \quad (7.21)$$
### 7.2.3.7 KG2E

Existing translation-based models usually consider entities and relations as vectors embedded in low-dimensional semantic spaces. However, as explained above, entities and relations in KGs are various with different granularities. Therefore, the margin in the margin-based score function that is used to distinguish positive triples from negative triples should be more flexible due to the diversity, and the uncertainties of entities and relations should be taken into consideration.

To solve this, KG2E [30] is proposed, introducing the multidimensional Gaussian distributions to KG representations. As illustrated in Fig. 7.8, KG2E represents each entity and relation with a Gaussian distribution. Specifically, the mean vector denotes the entity/relation’s central position, and the covariance matrix denotes its uncertainties. To learn the Gaussian distributions for entities and relations, KG2E also follows the score function proposed in TransE. For a triple \( (h, r, t) \), the Gaussian distributions of entity and relation are defined as follows:

\[
\begin{align*}
    h & \sim \mathcal{N}(\mu_h, \Sigma_h), \\
    t & \sim \mathcal{N}(\mu_t, \Sigma_t), \\
    r & \sim \mathcal{N}(\mu_r, \Sigma_r).
\end{align*}
\]

(7.22)

Note that the covariances are diagonal for the consideration of efficiency. KG2E hypothesizes that the head and tail entity are independent with specific relations, then the translation \( h - t \) could be defined as

\[
    h - t = e \sim \mathcal{N}(\mu_h - \mu_t, \Sigma_h + \Sigma_t).
\]

(7.23)

To measure the dissimilarity between \( e \) and \( r \), KG2E proposes two methods considering both asymmetric similarity and symmetric similarity.

The asymmetric similarity is based on the KL divergence between \( e \) and \( r \), which is a straightforward method to measure the similarity between two probability distributions. The energy function is as follows:

\[
    \mathcal{E}(h, r, t) = D_{KL}(e \| r) = \int_{x \in \mathbb{R}^d} \mathcal{N}(x; \mu_r, \Sigma_r) \log \frac{\mathcal{N}(x; \mu_e, \Sigma_e)}{\mathcal{N}(x; \mu_r, \Sigma_r)} dx
    = \frac{1}{2} \left\{ \text{tr}(\Sigma_r^{-1} \Sigma_e) + (\mu_r - \mu_e)^\top \Sigma_r^{-1} (\mu_r - \mu_e) - \log \frac{\det(\Sigma_e)}{\det(\Sigma_r)} - k_e \right\},
\]

(7.24)

where \( \text{tr}(\Sigma) \) indicates the trace of \( \Sigma \), and \( \Sigma^{-1} \) indicates the inverse.

The symmetric similarity is based on the expected likelihood or probability product kernel. KE2G takes the inner product between \( P_e \) and \( P_r \) as the measurement of similarity. The logarithm of energy function is
Fig. 7.8 The architecture of KG2E model [47]

\[
\mathcal{E}(h, r, t) = \int_{x \in \mathbb{R}^{ke}} \mathcal{N}(x; \mu_e, \Sigma_e) \mathcal{N}(x; \mu_r, \Sigma_r) dx \\
= \log \mathcal{N}(0; \mu_e - \mu_r, \Sigma_e + \Sigma_r) \\
= \frac{1}{2} \left\{ (\mu_e - \mu_r)^\top (\Sigma_e + \Sigma_r)^{-1} (\mu_e - \mu_r) + \log \det (\Sigma_e + \Sigma_r) + ke \log(2\pi) \right\}.
\]

\[ (7.25) \]

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

\[
\forall l \in E \cup R, \quad \|\mu_l\|_2 \leq 1, \quad c_{min} I \leq \Sigma_l \leq c_{max} I, \quad c_{min} > 0.
\]

\[ (7.26) \]

Figure 7.8 shows a brief example of representations in KG2E.

7.2.3.8 TransG

We have discussed the problem of TransE in the session of TransR/CTransR that some relations in knowledge graphs such as location_contains or has_part may have multiple sub-meanings. These relations are more likely to be some combinations that could be divided into several more precise relations. To address this issue, CTransR is proposed with a preprocess of clustering for each relation \( r \) depending on the entity pairs \((h, t)\). TransG [79] also focuses on this issue more elegantly by introducing a generative model. As illustrated in Fig. 7.9, it assumes that different semantic component embeddings should follow a Gaussian Mixture Model. The generative process is as follows:

1. For each entity \( e \in E \), TransG sets a standard normal distribution: \( \mu_e \sim \mathcal{N}(0, I) \).
2. For a triple \( (h, r, t) \), TransG uses Chinese Restaurant Process to automatically detect semantic components (i.e., sub-meanings in a relation): \( \pi_{r,n} \sim \text{CRP}(\beta) \).
3. Draw the head embedding to form a standard normal distribution: \( h \sim \mathcal{N}(\mu_h, \sigma_h^2 I) \).
4. Draw the tail embedding to form a standard normal distribution: \( t \sim \mathcal{N}(\mu_t, \sigma_t^2 I) \).
5. Draw the relation embedding for this semantic component: \( \mu_{r,n} = t - h \sim \mathcal{N}(\mu_t - \mu_h, (\sigma_h^2 + \sigma_t^2) I) \).

\( \mu \) is the mean embedding and \( \sigma \) is the variance. Finally, the score function is

\[
E(h, r, t) \propto \sum_{n=1}^{N_r} \pi_{r,n} \mathcal{N}(\mu_t - \mu_h, (\sigma_h^2 + \sigma_t^2) I),
\]

in which \( N_r \) is the number of semantic components of \( r \), and \( \pi_{r,n} \) is the weight of \( i \)th component generated by the Chinese Restaurant Process.

Figure 7.9 shows the advantages of the generative Gaussian Mixture Model.

### 7.2.3.9 ManifoldE

KG2E and TransG introduce Gaussian distributions to knowledge graph representation learning, improving the flexibility and diversity with the various forms of entity and relation representation. However, TransE and its most extensions view the golden triples as almost points in the low-dimensional vector space, following the assumption of translation. This point assumption may lead to two problems: being an ill-posed algebraic system and being over-strict with the geometric form.

ManifoldE [80] is proposed to address this issue, considering the possible position of the golden candidate in vector space as a manifold instead of one point. The overall score function of ManifoldE is defined as follows:

\[
E(h, r, t) = \| M(h, r, t) - D_r^2 \|^2,
\]

in which \( M(h, r, t) \) is the manifold representation of the triple, and \( D_r^2 \) is the distance metric.
in which $D_r^2$ is a relation-specific manifold parameter indicating the bias. Two kinds of manifolds are then proposed in ManifoldE. ManifoldE(Sphere) is a straightforward manifold that supposes $t$ should be located in the sphere which has $h+r$ to be the center and $D_r$ to be the radius. We have

$$M(h, r, t) = \|h + r - t\|_2^2.$$  

(7.29)

The second manifold utilized is the hyperplane for it is much easier for two hyperplanes to intersect. The function of ManifoldE(Hyperplane) is

$$M(h, r, t) = (h + r_h)\top (t + r_t),$$  

(7.30)

in which $r_h$ and $r_t$ represent the two relation embeddings. This indicates that for a triple $\langle h, r, t \rangle$, the tail entity $t$ should locate in the hyperplane whose direction is $h + r_h$ with the bias to be $D_r^2$. Furthermore, ManifoldE(Hyperplane) considers absolute values in $M(h, r, t)$ as $|h + r_h| |t + r_t|$ to double the solution number of possible tails. For both manifolds, the author applies kernel forms on Reproducing Kernel Hilbert Space.

### 7.2.4 Other Models

Translation-based methods such as TransE are simple but effective, whose power has been consistently verified on various tasks like knowledge graph completion and triple classification, achieving state-of-the-art performance. However, there are also some other representation learning methods performing well on knowledge graph representation. In this part, we will take a brief look at these methods as inspiration.

#### 7.2.4.1 Structured Embeddings

Structured Embeddings (SE) [8] is a classical representation learning method for KGs. In SE, each entity is projected to a $d$-dimensional vector space. SE designs two relation-specific matrices $M_{r,1}, M_{r,2} \in \mathbb{R}^{d \times d}$ for each relation $r$, projecting both head and tail entities with these relation-specific matrices when calculating the similarities. The score function of SE is defined as follows:

$$E(h, r, t) = ||M_{r,1}h - M_{r,2}t||_1.$$  

(7.31)

in which both $h$ and $t$ are transformed into a relation-specific vector space with those projection matrices. The assumption of SE is that the projected head and tail embeddings should be as similar as possible according to the loss function. Different from the translation-based methods, SE models entities as embeddings and relations.
as projection matrices. In training, SE considers all triples in the training set and minimizes the overall loss function.

### 7.2.4.2 Semantic Matching Energy

Semantic Matching Energy (SME)\cite{5, 6} proposes a more complicated representation learning method. Differing from SE, SME considers both entities and relations as low-dimensional vectors. For a triple \( \langle h, r, t \rangle \), \( h \) and \( r \) are combined using a projection function \( g \) to get a new embedding \( l_{h,r} \), and the same with \( t \) and \( r \) to get \( l_{t,r} \). Next, a point-wise multiplication function is used on the two combined embeddings \( l_{h,r} \) and \( l_{t,r} \) to get the score of this triple. SME proposes two different projection functions in the second step, among which the linear form is

\[
\mathcal{E}(h, r, t) = (M_1h + M_2r + b_1)\top (M_3t + M_4r + b_2), \tag{7.32}
\]

and the bilinear form is:

\[
\mathcal{E}(h, r, t) = ((M_1h \odot M_2r) + b_1)\top ((M_3t \odot M_4r) + b_2), \tag{7.33}
\]

where \( \odot \) is the element-wise (Hadamard) product. \( M_1, M_2, M_3, M_4 \) are weight matrices in the projection function, and \( b_1 \) and \( b_2 \) are the bias. Bordes et al.\cite{6} is based on SME and improves the bilinear form with three-way tensors instead of matrices.

### 7.2.4.3 Latent Factor Model

Latent Factor Model (LFM) is proposed for modeling large multi-relational datasets. LFM is based on a bilinear structure, which models entities as embeddings and relations as matrices. It could share sparse latent factors among different relations, significantly reducing the model and computational complexity. The score function of LFM is defined as follows:

\[
\mathcal{E}(h, r, t) = h\top M_r t, \tag{7.34}
\]

in which \( M_r \) is the representation of the relation \( r \). Moreover,\cite{92} proposes DISTMULT model, which restricts \( M_r \) to be a diagonal matrix. This enhanced model not only reduces the parameter number of LFM and thus lowers the model’s computational complexity, but also achieves better performance.
7.2.4.4 RESCAL

RESCAL is a knowledge graph representation learning method based on matrix factorization [54, 55]. In RESCAL, to represent all triple facts in knowledge graphs, the authors employ a three-way tensor $\mathbf{X} \in \mathbb{R}^{d \times d \times k}$ in which $d$ is the dimension of entities and $k$ is that of relations. In the three-way tensor $\mathbf{X}$, two modes stand for the head and tail entities while the third mode represents the relations. The entries of $\mathbf{X}$ are based on the existence of the corresponding triple facts. That is, $\mathbf{X}_{ijm} = 1$ if the triple $\langle i\text{th entity, } m\text{th relation, } j\text{th entity} \rangle$ holds in the training set, and otherwise $\mathbf{X}_{ijm} = 0$ if the triple is nonexisting.

To capture the inherent structure of all triples, a tensor factorization model named RESCAL is then proposed. Suppose $\mathbf{X} = \{\mathbf{X}_1, \ldots, \mathbf{X}_k\}$, for each slice $\mathbf{X}_n$, we have the following rank-$r$ factorization:

$$\mathbf{X}_n \approx \mathbf{A} \mathbf{R}_n \mathbf{A}^\top,$$

(7.35)

where $\mathbf{A} \in \mathbb{R}^{d \times r}$ stands for the $r$-dimensional entity representations, and $\mathbf{R}_n \in \mathbb{R}^{r \times r}$ represents the interactions of the $r$ latent components for $n$-th relation. The assumption in this factorization is similar to LFM, while RESCAL also optimizes the nonexisting triples where $\mathbf{X}_{ijm} = 0$ instead of only considering the positive instances.

Following this tensor factorization assumption, the loss function of RESCAL is defined as follows:

$$\mathcal{L} = \frac{1}{2} \left( \sum_n \|\mathbf{X}_n - \mathbf{A} \mathbf{R}_n \mathbf{A}^\top \|_F^2 \right) + \frac{1}{2} \lambda \left( \|\mathbf{A}\|_F^2 + \sum_n \|\mathbf{R}_n\|_F^2 \right),$$

(7.36)

in which the second term is a regularization term and $\lambda$ is a hyperparameter.

7.2.4.5 HOLE

RESCAL works well with multi-relational data but suffers from high computational complexity. To leverage both effectiveness and efficiency, Holographic Embeddings (HOLE) is proposed as an enhanced version of RESCAL [53].

HOLE employs an operation named circular correlation to generate compositional representations, which is similar to those holographic models of associative memory. The circular correlation operation $\star : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$ between two entities $h$ and $t$ is as follows:

$$h \star t_k = \sum_{i=0}^{d-1} h_{i} t_{(k+i)\text{mod } d}.$$

(7.37)

Figure 7.10a also demonstrates a simple instance of this operation. The probability of a triple $\langle h, r, t \rangle$ is then defined as
Considering circular correlation brings in lots of advantages: (1) unlike other operations like multiplication or convolution, circular correlation is noncommutative (i.e., $h \star t \neq t \star h$), which is capable of modeling asymmetric relations in knowledge graphs. (2) Circular correlation has lower computational complexity compared to tensor product in RESCAL. What’s more, the circular correlation could further speed up with the help of Fast Fourier Transform (FFT), which is formalized as follows:

$$h \star t = \mathcal{F}^{-1}(\mathcal{F}(h) \odot \mathcal{F}(t)).$$

(7.39)
$\mathcal{F} (\cdot)$ and $\mathcal{F} (\cdot)^{-1}$ represent the FFT and its inverse, while $\overline{\mathcal{F}} (\cdot)$ denotes the complex conjugate in $\mathbb{C}^d$, and $\odot$ stands for the element-wise (Hadamard) product. Due to FFT, the computational complexity of circular correlation is $O(d \log d)$, which is much lower than that of tensor product.

### 7.2.4.6 Complex Embedding (ComplEx)

ComplEx [70] employs an eigenvalue decomposition model, which makes use of complex valued embeddings. The composition of complex embeddings can handle a large variety of binary relations, among the symmetric and antisymmetric relations. Formally, the log-odd of the probability that the fact $\langle h, r, t \rangle$ is true is

$$f_r(h, t) = \text{Sigmoid}(X_{hrt}), \quad (7.40)$$

where $f_r(h, t)$ is expected to be 1 when $(h, r, t)$ holds, otherwise $-1$. Here, $X_{hrt}$ is calculated as follows:

$$X_{hrt} = \Re(\langle r, h, t \rangle)$$

$$= \langle \Re(r), \Re(h), \Re(t) \rangle + \langle \Re(r), \Im(h), \Im(t) \rangle$$

$$- \langle \Im(r), \Re(h), \Im(t) \rangle - \langle \Im(r), \Im(h), \Re(t) \rangle, \quad (7.41)$$

where $\langle x, y, z \rangle = \sum_i x_i y_i z_i$ denotes the trilinear dot product, $\Re(x)$ and $\Im(x)$ indicate the real part and the imaginary part of the number $x$ respectively. In fact, ComplEx can be viewed as an extension of RESCAL, which assigns complex embedding of the entities and relations.

Besides, [29] has proved that HolE is mathematically equivalent to ComplEx recently.

### 7.2.4.7 Convolutional 2D Embeddings (ConvE)

ConvE [16] uses 2D convolution over embeddings and multiple layers of nonlinear features to model knowledge graphs. It is the first nonlinear model that significantly outperforms previous linear models.

Specifically, ConvE uses convolutional and fully connected layers to model the interactions between input entities and relationships. After that, the obtained features are flattened, transformed through a fully connected layer, and the inner product is taken with all object entity vectors to generate a score for each triple.

For each triple $\langle h, r, t \rangle$, ConvE defines its score function as

$$f_r(h, t) = f(\text{vec}(f([\bar{h}; \bar{r}] \ast \omega))W)t, \quad (7.42)$$
7.2 Knowledge Graph Representation

where \(*\) denotes the convolution operator, and \(\text{vec}(\cdot)\) means compressing a matrix into a vector. \(r \in \mathbb{R}^k\) is a relation parameter depending on \(r\), \(\hat{h}\) and \(\hat{r}\) denote a 2D reshaping of \(h\) and \(r\), respectively: if \(h, r \in \mathbb{R}^k\), then \(\hat{h}, \hat{r} \in \mathbb{R}^{ka \times kb}\), where \(k = ka \times kb\).

ConvE can be seen as an improvement on HolE. Compared with HolE, it learns multiple layers of nonlinear features, and thus theoretically more expressive than HolE.

7.2.4.8 Rotation Embeddings (RotatE)

RotatE [67] defines each relation as a rotation from the head entity to the tail entity in the complex vector space. Thus, it is able to model and infer various relation patterns, including symmetry/antisymmetry, inversion, and composition. Formally, the score function of the fact \(\langle h, r, t \rangle\) of RotatE is defined as

\[
f_r(h, t) = \| h \odot r - t \|, \tag{7.43}\]

where \(\odot\) denotes the element-wise (Hadamard) product, \(h, r, t \in \mathbb{C}^k\) and \(|r| = 1\).

RotatE is simple but achieves quite good performance. Compared with previous work, it is the first model that is capable of modeling and inferring all the three relation patterns above.

7.2.4.9 Neural Tensor Network

Socher et al. [65] propose Neural Tensor Network (NTN) as well as Single Layer Model (SLM), while NTN is an enhanced version of SLM. Inspired by the previous attempts in KRL, SLM represents both entities and relations as low-dimensional vectors, and also designs relation-specific projection matrices to map entities from entity space to relation space. Similar to SE, the score function of SLM is as follows:

\[
E(h, r, t) = r^\top \tanh(M_{r,1}h + M_{r,2}t), \tag{7.44}\]

where \(h, t \in \mathbb{R}^d\) represent head and tail embeddings, \(r \in \mathbb{R}^k\) represents relation embedding, and \(M_{r,1}, M_{r,2} \in \mathbb{R}^{d \times k}\) stand for the relation-specific matrices.

Although SLM has introduced relation embeddings as well as a nonlinear layer into the score function, the model representation capability is still restricted. Neural tensor network is then proposed with tensors being introduced into the SLM framework. Besides the original linear neural network layer that projects entities to the relation space, NTN also adds another tensor-based neural layer which combines head and tail embeddings with a relation-specific tensor, as illustrated in Fig. 7.11. The score function of NTN is then defined as follows:

\[
E(h, r, t) = r^\top \tanh(h^\top \bar{M}_r t + M_{r,1}h + M_{r,2}t + b_r), \tag{7.45}\]
where $\mathbf{M}_r \in \mathbb{R}^{d \times d \times k}$ is a 3-way relation-specific tensor, $\mathbf{b}_r$ is the bias, and $\mathbf{M}_{r,1}, \mathbf{M}_{r,2} \in \mathbb{R}^{d \times k}$ is the relation-specific matrices similar to SLM. Note that SLM is the simplified version of NTN if the tensor and bias are set to zero.

Besides the improvements in score function, NTN also attempts to utilize the latent textual information located in entity names and successfully achieves significant improvements. Differing from previous RL models 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. Moreover, the number of words is far less than that of entities. Therefore, using the average word embeddings of entity names will also lower the computational complexity and alleviate the issue of data sparsity.

NTN utilizes tensor-based neural networks to model triple facts and achieves excellent successes. However, the overcomplicated method will lead to higher computational complexity compared to other methods, and the vast number of parameters will limit the performance on rather sparse and large-scale KGs.

### 7.2.4.10 Neural Association Model (NAM)

NAM [43] adopts multilayer nonlinear activations in the deep neural network to model the conditional probabilities between head and tail entities. NAM studies two model structures Deep Neural Network (DNN) and Relation Modulated Neural Network (RMNN).

NAM-DNN feeds the head and tail entities’ embeddings into an MLP with $L$ fully connected layers, which is formalized as follows:
\[ z^{(l)} = \text{Sigmoid}(M^l z^{(l-1)} + b^{(l)}), \quad l = 1, \ldots, L, \quad (7.46) \]

where \( z^{(0)} = [h; r] \), \( M^l \) and \( b^{(l)} \) is the weight matrix and bias vector for the \( l \)-th fully connected layer, respectively. And finally the score function of NAM-DNN is defined as

\[ f_r(h, t) = \text{Sigmoid}(t^\top z^{(L)}). \quad (7.47) \]

Different from NAM-DNN, NAM-RMNN feeds the relation embedding \( r \) into each layer of the deep neural network as follows:

\[ z^{(l)} = \text{Sigmoid}(M^l z^{(l-1)} + B^l r), \quad l = 1, \ldots, L, \quad (7.48) \]

where \( z^{(0)} = [h; r] \), \( M^l \) and \( B^l \) indicate the weight matrices. The score function of NAM-RMNN is defined as

\[ f_r(h, t) = \text{Sigmoid}(t^\top z^{(L)} + B^{(l+1)} r). \quad (7.49) \]

### 7.3 Multisource Knowledge Graph Representation

We are living in a complicated pluralistic real world, in which we can get information through all senses and learn knowledge not only from structured knowledge graphs but also from plain texts, categories, images, and videos. This cross-modal information is considered as multisource information. Besides the structured knowledge graph which is well utilized in previous KRL methods, we will introduce the other kinds of KRL methods utilizing multisource information:

1. **Plain text** is one of the most common information we deliver, receive, and analyze every day. There are vast amounts of plain texts we possess remaining to be detected, in which the significant knowledge that structured knowledge graphs may not include locates. **Entity description** is a special kind of textual information that describes the corresponding entity within a few sentences or a short paragraph. Usually, entity descriptions are maintained by some knowledge graphs (i.e., Freebase) or could be automatically extracted from huge databases like Wikipedia.

2. **Entity type** is another important structured information for building knowledge representations. To learn new objects within our prior knowledge systems, human beings tend to systemize those objects into existing categories. An entity type is usually represented with hierarchical structures, which consist of different granularities of entity subtypes. It is natural that entities in the real world usually have multiple entity types. Most of the existing famous knowledge graphs own their customized hierarchical structures of entity types.

3. **Images** provide intuitive visual information to describe what the entity looks like, which is confirmed to be the most significant information we receive and process every day. The latent information located in images helps a lot, especially when dealing with concrete entities. For instance, we may find out the potential relationship
between Cherry and Plum (there are both plants belonging to Rosaceae) from their appearances. Images could be downloaded from websites, and there are also substantial image datasets like ImageNet.

Multisource information learning provides a novel method to learn knowledge representations not only from the internal information of structured knowledge graphs but also from the external information of plain texts, hierarchical types, and images. Moreover, the exploration in multisource information learning helps to further understand human cognition with all senses in the real world. The cross-modal representations learned based on knowledge graphs will also provide possible relationships between different kinds of information.

### 7.3.1 Knowledge Graph Representation with Texts

Textual information is one of the most common and widely used information these days. There are large plain texts generated every day on the web and easy to be extracted. Words are compressed symbols of our thoughts and can provide the connections between entities, which are of great significance in KRL.

#### 7.3.1.1 Knowledge Graph and Text Joint Embedding

Wang et al. [76] attempt to utilize textual information by jointly embedding entities, relations, and words into the same low-dimensional continuous vector space. Their joint model contains three parts, namely, the knowledge model, the text model, and the alignment model. More specifically, the knowledge model is learned based on the triple facts in KGs by translation-based models, while the text model is learned based on the concurrences of words in the large corpus by Skip-gram. As for the alignment model, two methods are proposed utilizing Wikipedia anchors and entity names. The main idea of alignment by Wikipedia anchors is replacing the word-word pair \((w, v)\) with the word-entity pair \((w, e_v)\) according to the anchors in Wiki pages, while the main idea of alignment by entity names is replacing the entities in original triple \(\langle h, r, t \rangle\) with the corresponding entity names \(\langle w_h, r, t \rangle, \langle h, r, w_t \rangle,\) and \(\langle w_h, r, w_t \rangle\).

Modeling entities and words into the same vector space are capable of encoding both information in knowledge graphs and that in plain texts, while the performance of this joint model depends on the completeness of Wikipedia anchors and may suffer from the weak interactions merely based on entity names. To address this issue, [101] proposes a new joint embedding based on [76] and improves the alignment model with entity descriptions into consideration, assuming that entities should be similar to all words in their descriptions. These joint models learn knowledge and text joint embeddings, improving evaluation performance in both word and knowledge representations.
7.3.1.2 Description-Embodied Knowledge Graph Representation

Another way of utilizing textual information is directly constructing knowledge representations from entity descriptions instead of merely considering the alignments. Xie et al. [82] proposes Description-embodied Knowledge Graph Representation Learning (DKRL) that provides two kinds of knowledge representations: the first is the structure-based representation $h_S$ and $t_S$, which can directly represent entities widely used in previous methods, and the second is the description-based representation $h_D$ and $t_D$ which derives from entity descriptions. The energy function derives from translation-based framework:

$$E(h, r, t) = \| h_S + r - t_S \| + \| h_S + r - t_D \| + \| h_D + r - t_S \| + \| h_D + r - t_D \|. \tag{7.50}$$

The description-based representation is constructed via CBOW or CNN encoders that encode rich textual information from plain texts into knowledge representations. The architecture of DKRL is shown in Fig. 7.12.

Compared to conventional translation-based methods, the two types of entity representations in DKRL are constructed with both structural information and textual information, and thus could get better performance in knowledge graph completion and type classification. Besides, DKRL could represent an entity even if it is not in the training set, as long as there are a few sentences to describe the entity. As their millions of new entities come up every day, DKRL is capable of handling zero-shot learning.
7.3.2 Knowledge Graph Representation with Types

Entity types, which serve as a kind of category information of entities and are usually arranged with hierarchical structures, could provide structured information to understand entities in KRL better.

7.3.2.1 Type-Constrain Knowledge Graph Representation

Krompaß et al. [36] take type information as type constraints, and improves existing methods like RESCAL and TransE via type constraints. It is intuitive that in a particular relation, the head or tail entities should belong to some specific types. For example, the head entities of the relation write_books should be a human (or more precisely an author), and the tail entities should be a book.

Specifically, in RESCAL, the original factorization $X_r \approx AR, A^\top$ is modified to

$$X'_r \approx A_{[\text{head}r,:]} R_r A_{[\text{tail}r,:]}^\top,$$

(7.51)

in which $\text{head}r, \text{tail}r$ are the set of entities fitting the type constraints of head or tail and $X'_r$ is a sparse adjacency matrix of shape $|\text{head}r| \times |\text{tail}r|$. In the enhanced version, only the entities that fit type constraints will be considered during factorization.

In TransE, type constraints are utilized in negative sampling. The margin-based score functions of translation-based methods need negative instances, which are generated through randomly replacing head or tail entities with another entity in triples. With type constraints, the negative samples are chosen by

$$h' \in E_{[\text{head}r]} \subseteq E, \quad t' \in E_{[\text{tail}r]} \subseteq E,$$

(7.52)

where $E_{[\text{head}r]}$ is the subset of entities following type constraints for head in relation $r$, and $E_{[\text{tail}r]}$ is that for tail.

7.3.2.2 Type-Embodied Knowledge Graph Representation

Considering type information as constraints is simple but effective, while the performance is still limited. Instead of merely viewing type information as type constraints, Xie et al. [83] propose Type-embodied Knowledge Graph Representation Learning (TKRL), utilizing hierarchical-type structures to instruct the construction of projection matrices. Inspired by TransR that every entity should have multiple representations in different scenarios, the energy function of TKRL is defined as follows:

$$\mathcal{E}(h, r, t) = \| M_{rh}h + r - M_{rt}t \|,$$

(7.53)
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 triple. Two hierarchical-type encoders are proposed to learn the projection matrices, regarding all subtypes in the hierarchy as projection matrices, in which Recursive Hierarchy Encoder is based on matrix multiplication, while Weighted Hierarchy Encoder is based on matrix summation:

$$\mathbf{M}_{RHE_c} = \prod_{i=1}^{m} \mathbf{M}_{c^{(i)}} = \mathbf{M}_{c^{(1)}} \mathbf{M}_{c^{(2)}} \cdots \mathbf{M}_{c^{(m)}}, \quad (7.54)$$

$$\mathbf{M}_{WHE_c} = \sum_{i=1}^{m} \beta_i \mathbf{M}_{c^{(i)}} = \beta_1 \mathbf{M}_{c^{(1)}} + \cdots + \beta_m \mathbf{M}_{c^{(m)}}, \quad (7.55)$$

where $\mathbf{M}_{c^{(i)}}$ stands for the projection matrix of the $i$th subtype of the hierarchical type $c$, $\beta_i$ is the corresponding weight of the subtype. Figure 7.13 demonstrates a simple illustration of TKRL. Taking RHE, for instance, given an entity William Shakespeare, it is first projected to a rather general sub-type space like human, and then sequentially projected to a more precise subtype like author or English author. Moreover, TKRL also proposes an enhanced soft-type constraint to alleviate the problems caused by type information incompleteness.
7.3.3 Knowledge Graph Representation with Images

Images could provide intuitive visual information of their corresponding entities’ outlook, which may give significant hints suggesting some latent attributes of entities from certain aspects. For instance, Fig. 7.14 demonstrates some examples of entity images of their corresponding entities Suit of armour and Armet. The left side shows the triple facts that \((\text{Suit of armour, has_a_part, Armet})\), and surprisingly, we can infer this knowledge directly from the images.

7.3.3.1 Image-Embodied Knowledge Graph Representation

Xie et al. [81] propose Image-embodied Knowledge Graph Representation Learning (IKRL) to take visual information into consideration when constructing knowledge representations. Inspired by the multiple entity representations in [82], IKRL also proposes the image-based representation \(h_I\) and \(t_I\) besides the original structure-based representation, and jointly learn both two types of entity representations simultaneously within the translation-based framework.

\[
\mathcal{E}(h, r, t) = \|h_S + r - t_S\| + \|h_S + r - t_I\| + \|h_I + r - t_S\| + \|h_I + r - t_I\|. 
\] (7.56)

More specifically, IKRL first constructs the image representations for all entity images with neural networks, and then project these image representations from image space to entity space via a projection matrix. Since most entities may have multiple images with different qualities, IKRL selects the more informative and discriminative images via an attention-based method. The evaluation results of IKRL not only confirm the significance of visual information in understanding...
entities but also show the possibility of a joint heterogeneous semantic space. Moreover, the authors also find some interesting semantic regularities such as \( w(\text{man}) - w(\text{king}) \approx w(\text{woman}) - w(\text{queen}) \) found in word space, which are shown in Fig. 7.15.

### 7.3.4 Knowledge Graph Representation with Logic Rules

Typical knowledge graphs store knowledge in the form of triple facts with one relation linking two entities. Most existing KRL methods only consider the information within triple facts separately, ignoring the possible interactions and correlations between different triples. Logic rules, which are certain kinds of summaries deriving from human beings’ prior knowledge, could help us with knowledge inference and reasoning. For instance, if we know the triple fact that \( \langle \text{Beijing}, \text{is\_capital\_of}, \text{China} \rangle \), we can easily infer with high confidence that \( \langle \text{Beijing}, \text{located\_in}, \text{China} \rangle \), since we know the logic rule that the relation \( \text{is\_capital\_of} \Rightarrow \text{located\_in} \).

Some works are focusing on introducing logic rules to knowledge acquisition and inference, among which Markov Logic Networks are intuitively utilized to address this challenge [3, 58, 75]. The path-based TransE [38] stated above also implicitly considers the latent logic rules between different relations via relation paths.

#### 7.3.4.1 KALE

KALE is a translation-based KRL method that jointly learns knowledge representations with logic rules [24]. The joint learning consists of two parts, namely, the triple modeling and the rule modeling. For triple modeling, KALE follows the translation assumption with minor alteration in scoring function as follows:
\[
E(h, r, t) = 1 - \frac{1}{3 \sqrt{d}} \|h + r - t\|,
\]

in which \(d\) stands for the dimension of knowledge embeddings. \(E(h, r, t)\) takes value in \([0, 1]\) for the convenience of joint learning.

For the newly added rule modeling, KALE uses the t-norm fuzzy logics proposed in [25] that represent the truth value of a complex formula with the truth values of its constituents. Specially, KALE focuses on two typical types of logic rules. The first is \(\forall h, t : \langle h, r_1, t \rangle \Rightarrow \langle h, r_2, t \rangle\) (e.g., given \(\langle\text{Beijing}, \text{is\_capital\_of}, \text{China} \rangle\), we can infer that \(\langle\text{Beijing}, \text{located\_in}, \text{China} \rangle\)). KALE represents the scoring function of this logic rule \(f_1\) via specific t-norm based logical connectives as follows:

\[
E(f_1) = E(h, r_1, t) E(h, r_2, t) - E(h, r_1, t) + 1. \tag{7.58}
\]

The second is \(\forall h, e, t : \langle h, r_1, e \rangle \land \langle e, r_2, t \rangle \Rightarrow \langle h, r_3, t \rangle\) (e.g., given \(\langle\text{Tsinghua}, \text{located\_in}, \text{Beijing} \rangle\) and \(\langle\text{Beijing}, \text{located\_in}, \text{China} \rangle\), we can infer that \(\langle\text{Tsinghua}, \text{located\_in}, \text{China} \rangle\)). And KALE defines the second scoring function as

\[
E(f_2) = E(h, r_1, e) E(e, r_2, t) E(h, r_3, t) - E(h, r_1, e) E(e, r_2, t) + 1. \tag{7.59}
\]

The joint training contains all positive formulae, including triple facts as well as logic rules. Note that for the consideration of logic rule qualities, KALE ranks all possible logic rules by their truth values with pretrained TransE and manually filters some rules ranked at the top.

### 7.4 Applications

Recent years have witnessed the great thrive in knowledge-driven artificial intelligence, such as QA systems and chatbot. AI agents are expected to accurately and deeply understand user demands, and then appropriately and flexibly give responses and solutions. Such kind of work cannot be done without certain forms of knowledge.

To introduce knowledge to AI agents, researchers first extract knowledge from heterogeneous information like plain texts, images, and structured knowledge bases. These various kinds of heterogeneous information are then fused and stored with certain structures like knowledge graphs. Next, the knowledge is projected to a low-dimensional semantic space following some KRL methods. And finally, these learned knowledge representations are utilized in various knowledge applications like information retrieval and dialogue system. Figure 7.16 demonstrates a brief pipeline of knowledge-driven applications from scratch.

From the illustration, we can observe that knowledge graph representation learning is the critical component in the whole knowledge-driven application’s pipeline. It bridges the gap between knowledge graphs that store knowledge and knowledge
applications that use knowledge. Knowledge representations with distributed methods, compared to those with symbolic methods, are able to solve the data sparsity and modeling the similarities between entities and relations. Moreover, embedding-based methods are convenient to be used with deep learning methods and are naturally fit for the combination with heterogeneous information.

In this section, we will introduce possible applications of knowledge representations mainly from two aspects. First, we will introduce the usage of knowledge representations for knowledge-driven applications, and then we will show the power of knowledge representations for knowledge extraction and construction.

### 7.4.1 Knowledge Graph Completion

Knowledge graph completion aims to build structured knowledge bases by extracting knowledge from heterogeneous sources such as plain texts, existing knowledge bases, and images. Knowledge construction consists of several subtasks like relation extraction and information extraction, making the fundamental step in the whole knowledge-driven framework.

Recently, automatic knowledge construction has attracted considerable attention since it is incredibly time consuming and labor intensive to deal with enormous existing and new information. In the following section, we will introduce some explorations on neural relation extraction, and concentrate on the combination of knowledge representations.

#### 7.4.1.1 Knowledge Representations for Relation Extraction

Relation extraction focuses on predicting the correct relation between two entities given a short plain text containing the two entities. Generally, all relations to predict are predefined, which is different to open information extraction. Entities are usually marked with named entity recognition systems or extracted according to anchor texts, or automatically generated via distance supervision [50].
Conventional methods for relation extraction and classification are mainly based on statistical machine learning, which strongly depends on the qualities of extracted features. Zeng et al. [96] first introduce CNN to relation classification and achieve great improvements. Lin et al. [40] further improves neural relation extraction models with attention-based models over instances.

Han et al. [27, 28] propose a novel joint representation learning framework for knowledge acquisition. The key idea is that the joint model learns knowledge and text representations within a unified semantic space via KG-text alignments. Figure 7.17 shows the brief framework of the KG-text joint model. For the text part, the sentence with two entities Mark Twain and Florida is regarded as the input for a CNN encoder, and the output of CNN is considered to be the latent relation place_of_birth of this sentence. While for the KG part, entity and relation representations are learned via translation-based methods. The learned representations of KG and text parts are aligned during training. This work is the first attempt to encode knowledge representations from existing KGs to knowledge construction tasks and achieves improvements in both knowledge completion and relation extraction.
7.4 Applications

Fig. 7.18 The architecture of KNET model

7.4.2 Knowledge-Guided 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, an athlete, and even a basketball player.” 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 [10]. Moreover, entity typing also benefits massive Natural Language Processing (NLP) tasks such as relation extraction [46], question answering [90], and knowledge base population [9].

Conventional named entity recognition models [69, 73] typically classify entity mentions into a small set of coarse labels (e.g., person, organization, location, and others). Since these entity types are too coarse grained for many NLP tasks, a number of works [15, 41, 94, 95] have been proposed to introduce a much larger set of fine-grained types, which are typically subtypes of those coarse-grained types. Previous fine-grained entity typing methods usually derive features using NLP tools such as POS tagging and parsing, and inevitably suffer from error propagation. Dong et al. [18] make the first attempt to explore deep learning in entity typing. The method only employs word vectors as features, discarding complicated feature engineering. Shimaoka et al. [63] further introduce the attention scheme into neural models for fine-grained entity typing.

Neural models have achieved state-of-the-art performance for fine-grained entity typing. However, these methods face the following nontrivial challenges:

1) **Entity-Context Separation.** Existing methods typically encode context words without utilizing crucial correlations between entity and context. However, it is intuitive that the importance of words in the context for entity typ-
ing is significantly influenced by which entity mentions we concern about. For example, in a sentence In 1975, Gates and Paul Allen co-founded Microsoft, which became the world’s largest PC software company, the word company is much more important for determining the type of Microsoft than for the type of Gates.

(2) **Entity-Knowledge Separation.** Existing methods only consider text information of entity mentions for entity typing. In fact, Knowledge Graphs (KGs) provide rich and effective additional information for determining entity types. For example, in the above sentence In 1975, Gates ... Microsoft ... company, even if we have no type information of Microsoft in KG, entities similar to Microsoft (such as IBM) will also provide supplementary information.

In order to address the issues of entity-context separation and entity-knowledge separation, we propose **Knowledge-guided Attention (KNET)** Neural Entity Typing. As illustrated in Fig. 7.18, KNET mainly consists of two parts. Firstly, KNET builds a neural network, including a Long Short-Term Memory (LSTM) and a fully connected layer, to generate context and named entity representations. Secondly, KNET introduces knowledge attention to emphasize those critical words and improve the quality of context representations. Here we introduce the knowledge attention in detail.

Knowledge graphs provide rich information about entities in the form of triples \( \langle h, r, t \rangle \), where \( h \) and \( t \) are entities and \( r \) is the relation between them. Many KRL works have been devoted to encoding entities and relations into real-valued semantic vector space based on triple information in KGs. KRL provides us with an efficient way to exploit KG information for entity typing.

KNET employs the most widely used KRL method TransE to obtain entity embedding \( e \) for each entity \( e \). During the training scenario, it is known that the entity mention \( m \) indicates the corresponding \( e \) in KGs with embedding \( e \), and hence, KNET can directly compute knowledge attention as follows:

\[
a^K_{i} = f \left( W^K_{KA} \begin{bmatrix} \overrightarrow{h_i} \\ \overleftarrow{h_i} \end{bmatrix} \right),
\]

where \( W^K_{KA} \) is a bilinear parameter matrix, and \( a^K_{i} \) is the attention weight for the \( i \)th word.

**Knowledge Attention in Testing.** The challenge is that, in the testing scenario, we do not know the corresponding entity in the KG of a certain entity mention. A solution is to perform entity linking, but it will introduce linking errors. Besides, in many cases, KGs may not contain the corresponding entities for many entity mentions.

To address this challenge, we build an additional text-based representation for entities in KGs during training. Concretely, for an entity \( e \) and its context sentence \( s \), we encode its left and right context into \( c_l \) and \( c_r \), using an one-directional LSTM, and further learn the text-based representation \( \hat{e} \) as follows:
\[
\hat{e} = \tanh \left( W \begin{bmatrix} m \\ c_l \\ c_r \end{bmatrix} \right),
\]

(7.61)

where \( W \) is the parameter matrix, and \( m \) is the mention representation. Note that, LSTM used here is different from those in context representation in order to prevent interference. In order to bridge text-based and KG-based representations, in the training scenario, we simultaneously learn \( \hat{e} \) by putting an additional component in the objective function:

\[
\mathcal{L}_{KG}(\theta) = - \sum_e \| e - \hat{e} \|_2^2.
\]

(7.62)

In this way, in the testing scenario, we can directly use Eq. 7.61 to obtain the corresponding entity representation and compute knowledge attention using Eq. 7.60.

### 7.4.3 Knowledge-Guided Information Retrieval

The emergence of large-scale knowledge graphs has motivated the development of entity-oriented search, which utilizes knowledge graphs to improve search engines. Recent progresses in entity-oriented search include better text representations with entity annotations [61, 85], richer ranking features [14], entity-based connections between query and documents [45, 84], and soft-match query and documents through knowledge graph relations or embeddings [19, 88]. These approaches bring in entities and semantics from knowledge graphs and have greatly improved the effectiveness of feature-based search systems.

Another frontier of information retrieval is the development of neural ranking models (neural-IR). Deep learning techniques have been used to learn distributed representations of queries and documents that capture their relevance relations (representation-based) [62], or to model the query-document relevancy directly from their word-level interactions (interaction-based) [13, 23, 87]. Neural-IR approaches, especially the interaction-based ones, have greatly improved the ranking accuracy when large-scale training data are available [13].

Entity-oriented search and neural-IR push the boundary of search engines from two different aspects. Entity-oriented search incorporates human knowledge from entities and knowledge graph semantics. It has shown promising results on feature-based ranking systems. On the other hand, neural-IR leverages distributed representations and neural networks to learn more sophisticated ranking models form large-scale training data. Entity-Duet Neural Ranking Model (EDRM), as shown in Fig. 7.19, incorporates entities in interaction-based neural ranking models. EDRM first learns the distributed representations of entities using their semantics from knowledge graphs: descriptions and types. Then it follows a recent state-of-the-art entity-oriented search framework, the word-entity duet [86], and matches documents to queries with both bag-of-words and bag-of-entities. Instead of manual features,
EDRM uses interaction-based neural models [13] to match the query and documents with word-entity duet representations. As a result, EDRM combines entity-oriented search and the interaction-based neural-IR; it brings the knowledge graph semantics to neural-IR and enhances entity-oriented search with neural networks.

### 7.4.3.1 Interaction-Based Ranking Models

Given a query $q$ and a document $d$, interaction-based models first build the word-level translation matrix between $q$ and $d$. The translation matrix describes word-pairs similarities using word correlations, which are captured by word embedding similarities in interaction-based models.

Typically, interaction-based ranking models first map each word $w$ in $q$ and $d$ to an $L$-dimensional embedding $v_w$.

$$v_w = \text{Emb}_w(w). \quad (7.63)$$

It then constructs the interaction matrix $M$ based on query and document embeddings. Each element $M_{ij}$ in the matrix, compares the $i$th word in $q$ and the $j$th word in $d$, e.g., using the cosine similarity of word embeddings:

$$M_{ij} = \cos(v_{wi}, v_{wj}). \quad (7.64)$$

With the translation matrix describing the term level matches between query and documents, the next step is to calculate the final ranking score from the matrix. Many approaches have been developed in interaction-based neural ranking models, but in general, that would include a feature extractor on $M$ and then one or several ranking layers to combine the features to the ranking score.
7.4.3.2 Semantic Entity Representation

EDRM incorporates the semantic information about an entity from the knowledge graphs into its representation. The representation includes three embeddings: entity embedding, description embedding, and type embedding, all in \( L \) dimension and are combined to generate the semantic representation of the entity.

**Entity Embedding** uses an \( L \)-dimensional embedding layer \( \text{Emb}_e \) to get the entity embedding \( e \) for \( e \):

\[
v_e = \text{Emb}_e(e).
\] (7.65)

**Description Embedding** encodes an entity description which contains \( m \) words and explains the entity. EDRM first employs the word embedding layer \( \text{Emb}_v \) to embed the description word \( v \) to \( v \). Then it combines all embeddings in the text to an embedding matrix \( V \). Next, it leverages convolutional filters to slide over the text and compose the \( l \) length \( n \)-gram as \( g_{ej} \):

\[
g_{ej} = \text{ReLU}(W_{\text{CNN}} \cdot V_{j} + b_{\text{CNN}}),
\] (7.66)

where \( W_{\text{CNN}} \) and \( b_{\text{CNN}} \) are two parameters of the convolutional filter.

Then we use max pooling after the convolution layer to generate the description embedding \( v_{e}^{\text{des}} \):

\[
v_{e}^{\text{des}} = \max(g_{e1}, \ldots, g_{ej}, \ldots, g_{em}).
\] (7.67)

**Type Embedding** encodes the categories of entities. Each entity \( e \) has \( n \) kinds of types \( F_e = \{ f_1, \ldots, f_j, \ldots, f_n \} \). EDRM first gets the \( f_j \) embedding \( v_{fj} \) through the type embedding layer \( \text{Emb}_{\text{type}} \):

\[
v_{fj}^{\text{emb}} = \text{Emb}_{\text{type}}(e).
\] (7.68)

Then EDRM utilizes an attention mechanism to combine entity types to the type embedding \( v_{e}^{\text{type}} \):

\[
v_{e}^{\text{type}} = \sum_{j}^{n} \alpha_j v_{fj},
\] (7.69)

where \( \alpha_j \) is the attention score, calculated as:

\[
\alpha_j = \frac{\exp(y_j)}{\sum_{l}^{n} \exp(y_l)},
\] (7.70)

\[
y_j = \left( \sum_{l} W_{\text{bow}} v_{l} \right) \cdot v_{fj},
\] (7.71)
where \( y_j \) is the dot product of the query or document representation and type embedding \( f_j \). We leverage bag-of-words for query or document encoding. \( W_{bow} \) is a parameter matrix.

**Combination.** The three embeddings are combined by a linear layer to generate the semantic representation of the entity:

\[
v_{sem}^e = v_{emb}^e + W_e [v_{des}^e; v_{type}^e]^T + b_e,
\]

in which \( W_e \) is an \( L \times 2L \) matrix and \( b_e \) is an \( L \)-dimensional vector.

### 7.4.3.3 Neural Entity-Duet Framework

Word-entity duet [86] is a recently developed framework in entity-oriented search. It utilizes the duet representation of bag-of-words and bag-of-entities to match question \( q \) and document \( d \) with handcrafted features. This work introduces it to neural-IR.

They first construct bag-of-entities \( q^e \) and \( d^e \) with entity annotation as well as bag-of-words \( q^w \) and \( d^w \) for \( q \) and \( d \). The duet utilizes a four-way interaction: query words to document words (\( q^w-d^w \)), query words to documents entities (\( q^w-d^e \)), query entities to document words (\( q^e-d^w \)), and query entities to document entities (\( q^e-d^e \)).

Instead of features, EDRM uses a translation layer that calculates the similarity between a pair of query-document terms: \( (v_{i,w}^q \text{ or } v_{i,e}^q) \) and \( (v_{j,w}^d \text{ or } v_{j,e}^d) \). It constructs the interaction matrix \( M = \{M_{ww}, M_{we}, M_{ew}, M_{ee}\} \). And \( M_{ww}, M_{we}, M_{ew}, M_{ee} \) denote interactions of \( q^w-d^w \), \( q^w-d^e \), \( q^e-d^w \), \( q^e-d^e \) respectively. And elements in them are the cosine similarities of corresponding terms:

\[
M_{ij}^{ww} = \cos(v_{i,w}^q, v_{j,w}^d);
M_{ij}^{we} = \cos(v_{i,w}^q, v_{j,e}^d);
M_{ij}^{ew} = \cos(v_{i,e}^q, v_{j,w}^d);
M_{ij}^{ee} = \cos(v_{i,e}^q, v_{j,e}^d).
\]

(7.73)

The final ranking feature \( \Phi(M) \) is a concatenation of four cross matches (\( \phi(M) \)):

\[
\Phi(M) = [\phi(M_{ww}); \phi(M_{we}); \phi(M_{ew}); \phi(M_{ee})],
\]

(7.74)

where the \( \phi \) can be any function used in interaction-based neural ranking models.

The entity-duet presents an effective way to crossly match query and document in entity and word spaces. In EDRM, it introduces the knowledge graph semantics representations into neural-IR models.

The duet translation matrices provided by EDRM can be plugged into any standard interaction-based neural ranking models such as K-NRM [87] and Conv-KNRM [13]. With sufficient training data, the whole model is optimized end-to-end with back-propagation. During the process, the integration of the knowledge graph semantics, entity embedding, description embeddings, type embeddings, and matching with entities is learned jointly with the ranking neural network.
7.4 Applications

7.4.4 Knowledge-Guided Language Models

Knowledge is an important external information for language modeling. It is because the statistical co-occurrences cannot instruct the generation of all kinds of knowledge, especially for those named entities with low frequencies. Researchers try to incorporate external knowledge into language models for better performance on generation and representation.

7.4.4.1 NKLM

Language models aim to learn the probability distribution over sequences of words, which is a classical and essential NLP task widely studied. Recently, sequence to sequence neural models (seq2seq) are blooming and widely utilized in sequential generative tasks like machine translation [68] and image caption generation [72]. However, most seq2seq models have significant limitations when modeling and using background knowledge.

To address this problem, Ahn et al. [1] propose a Neural Knowledge Language Model (NKLM) that considers knowledge provided by knowledge graphs when generating natural language sequences with RNN language models. The key idea is that NKLM has two ways to generate a word. The first is the same way as conventional seq2seq models that generate a “vocabulary word” according to the probabilities of softmax, and the second is to generate a “knowledge word” according to the external knowledge graphs.

Specifically, the NKLM model takes LSTM as the framework of generating “vocabulary word”. For external knowledge graph information, NKLM denotes the topic knowledge as $\mathcal{K} = \{a_1, \ldots, a_{|\mathcal{K}|}\}$, in which $a_i$ represents the entities (i.e., named as “topic” in [1]) that appear in the same triple of a certain entity. At each step $t$, NKLM takes both “vocabulary word” $w^v_{t-1}$ and “knowledge word” $w^o_{t-1}$ as well as the fact $a_{t-1}$ predicted at step $t-1$ as the inputs of LSTM. Next, the hidden state of LSTM $h_t$ is combined with the knowledge context $e_k$ to get the fact key $k_t$ via an MLP module. The knowledge context $e_k$ derives from the mean embeddings of all related facts of fact $k$. The fact key $k_t$ is then used to extract the most appropriate fact $a_t$ from the corresponding topic knowledge. And finally, the selected fact $a_t$ is combined with hidden state $h_t$ to predict (1) both “vocabulary word” $w^v_t$ and “knowledge word” $w^o_t$, and (2) which word to generate at this step. The architecture of NKLM is shown in Fig. 7.20.

The NKLM model explores a novel neural model that combines the symbolic knowledge information in external knowledge graphs with seq2seq language models. However, the topic of knowledge is given when generating natural languages, which makes NKLM less practical and scalable for more general free talks. Nevertheless, we still believe that it is promising to encode knowledge into language models with such methods.
Pretrained language models like BERT [17] have a strong ability to represent language information from text. With rich language representation, pretrained models obtain state-of-the-art results on various NLP applications. However, the existing pretrained language models rarely consider incorporating external knowledge to provide 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* and *Chronicles: Volume One* are song and book respectively, it is difficult to recognize the two occupations of Bob Dylan, i.e., songwriter and writer.

To enhance language representation models with external knowledge, Zhang et al. [100] propose an enhanced language representation model with informative entities (ERNIE). Knowledge Graphs (KGs) are important external knowledge resources, and they think informative entities in KGs can be the bridge to enhance language representation with knowledge. ERNIE considers overcoming two main challenges for incorporating external knowledge: *Structured Knowledge Encoding* and *Heterogeneous Information Fusion*.

For extracting and encoding knowledge information, ERNIE firstly recognizes named entity mentions in text and then aligns these mentions to their corresponding entities in KGs. Instead of directly using the graph-based facts in KGs, ERNIE encodes the graph structure of KGs with knowledge embedding algorithms like TransE [7], and then takes the informative entity embeddings as input. Based on the alignments between text and KGs, ERNIE integrates entity representations in the knowledge module into the underlying layers of the semantic module.
Similar to BERT, ERNIE adopts the masked language model and the next sentence prediction as the pretraining objectives. Besides, for the better fusion of textual and knowledge features, ERNIE uses a new pretraining objective (denoising entity auto-encoder) by randomly masking some of the named entity alignments in the input text and training to select appropriate entities from KGs to complete the alignments. Unlike the existing pre-trained language representation models only utilizing local context to predict tokens, these objectives require ERNIE to aggregate both context and knowledge facts for predicting both tokens and entities, and lead to a knowledgeable language representation model.

Figure 7.21 is the overall architecture. The left part shows that ERNIE consists of two encoders (T-Encoder and K-Encoder), where T-Encoder is stacked by several classical transformer layers and K-Encoder is stacked by the new aggregator layers designed for knowledge integration. The right part is the detail of the aggregator layer. In the aggregator layer, the input token embeddings and entity embeddings from the preceding aggregator are fed into two multi-head self-attention, respectively. Then, the aggregator adopts an information fusion layer for the mutual integration of the token and entity sequence and computes the output embedding for each token and entity.

ERNIE explores how to incorporate knowledge information into language representation models. The experimental results demonstrate that ERNIE has more powerful abilities of both denoising distantly supervised data and fine-tuning on limited data than BERT.

**7.4.4.3 KALM**

Pre-trained language models can do many tasks without supervised training data, like reading comprehension, summarization, and translation [60]. However, traditional language models are unable to efficiently model entity names observed in text.
solve this problem, Liu et al. [42] propose a new language model architecture, called Knowledge-Augmented Language Model (KALM), to use the entity types of words for better language modeling.

KALM is a language model with the option to generate words from a set of entities from a knowledge database. An individual word can either come from a general word dictionary as in the traditional language model or be generated as a name of an entity from a knowledge database. The training objectives just supervise the output and ignore the decision of the word type. Entities in the knowledge database are partitioned by type and they use the database to build the types of words. According to the context observed so far, the model decides whether the word is a general term or a named entity in a given type. Thus, KALM learns to predict whether the context observed is indicative of a named entity and what tokens are likely to be entities of a given type.

With the language modeling, KALM learns a named entity recognizer without any explicit supervision by using only plain text and the potential types of words. And, it achieves a comparable performance with the state-of-the-art supervised methods.

### 7.4.5 Other Knowledge-Guided Applications

Knowledge enables AI agents to understand, infer, and address user demands, which is essential in most knowledge-driven applications like information retrieval, question answering, and dialogue system. The behavior of AI agents will be more reasonable and accurate with the favor of knowledge representations. In the following subsections, we will introduce the great improvements made by knowledge representation in question answering.

#### 7.4.5.1 Knowledge-Guided Question Answering

Question answering aims to give correct answers according to users’ questions, which needs the capabilities of both natural language understanding of questions and inference on answer selection. Therefore, combining knowledge with question answering is a straightforward application for knowledge representations. Most conventional question answering systems directly utilize knowledge graphs as certain databases, ignoring the latent relationships between entities and relations. Recently, with the thriving in deep learning, explorations have focused on neural models for understanding questions and even generating answers.

Considering the flexibility and diversity of generated answers in natural languages, Yin et al. [93] propose a neural Generative Question Answering model (GENQA), which explores on generating answers to simple factoid questions in natural languages. Figure 7.22 demonstrates the workflow of GENQA. First, a bidirectional RNN is regarded as the Interpreter to transform question $q$ from natural language to compressed representation $H_q$. Next, Enquirer takes $H_Q$ as the key to rank rel-
evant triples facts of $q$ in knowledge graphs and retrieves possible entities in $r_q$. Finally, Answerer combines $H_q$ and $r_q$ to generate answers in the form of natural languages. Similar to [1], at each step, Answerer first decides whether to generate common words or knowledge words according to a logistic regression model. For common words, Answerer acts in the same way as RNN decoders with $H_q$ selected by attention-based methods. As for knowledge words, Answerer directly generates entities with higher ranks.

There are gradually more efforts focusing on encoding knowledge representations into knowledge-driven tasks like information retrieval and dialogue systems. However, how to flexibly and effectively combine knowledge with AI agents remains to be explored in the future.

### 7.4.5.2 Knowledge-Guided Recommendation System

Due to the rapid growth of web information, recommendation systems have been playing an essential role in the web application. The recommendation system aims to predict the “rating” or “preference” that users may give to items. And since KGs can provide rich information, including both structured and unstructured data, recommendation systems have utilized more and more knowledge from KGs to enrich their contexts.

Cheekula et al. [11] explore to utilize the hierarchical knowledge from the DBpedia category structure in the recommendation system and employs the spreading activation algorithm to identify entities of interest to the user. Besides, Passant [56] measures the semantic relatedness of the artist entity in a KG to build music recommendation systems. However, most of these systems mainly investigate the problem by leveraging the structure of KGs. Recently, with the development of representation
learning, [98] proposes to jointly learn the latent representations in a collaborative filtering recommendation system as well as entities’ representations in KGs.

Except the tasks stated above, there are gradually more efforts focusing on encoding knowledge graph representations into other tasks such as dialogue system [37, 103], entity disambiguation [20, 31], knowledge graph alignment [12, 102], dependency parsing [35], etc. Moreover, the idea of KRL has also motivated the research on visual relation extraction [2, 99] and social relation extraction [71].

7.5 Summary

In this chapter, we first introduce the concept of the knowledge graph. Knowledge graph contains both entities and the relationships among them in the form of triple facts, providing an effective way of human beings learning and understanding the real world. Next, we introduce the motivations of knowledge graph representation, which is considered as a useful and convenient method for a large amount of data and is widely explored and utilized in multiple knowledge-based tasks and significantly improves the performance. And we describe existing approaches for knowledge graph representation. Further, we discuss several advanced approaches that aim to deal with the current challenges of knowledge graph representation. We also review the real-world applications of knowledge graph representation such as language modeling, question answering, information retrieval, and recommendation systems.

For further understanding of knowledge graph representation, you can find more related papers in this paper list https://github.com/thunlp/KRLPapers. There are also some recommended surveys and books including:

- Bengio et al. Representation learning: A review and new perspectives [4].
- Liu et al. Knowledge representation learning: A review [47].
- Nickel et al. A review of relational machine learning for knowledge graphs [52].
- Wang et al. Knowledge graph embedding: A survey of approaches and applications [74].
- Ji et al. A survey on knowledge graphs: representation, acquisition and applications [34].

In the future, for better knowledge graph representation, there are some directions requiring further efforts:

1. **Utilizing More Knowledge.** Current KRL approaches focus on representing triple-based knowledge from world knowledge graphs such as Freebase, Wikidata, etc. In fact, there are various kinds of knowledge in the real world such as factual knowledge, event knowledge, commonsense knowledge, etc. What’s more, the knowledge is stored with different formats, such as attributions, quantifier, text, and so on. The researchers have formed a consensus that utilizing more knowledge is a potential way toward more interpretable and intelligent NLP. Some existing works [44, 82] have made some preliminary attempts of utilizing more knowledge
in KRL. Beyond these works, is it possible to represent different knowledge in a unified semantic space, which can be easily applied in downstream NLP tasks?

(2) **Performing Deep Fusion** of knowledge and language. There is no doubt that the joint learning of knowledge and language information can further benefit downstream NLP tasks. Existing works [76, 89, 97] have preliminarily verified the effectiveness of joint learning. Recently, ERINE [100] and KnowBERT [57] further provide us a novel perspective to fuse knowledge and language in pretraining. Soares et al. [64] learn the relational similarity in text with the guidance of KGs, which is also a pioneer of knowledge fusion. Besides designing novel pretraining objectives, we could also design novel model architectures for downstream tasks, which are more suitable to utilize KRL, such as memory-based models [48, 91] and graph network-based models [66]. Nevertheless, it still remains an unsolved problem for effectively performing the deep fusion of knowledge and language.

(3) **Orienting Heterogeneous Modalities**. With the fast development of the World Wide Web, the data size of audios, images, and videos on the Web have become larger and larger, which are also important resources for KRL besides texts. Some pioneer works [51, 81] explore to learn knowledge representations on a multi-modal knowledge graph, but are still preliminary attempts. Intuitively, audio and visual knowledge can provide complementary information, which benefits related NLP tasks. To the best of our knowledge, there still lacks research on applying multi-modal KRL in downstream tasks. How to efficiently and effectively integrate multi-modal knowledge is becoming a critical and challenging problem for KRL.

(4) **Exploring Knowledge Reasoning**. Most of the existing KRL methods represent knowledge information in low-dimensional semantic space, which is feasible for the computation of complex knowledge graphs in neural-based NLP models. Although benefiting from the usability of low-dimensional embeddings, KRL cannot perform explainable reasoning such as symbolic rules, which is of great importance for downstream NLP tasks. Recently, there has been increasing interest in the combination of embedding methods and symbolic reasoning methods [26, 59], aiming at taking both advantages of them. Beyond these works, there remain lots of unsolved problems for developing better knowledge reasoning ability for KRL.

**References**

1. Sungjin Ahn, Heeyoul Choi, Tanel Pärnamaa, and Yoshua Bengio. A neural knowledge language model. arXiv preprint arXiv:1608.00318, 2016.
2. Stephan Baier, Yunpu Ma, and Volker Tresp. Improving visual relationship detection using semantic modeling of scene descriptions. In Proceedings of ISWC, 2017.
3. Islam Beltagy and Raymond J Mooney. Efficient markov logic inference for natural language semantics. In Proceedings of AAAI Workshop, 2014.
4. Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. TPAMI, 35(8):1798–1828, 2013.
5. 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.
6. Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94(2):233–259, 2014.

7. Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Proceedings of NeurIPS*, 2013.

8. Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. Learning structured embeddings of knowledge bases. In *Proceedings of AAAI*, 2011.

9. 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.

10. Mohamed Chabchoub, Michel Gagnon, and Amal Zouaq. Collective disambiguation and semantic annotation for entity linking and typing. In *Proceedings of SWEC*, 2016.

11. Siva Kumar Cheekula, Pavan Kapanipathi, Derek Doran, Prateek Jain, and Amit P Sheth. Entity recommendations using hierarchical knowledge bases. 2015.

12. Muhao Chen, Yingtao Tian, Mohan Yang, and Zaniolo Carlo. Multilingual knowledge graph embeddings for cross-lingual knowledge alignment. In *Proceedings of IJCAI*, 2017.

13. Zhuyun Dai, Chenyang Xiong, Jamie Callan, and Zhiyuan Liu. Convolutional neural networks for soft-matching n-grams in ad-hoc search. In *Proceedings of WSDM*, 2018.

14. Jeffrey Dalton, Laura Dietz, and James Allan. Entity query feature expansion using knowledge base links. In *Proceedings of SIGIR*, 2014.

15. Luciano Del Corro, Abdalghani Abujabal, Rainer Gemulla, and Gerhard Weikum. Finet: Context-aware fine-grained named entity typing. In *Proceedings of EMNLP*, 2015.

16. Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *Proceedings of AAAI*, 2018.

17. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL*, 2019.

18. 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.

19. Faezeh Ensan and Ebrahim Bagheri. Document retrieval model through semantic linking. In *Proceedings of WSDM*, 2017.

20. Wei Fang, Jianwen Zhang, Dilin Wang, Zheng Chen, and Ming Li. Entity disambiguation by knowledge and text jointly embedding. In *Proceedings of CoNLL*, 2016.

21. Alberto García-Durán, Antoine Bordes, and Nicolas Usunier. Composing relationships with translations. In *Proceedings of EMNLP*, 2015.

22. Kelvin Gu, John Miller, and Percy Liang. Traversing knowledge graphs in vector space. In *Proceedings of EMNLP*, 2015.

23. Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. Semantic matching by non-linear word transportation for information retrieval. In *Proceedings of CIKM*, 2016.

24. Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. Jointly embedding knowledge graphs and logical rules. In *Proceedings of EMNLP*, 2016.

25. Petr Hájek. *Metamathematics of fuzzy logic*, volume 4. Springer Science & Business Media, 1998.

26. Will Hamilton, Payal Bajaj, Marinka Zitnik, Dan Jurafsky, and Jure Leskovec. Embedding logical queries on knowledge graphs. In *Proceedings of NIPS*, 2018.

27. Xu Han, Zhiyuan Liu, and Maosong Sun. Joint representation learning of text and knowledge for knowledge graph completion. arXiv preprint arXiv:1611.04125, 2016.

28. Xu Han, Zhiyuan Liu, and Maosong Sun. Neural knowledge acquisition via mutual attention between knowledge graph and text. In *Proceedings of AAAI*, pages 4832–4839, 2018.

29. Katsuhiko Hayashi and Masashi Shimbo. On the equivalence of holographic and complex embeddings for link prediction. In *Proceedings of ACL*, 2017.

30. Shizhu He, Kang Liu, Guoliang Ji, and Jun Zhao. Learning to represent knowledge graphs with gaussian embedding. In *Proceedings of CIKM*, 2015.
31. Hongzhao Huang, Larry Heck, and Heng Ji. Leveraging deep neural networks and knowledge graphs for entity disambiguation. arXiv preprint arXiv:1504.07678, 2015.
32. Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. Knowledge graph embedding via dynamic mapping matrix. In Proceedings of ACL, 2015.
33. Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. Knowledge graph completion with adaptive sparse transfer matrix. In Proceedings of AAAI, 2016.
34. Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S Yu. A survey on knowledge graphs: Representation, acquisition and applications. arXiv preprint arXiv:2002.00388, 2020.
35. A-Yeong Kim, Hyun-Je Song, Seong-Bae Park, and Sang-Jo Lee. A re-ranking model for dependency parsing with knowledge graph embeddings. In Proceedings of IALP, 2015.
36. Denis Krompaß, Stephan Baier, and Volker Tresp. Type-constrained representation learning in knowledge graphs. In Proceedings of ISWC, 2015.
37. Phong Le, Marc Dymetman, and Jean-Michel Renders. Lstm-based mixture-of-experts for knowledge-aware dialogues. In Proceedings of ACL Workshop, 2016.
38. 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.
39. 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.
40. Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. Neural relation extraction with selective attention over instances. In Proceedings of ACL, 2016.
41. Xiao Ling and Daniel S Weld. Fine-grained entity recognition. In Proceedings of AAAI, 2012.
42. Angli Liu, Jingfei Du, and Veselin Stoyanov. Knowledge-enhanced language model and its application to unsupervised named-entity recognition. In Proceedings of NAACL, 2019.
43. 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.
44. Quan Liu, Hui Jiang, Zhen-Hua Ling, Xiaodan Zhu, Si Wei, and Yu Hu. Commonsense knowledge enhanced embeddings for solving pronoun disambiguation problems in winograd schema challenge. arXiv preprint arXiv:1611.04146, 2016.
45. Xitong Liu and Hui Fang. Latent entity space: A novel retrieval approach for entity-bearing queries. Information Retrieval Journal, 18(6):473–503, 2015.
46. 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.
47. Zhiyuan Liu, Maosong Sun, Yankai Lin, and Ruobing Xie. Knowledge representation learning: a review. JCRD, 53(2):247–261, 2016.
48. Todor Mihaylov and Anette Frank. Knowledgeable reader: Enhancing cloze-style reading comprehension with external commonsense knowledge. In Proceedings of ACL, 2018.
49. Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Proceedings of ICLR, 2013.
50. Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. Distant supervision for relation extraction without labeled data. In Proceedings of ACL-IJCNLP, 2009.
51. Hatem Mousselly-Sergieh, Teresa Botschen, Iryna Gurevych, and Stefan Roth. A multimodal translation-based approach for knowledge graph representation learning. In Proceedings of JCLCS, 2018.
52. Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. 2015.
53. Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. Holographic embeddings of knowledge graphs. In Proceedings of AAAI, 2016.
54. Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In Proceedings of ICML, 2011.
55. Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. Factorizing yago: scalable machine learning for linked data. In Proceedings of WWW, 2012.
56. Alexandre Passant. dbrec–music recommendations using dbpedia. In *Proceedings of ISWC*, 2010.
57. Matthew E. Peters, Mark Neumann, Robert L Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. Knowledge enhanced contextual word representations. In *Proceedings of EMNLP-IJCNLP*, 2019.
58. Jay Pujara, Hui Miao, Lise Getoor, and William W Cohen. Knowledge graph identification. In *Proceedings of ISWC*, 2013.
59. Meng Qu and Jian Tang. Probabilistic logic neural networks for reasoning. In *Proceedings of NIPS*, 2019.
60. Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8), 2019.
61. Hadas Raviv, Oren Kurland, and David Carmel. Document retrieval using entity-based language models. In *Proceedings of SIGIR*, 2016.
62. Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of CIKM*, 2014.
63. 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.
64. 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.
65. Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. In *Proceedings of NeurIPS*, 2013.
66. Haitian Sun, Bhuvan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William Cohen. Open domain question answering using early fusion of knowledge bases and text. In *Proceedings of EMNLP*, 2018.
67. 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.
68. Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Proceedings of NeurIPS*, 2014.
69. Erik F Tjong Kim Sang and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of HLT-NAACL*, 2003.
70. Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *Proceedings of ICML*, 2016.
71. Cunchao Tu, Zhengyan Zhang, Zhiyuan Liu, and Maosong Sun. Transnet: translation-based network representation learning for social relation extraction. In *Proceedings of IJCAI*, 2017.
72. Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Proceedings of CVPR*, 2015.
73. Nina Wacholder, Yael Ravin, and Misook Choi. Disambiguation of proper names in text. In *Proceedings of ANLP*, 1997.
74. Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *TKDE*, 29(12):2724–2743, 2017.
75. Quan Wang, Bin Wang, and Li Guo. Knowledge base completion using embeddings and rules. In *Proceedings of IJCAI*, 2015.
76. Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph and text jointly embedding. In *Proceedings of EMNLP*, pages 1591–1601, 2014.
77. Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of AAAI*, 2014.
78. Han Xiao, Minlie Huang, Yu Hao, and Xiaoyan Zhu. Transa: An adaptive approach for knowledge graph embedding. arXiv preprint arXiv:1509.05490, 2015.
79. Han Xiao, Minlie Huang, Yu Hao, and Xiaoyan Zhu. Transg: A generative mixture model for knowledge graph embedding. arXiv preprint arXiv:1509.05488, 2015.
80. 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.
81. Ruobing Xie, Zhiyuan Liu, Tat-seng Chua, Huanbo Luan, and Maosong Sun. Image-embodied knowledge representation learning. In *Proceedings of IJCAI*, 2016.
82. Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. Representation learning of knowledge graphs with entity descriptions. In *Proceedings of AAAI*, 2016.
83. Ruobing Xie, Zhiyuan Liu, and Maosong Sun. Representation learning of knowledge graphs with hierarchical types. In *Proceedings of IJCAI*, 2016.
84. Chenyan Xiong and Jamie Callan. EsdRank: Connecting query and documents through external semi-structured data. In *Proceedings of CIKM*, 2015.
85. Chenyan Xiong, Jamie Callan, and Tie-Yan Liu. Bag-of-entities representation for ranking. In *Proceedings of AAAI*, 2016.
86. Chenyan Xiong, Jamie Callan, and Tie-Yan Liu. Word-entity duet representations for document ranking. In *Proceedings of SIGIR*, 2017.
87. 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.
88. Chenyan Xiong, Russell Power, and Jamie Callan. Explicit semantic ranking for academic search via knowledge graph embedding. In *Proceedings of WWW*, 2017.
89. Jiacheng Xu, Xipeng Qiu, Kan Chen, and Xuanjing Huang. Knowledge graph representation with jointly structural and textual encoding. In *Proceedings of IJCAI*, 2017.
90. Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, and Gerhard Weikum. Robust question answering over the web of linked data. In *Proceedings of CIKM*, 2013.
91. Bishan Yang and Tom Mitchell. Leveraging knowledge bases in LSTMs for improving machine reading. In *Proceedings of ACL*, 2017.
92. 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.
93. Jun Yin, Xin Jiang, Zhengdong Lu, Lifeng Shang, Hang Li, and Xiaoming Li. Neural generative question answering. In *Proceedings of IJCAI*, 2016.
94. Dani Yogatama, Daniel Gillick, and Nevena Lazic. Embedding methods for fine grained entity type classification. In *Proceedings of ACL*, 2015.
95. Mohamed Amir Yosef, Sandro Bauer, Johannes Hoffart, Marc Spaniol, and Gerhard Weikum. HYENA: Hierarchical type classification for entity names. In *Proceedings of COLING*, 2012.
96. Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. Relation classification via convolutional deep neural network. In *Proceedings of COLING*, 2014.
97. Dongxu Zhang, Bin Yuan, Dong Wang, and Rong Liu. Joint semantic relevance learning with text data and graph knowledge. In *Proceedings of ACL-IJCNLP*, 2015.
98. Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. Collaborative knowledge base embedding for recommender systems. In *Proceedings of SIGKDD*, 2016.
99. Hanwang Zhang, Zawlin Kyaw, Shih-Pu Chang, and Tat-Seng Chua. Visual translation embedding network for visual relation detection. In *Proceedings of CVPR*, 2017.
100. 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.
101. Huaping Zhong, Jianwen Zhang, Zhen Wang, Hai Wan, and Zheng Chen. Aligning knowledge and text embeddings by entity descriptions. In *Proceedings of EMNLP*, 2015.
102. Hao Zhu, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. Iterative entity alignment via joint knowledge embeddings. In *Proceedings of IJCAI*, 2017.
103. Wenya Zhu, Kaixiang Mo, Yu Zhang, Zhangbin Zhu, Xuezheng Peng, and Qiang Yang. Flexible end-to-end dialogue system for knowledge grounded conversation. arXiv preprint arXiv:1709.04264, 2017.
Open Access  This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

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