Chapter 5
Deep Learning in Knowledge Graph

Zhiyuan Liu and Xianpei Han

Abstract Knowledge Graph (KG) is a fundamental resource for human-like commonsense reasoning and natural language understanding, which contains rich knowledge about the world's entities, entities' attributes, and semantic relations between different entities. Recent years have witnessed the remarkable success of deep learning techniques in KG. In this chapter, we introduce three broad categories of deep learning-based KG techniques: (1) *knowledge representation learning* techniques which embed entities and relations in a KG into a dense, low-dimensional, and real-valued semantic space; (2) *neural relation extraction* techniques which extract facts/relations from text, which can then be used to construct/complete KG; (3) *deep learning-based entity linking* techniques which bridge Knowledge Graph with textual data, which can facilitate many different tasks.

5.1 Introduction

With the thriving development of Internet in twenty-first century, the amount of web information shows an explosive trend, during which people find it is getting harder and less efficient to extract valuable information, or more precisely, knowledge, from the huge noisy plaintexts. And then, people start to realize that the world is made up of entities instead of strings, just as Dr. Singhal said, "things, not strings". As a result, the concept of Knowledge Graph comes into the public view.

Knowledge Graph (KG), also known as Knowledge Base, is a significant dataset organizing human knowledge about the world in a structured form, where the knowledge is represented as concrete entities and the multi-relational abstract concepts among them. There are mainly two methods when it comes to the construction of Knowledge Graph. One is using the existing semantics web datasets in Resource

Z. Liu
Tsinghua University, Beijing, China
e-mail: liuzy@tsinghua.edu.cn

X. Han (✉)
Institute of Software, Chinese Academy of Sciences, Beijing, China
e-mail: xianpei@iscas.ac.cn

L. Deng and Y. Liu (eds.), *Deep Learning in Natural Language Processing*, https://doi.org/10.1007/978-981-10-5209-5_5

Description Framework (RDF) with the help of manually annotation. The other is using machine learning or deep learning method to automatically extract knowledge from enormous plaintexts in Internet.

Due to such well-structured united knowledge representation, KG can provide effective structured information about the complicated real world. Hence, it starts to play an important role in many applications of artificial intelligence, especially the field of natural language processing and information retrieval such as web search, question answering, speech recognition, etc., in recent years, attracting wide attention from both academia and industry.

In this chapter, we will first introduce the basic concepts and typical Knowledge Graphs in Sect. 5.1, and then, we introduce recent advances of knowledge representation learning in Sect. 5.2, relation extraction in Sect. 5.3, and entity linking in Sect. 5.4. Finally, we give a brief conclusion in Sect. 5.5.

5.1.1 Basic Concepts

A typical KG is usually composed of two elements, entities (i.e., concrete entities and abstract concepts in real world) and relations between entities. Thus, it arranges all kinds of knowledge into large quantities of triple facts in the form of (e_1, `relation`, e_2) where e_1 indicates the head entity and e_2 indicates the tail entity. For instance, we know that *Donald Trump* is the president of *United States*. This knowledge could be represented as (*Donald Trump*, `president_of`, *United States*). Furthermore, it should be noted that in real world, the same head entity and relation may have several different tail entities. For example, *Kak*á was a soccer player in *Real Madrid* and *A.C. Milan* football club. We can get such two triples from this common knowledge: (*Kak*á, `player_of_team`, *Real Madrid FC*), (*Kak*á, `player_of_team`, *A.C. Milan*). Reversely this situation could also happen when tail entity and relation are fixed. It is also possible when head and tail entity are both multiple (e.g., the relation `author_of_paper`). From this aspect, we can see that KG has great flexibility as well as the ability to represent knowledge. Through all these triples, knowledge is thus represented as a huge directed graph, in which entities are considered as nodes and relations as edges.

5.1.2 Typical Knowledge Graphs

The current Knowledge Graphs can be divided into two categories from the aspect of capacity and knowledge domain. The graphs in the first category contain great quantities of triples and well-known common relation, such as Freebase. The graphs in the second category are comparatively smaller but focus on specific knowledge domain and usually fine-grained.

There are several Knowledge Graphs widely used in applications and having great influence. In the following sections, we will introduce some well-known Knowledge Graphs.

5.1.2.1 Freebase

Freebase is one of the most popular Knowledge Graphs in the world. It is a large collaborative database consisting of data composed mainly of its community members. It is an online collection of structured data harvested from many sources, including Wikipedia, Fashion Model Directory, NNDB, MusicBrainz, and other individual, user-submitted wiki contributions. It also announced an open API, RDF endpoint and a dataset dump for its users for both commercial and noncommercial use.

Freebase was developed by the American software company Metaweb and ran publicly since March 2007. In July 2010, Metaweb was acquired by Google, and Google Knowledge Graph was powered in part by Freebase. In December 2014, the Freebase team officially announced that the Freebase website would be shut down together with its API by June 2015. Up to March 24, 2016, Freebase has 58,726,427 topics and 3,197,653,841 facts.

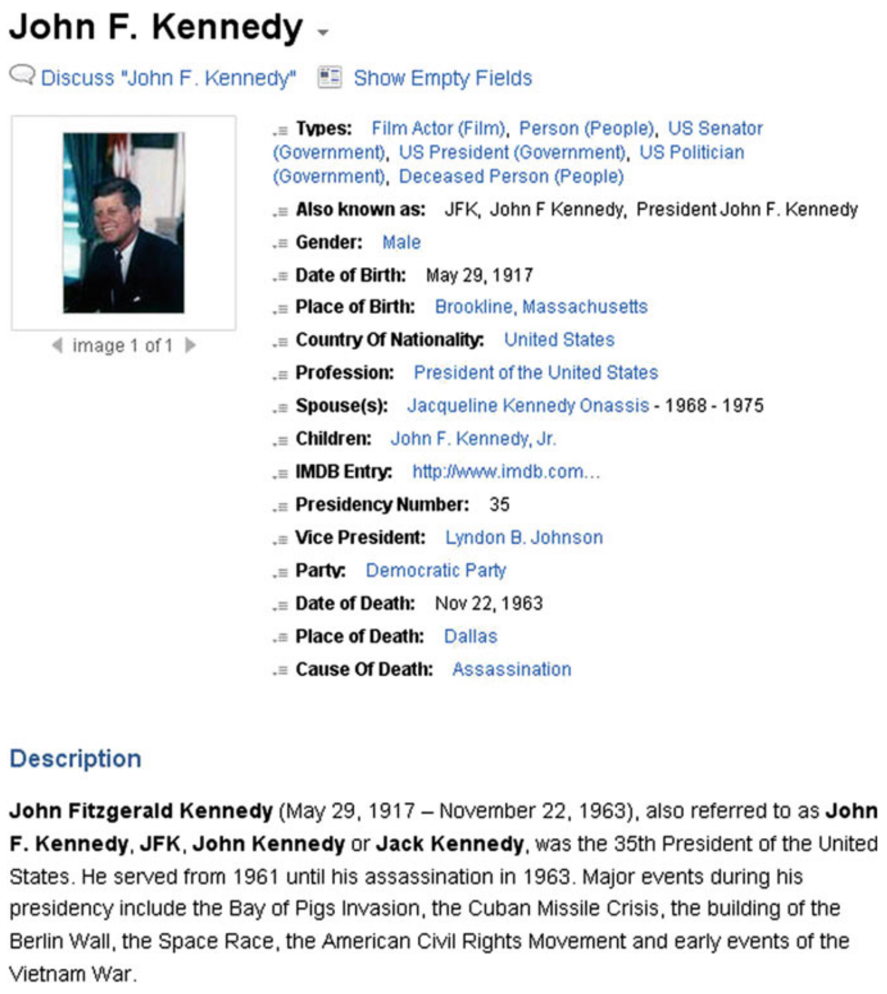
For instance, Fig. 5.1 is the example page of former American president John F. Kennedy in Freebase. It is easy to notice that the information such as date of birth, gender, and career are listed in structured form just like a resume.

5.1.2.2 DBpedia

DBpedia ("DB" stands for "dataset") is a crowdsourced community effort to extract structured information from Wikipedia and make this information available on the web. DBpedia allows users to ask sophisticated queries against Wikipedia, and to link the different datasets on the web to Wikipedia resources, which will make it easier for the huge amount of information in Wikipedia to be used in some new interesting ways. The project was started by people at the Free University of Berlin and Leipzig University, in collaboration with OpenLink Software, and the first publicly available dataset was published in 2007. The whole DBpedia dataset 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, 123,000 music albums, 87,000 films, 19,000 video games, 241,000 organizations, 251,000 species and 6,000 diseases. The dataset also features labels and abstracts for these entities in up to 125 different languages. What is more, due to the reason that DBpedia is linked to Wikipedia's infobox, it can make dynamic updates as the information changes.

5.1.2.3 Wikidata

Wikidata is a collaboratively edited Knowledge Base operated by the Wikimedia Foundation. It is intended to provide a common source of data which can be used by



John F. Kennedy

Discuss "John F. Kennedy" Show Empty Fields

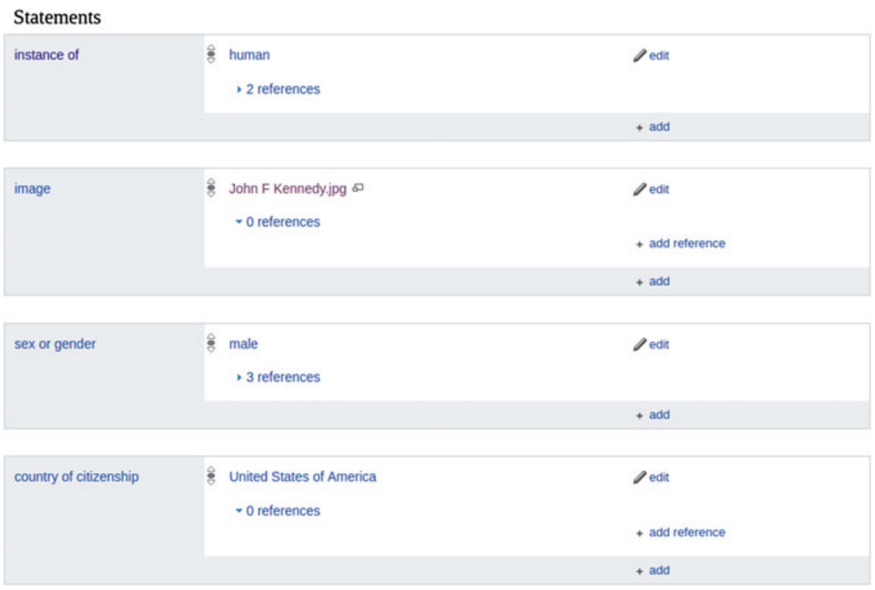
Types:	Film Actor (Film), Person (People), US Senator (Government), US President (Government), US Politician (Government), Deceased Person (People)
Also known as:	JFK, John F Kennedy, President John F. Kennedy
Gender:	Male
Date of Birth:	May 29, 1917
Place of Birth:	Brookline, Massachusetts
Country Of Nationality:	United States
Profession:	President of the United States
Spouse(s):	Jacqueline Kennedy Onassis - 1968 - 1975
Children:	John F. Kennedy, Jr.
IMDB Entry:	http://www.imdb.com...
Presidency Number:	35
Vice President:	Lyndon B. Johnson
Party:	Democratic Party
Date of Death:	Nov 22, 1963
Place of Death:	Dallas
Cause Of Death:	Assassination

Description

John Fitzgerald Kennedy (May 29, 1917 – November 22, 1963), also referred to as **John F. Kennedy**, **JFK**, **John Kennedy** or **Jack Kennedy**, was the 35th President of the United States. He served from 1961 until his assassination in 1963. Major events during his presidency include the Bay of Pigs Invasion, the Cuban Missile Crisis, the building of the Berlin Wall, the Space Race, the American Civil Rights Movement and early events of the Vietnam War.

Fig. 5.1 The Freebase page of John F. Kennedy

Wikimedia projects such as Wikipedia, and by anyone else. The creation of the project was funded by donations from the Allen Institute for Artificial Intelligence, the Gordon and Betty Moore Foundation, and Google, Inc., totaling *euro* 1.3 million. As for the inside detailed structure, Wikidata is a document-oriented database, focused on items. Each item represents a topic (or an administrative page used to maintain Wikipedia) and is identified by a unique number. Information is added to items by creating statements. Statements take the form of key-value pairs, with each statement consisting of a property (the key) and a value linked to the property. Up to May 2017, the Knowledge Base contains 25,887,362 data items that anyone can edit.


Statements

instance of	human 2 references	edit add
image	John F Kennedy.jpg 0 references	edit add reference add
sex or gender	male 3 references	edit add
country of citizenship	United States of America 0 references	edit add reference add

Fig. 5.2 The Wikidata page of John F. Kennedy

For instance, Fig. 5.2 is the example page of John F. Kennedy in Wikidata. It could be noticed that each relation is also attached to references which anyone can add or edit.

5.1.2.4 YAGO

YAGO, which stands for Yet Another Great Ontology, is a huge high-quality Knowledge Base developed by Max Planck Institute for Informatics and the Telecom Paris-Tech University. The knowledge inside is derived from Wikipedia, WordNet, and GeoNames. Currently, it has knowledge of more than 10 million entities (like persons, organizations, cities, etc.) and contains more than 120 million facts about these entities. The highlight spots in YAGO could be concluded as follows: First, the accuracy of YAGO has been manually evaluated, proving a confirmed accuracy of 95% and every relation is annotated with its confidence value. Second, YAGO combines the clean taxonomy of WordNet with the richness of the Wikipedia category system, assigning the entities to more than 350,000 classes. Third, YAGO is an ontology that is anchored in both time and space which means it attaches a temporal dimension and a spacial dimension to many of its facts and entities.

5.1.2.5 HowNet

HowNet (Dong and Dong 2003) is an online commonsense Knowledge Base unveiling inter-conceptual relations and inter-attribute relations of concepts as connoting in lexicons of the Chinese and their English equivalents. The main philosophy behind HowNet is its understanding and interpretation of the objective world. HowNet states that all matters (physical and metaphysical) are in constant motion and are ever changing in a given time and space. Things evolve from one state to another as recorded in the corresponding change in their attributes. Take the instance "human", for example, it is described by the following state of living: birth, aging, sickness, and death. As a person grows, his age (the attribute-value) also adds up. At the meantime, his hair color (an attribute) turns white (the attribute-value). It could be concluded that every object carries a set of attributes and the similarities and the differences between the objects are determined by the attributes they each carries. Besides `attribute`, `part` is also a significant key philosophy concept in HowNet. It could be understood that all objects are probably parts of something else while at the same time, all objects are also the whole of something else. For example, doors and windows are parts of buildings while meantime buildings are also part of a community. In total, HowNet contains 271 information structure patterns, 58 semantic structure patterns, 11,000 word instances, and 60,000 Chinese words in all (Fig. 5.3).

In addition, HowNet also lays emphasis on sememes in the construction process, which are defined as the minimum semantic units of word meanings, and there exists a limited close set of sememes to compose the semantic meanings of an open set of concepts. HowNet annotates precise senses to each word, and for each sense, HowNet annotates the significance of parts and attributes represented by sememes. For example, the word "apple" actually has two main senses: one is a sort of fruit

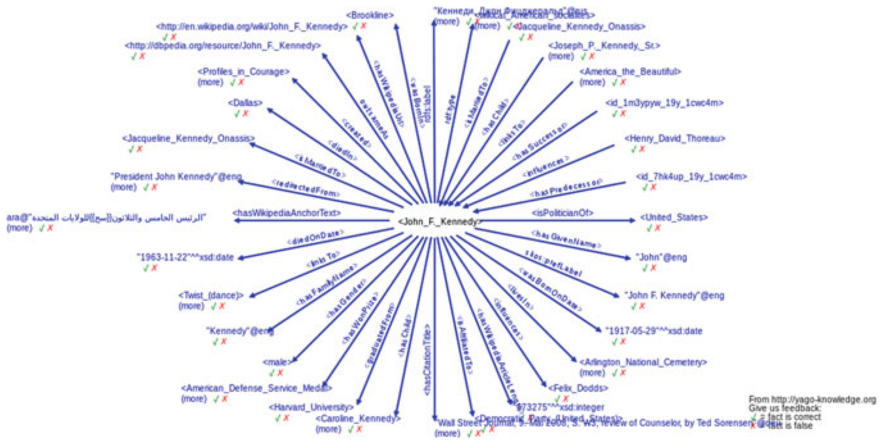


Fig. 5.3 The YAGO page of John F. Kennedy

and the other is a computer brand. Thus, in the first sense, it has the sememe `fruit` and in the second it has sememes `computer`, `bring`, and `SpeBrand`.

5.2 Knowledge Representation Learning

In the past years, various specific algorithms have been designed to store and utilize the information in KG according to its traditional representation (i.e., network representation), which is usually very time-consuming and suffers from data sparsity. Recently, representation learning, which is a subarea of deep learning, has attracted lots of attentions in different areas including natural language processing and artificial intelligence. Representation learning aims at embedding the objects into a dense, low-dimensional, and real-valued semantic space. And knowledge representation learning is a subarea of representation learning, which focuses on embedding the entities and relations in KG.

Recent studies reveal that translation-based representation learning methods are efficient and effective to encode relational facts in KG with low-dimensional representations of both entities and relations, which can alleviate the issue of data sparsity and be further employed to knowledge acquisition, fusion, and inference. TransE (Bordes et al. 2013) is one of typical translation-base knowledge representation learning methods, which learns low-dimensional vectors for both entities and relations and is very simple and effective. TransE regards the relation in a relational triple as a translation between the embeddings of the head and tail entities, that is, $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$ when the triple (h, r, t) holds. And it achieves amazing performance in the task of Knowledge Graph completion.

Although TransE has achieved great success, it still has issues when modeling 1-to-N, N-to-1, and N-to-N relations. The entity embeddings learnt by TransE are lacking in discrimination due to these complex relations. Therefore, how to deal with complex relations is one of the key challenges in knowledge representation learning. Recently, there are lots of extensions of TransE which focus on this challenge. TransH (Wang et al. 2014b) and TransR (Lin et al. 2015b) are proposed to represent an entity with different representations when involved in different relations. TransH models the relation as a translation vector on a hyperplane and projects the entity embeddings into the hyperplane with a normal vector. TransR represents entities in the entity semantic space and uses a relation-specific transform matrix to project it into the different relation spaces when involved in different relations. Further, researchers propose two extension of TransR including TransD (Ji et al. 2015) which considers the information of entities in the projecting matrices and TranSparse (Ji et al. 2016) which considers the heterogeneity and imbalance of relations via sparse matrices. In addition, there are many other extensions of TransE which focus on different characteristics of relations including TransG (Xiao et al. 2015) and KG2E (He et al. 2015) adopt Gaussian embeddings to model both entities and relations; ManifoldE (Xiao et al. 2016) employs a manifold-based embedding principle in knowledge representation learning; and so on.

Besides, TransE still has a problem that only considering direct relations between entities. To address this issue, Lin et al. (2015a) propose Path-based TransE which extends TransE to model relational paths by selecting reasonable relational paths and representing them with low-dimensional vectors. Almost at the same time, there are others researchers considering relational paths in KG successfully (Garcıa-Durán et al. 2015) using neural network. Besides, relational path learning has also been used in the KG-based QA (Gu et al. 2015).

Most existing knowledge representation learning methods discussed above only focus on the structure information in KG, regardless of the rich multisource information such as textual information, type information, and visual information. These cross-modal information can provide supplementary knowledge of the entities specially for those entities with less relational facts and is significant when learning knowledge representations. For textural information, Wang et al. (2014a) and Zhong et al. (2015) propose to jointly embed both entities and words into a unified semantic space by aligning them with entity names, descriptions, and Wikipedia anchors. Further, Xie et al. (2016b) propose to learn entity representations based on their descriptions with CBOW or CNN encoders. For type information, Krompaß et al. (2015) take type information as constraints of head and tail entity set for each relation to distinguish entities which belong to the same types. Instead of merely considering type information as type constraints, Xie et al. (2016c) utilize hierarchical type structures to enhance TransR via guiding the construction of projection matrices. For visual information, Xie et al. (2016a) propose image-embodied knowledge representation learning to take visual information into consideration via learning entity representations using their corresponding figures. It is natural that we learn things in real world with all kinds of multisource information. Multisource information such as plaintexts, hierarchical types, or even images and videos is of great importance when modeling the complicated world and constructing cross-modal representations. Moreover, other types of information could also be encoded into knowledge representation learning to enhance the performance.

5.3 Neural Relation Extraction

To enrich existing KGs, researchers have invested in automatically finding unknown relational facts, i.e., relation extraction (RE). Relation extraction aims at extracting relational data from plaintexts. In recent years, as the development of deep learning (Bengio 2009) techniques, neural relation extraction adopts an end-to-end neural network to model the relation extraction task. The framework of neural relation extraction includes a sentence encoder to capture the semantic meaning of the input sentence and represents it as a sentence vector, and a relation extractor to generate the probability distribution of extracted relations according to sentence vectors. We will give an in-depth review of recent works on neural relation extraction.

Neural relation extraction (NRE) has two main tasks including sentence-level NRE and document-level NRE. In this section, we will introduce these two tasks in detail, respectively.

5.3.1 Sentence-Level NRE

Sentence-level NRE aims at predicting the semantic relations between the entity (or nominal) pair in a sentence. Formally, given the input sentence x which consists of m words $x = (w_1, w_2, \ldots, w_m)$ and its corresponding entity pair e_1 and e_2 as inputs, sentence-level NRE wants to obtain the conditional probability $p(r|x, e_1, e_2)$ of relation r ($r \in \mathbb{R}$) via a neural network, which can be formalized as

$$p(r|x, e_1, e_2) = p(r|x, e_1, e_2, \theta), \tag{5.1}$$

where θ is parameter of the neural network, and r is a relation in the relation set $\mathbf{R}$.

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

5.3.1.1 Input Encoder

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

Word embeddings learn low-dimensional real-valued representation of words, which can reflect syntactic and semantic relationships between words. Formally, each word w_i is encoded by the corresponding column vector in an embedding matrix $\mathbf{V} \in \mathbb{R}^{d^a \times |V|}$, where V indicates a fix-sized vocabulary.

Position embeddings aim to specify the position information of the word with respect to two corresponding entities in the sentence. Formally, each word w_i is encoded by two position vectors with respect to the relative distances from the word to two target entities, respectively. For example, in the sentence *New York* is a city of *United States*, the relative distance from the word city to *New York* is 3 and *United States* is -2.

Part-of-speech tag embeddings represent the lexical information of target word in the sentence. Due to the fact that word embeddings are obtained from a generic corpus on a large scale, the information they contain may not be in accordance with the meaning in a specific sentence, it is necessary to align each word with its linguistic information, e.g., noun, verb, etc. Formally, each word w_i is encoded by the corresponding column vector in an embedding matrix $\mathbf{V}^p \in \mathbb{R}^{d^p \times |V^p|}$, where d^p

is the dimension of embedding vector and V^p indicates a fix-sized part-of-speech tag vocabulary.

WordNet hypernym embeddings aim to take advantages of the prior knowledge of hypernym to contribute to relation extraction. It is easier to build the link between different but conceptual similar words when given each word's hypernym information in WordNet, e.g., noun.food, verb.motion, etc. Formally, each word w_i is encoded by the corresponding column vector in an embedding matrix $\mathbf{V}^h \in \mathbb{R}^{d^h \times |V^h|}$, where d^h is the dimension of embedding vector and V^h indicates a fix-sized hypernym vocabulary.

5.3.1.2 Sentence Encoder

Next, the sentence encoder encodes input representations into either a single vector or a sequence of vectors $\mathbf{x}$. We will introduce the different sentence encoders in the following.

Convolution neural network encoder (Zeng et al. 2014) is proposed to embed input sentence using a convolutional neural network (CNN) which extracts local feature by a convolution layer and combines all local features via a max-pooling operation to obtain a fixed-sized vector for the input sentence. Formally, as illustrated in Fig. 5.4, convolution operation is defined as a matrix multiplication between a sequence of vectors and a convolution matrix $\mathbf{W}$ and a bias vector $\mathbf{b}$ with a sliding window. Let us define the vector $\mathbf{q}_i$ as the concatenation of a sequence of input representations in the i-th window, we have

$$[\mathbf{x}]_j = \max_i [f(\mathbf{W}\mathbf{q}_i + \mathbf{b})]_j, \tag{5.2}$$

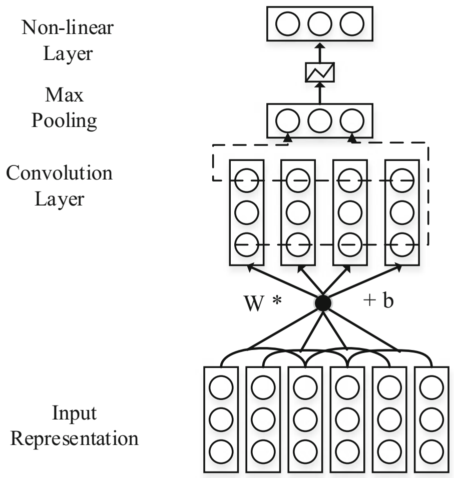


Fig. 5.4 The architecture of CNN encoder

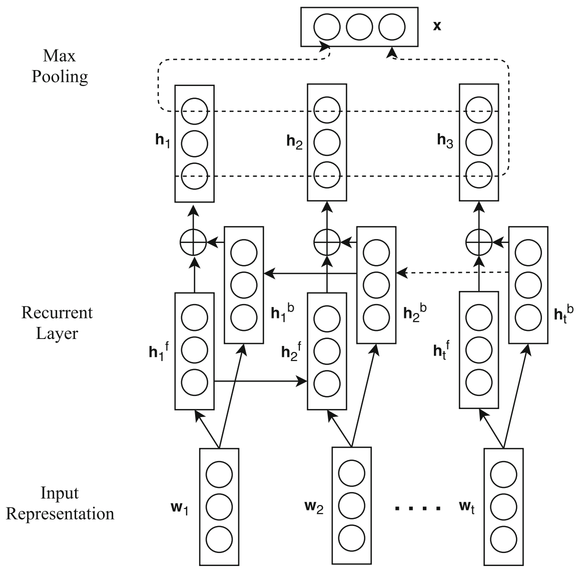


Fig. 5.5 The architecture of recurrent encoder

where f indicates a nonlinear function such as sigmoid or tangent function.

Further, to better capture the structural information between two entities, piecewise max-pooling operation (Zeng et al. 2015) is proposed instead of traditional max-pooling operation. Piecewise max-pooling operation returns the maximum value in three segments of the input sentence which are divided into two target entities.

Recurrent neural network encoder (Zhang and Wang 2015) is proposed to embed input sentence using a recurrent neural network (RNN) which has the capability to learn the temporal features. As illustrated in Fig. 5.5, each word representation vectors are put into recurrent layer step-by-step. For each step i, the network takes the word representation vector $\mathbf{w}_i$ and the previous step $i-1$'s output $\mathbf{h}_{i-1}$ as inputs, and then we have

$$\mathbf{h}_i = f(\mathbf{w}_t, \mathbf{h}_{i-1}), \tag{5.3}$$

where f indicates the transform function inside the RNN cell, which can be the LSTM units (Hochreiter and Schmidhuber 1997) (LSTM-RNNs) or the GRU units (Cho et al. 2014) (GRU-RNNs). In addition, a bidirectional RNN network is employed to fully utilize the information of future words when predicting the semantic meaning in the middle of a sentence.

Next, RNN combines the information from forward and backward network as a local feature and uses a max-pooling operation to extract the global feature, which forms the representation of the whole input sentence. The max-pooling layer could be formulated as

$$[\mathbf{x}]_j = \max_i [\mathbf{h}_i]_j. \tag{5.4}$$

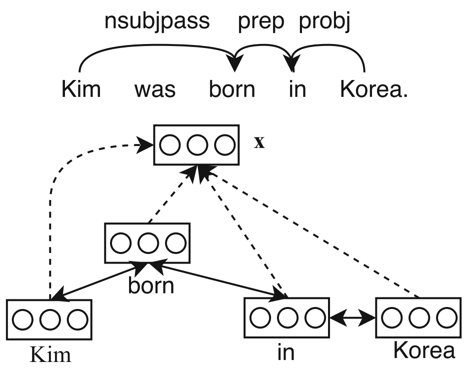


Fig. 5.6 The architecture of dependency tree-structured LSTM

Besides max-pooling, word attention can also combine all local feature vectors together. It uses attention mechanism (Bahdanau et al. 2014) to learn attention weights on each step. Suppose $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_m]$ is the matrix consisting of all output vectors that produced by the recurrent layer, the whole sentence's feature vector $\mathbf{x}$ is formed by a weighted sum of each step's output:

$$\alpha = \text{softmax}(\mathbf{s}^T \tanh(\mathbf{H})) \tag{5.5}$$

$$\mathbf{x} = \mathbf{H}\alpha^T, \tag{5.6}$$

where $\mathbf{s}$ is a trainable query vector and s^T indicates its transposition.

Besides, Miwa and Bansal (2016) proposed a model that captures both word sequence and dependency tree substructure information by stacking bidirectional path-based LSTM-RNNs (i.e., bottom-up and top-down) on bidirectional sequential LSTM-RNNs. As illustrated in Fig. 5.6, it focuses on the shortest path between the target entities in the dependency tree because experimental result in (Xu et al. 2015) shows that these paths are effective in relation classification.

Recursive neural network encoder aims to extract features from the information of syntactic parsing tree structure because the syntactic information is important for extracting relations from sentences. Generally, these encoders treat the tree structure inside the syntactic parsing tree as a strategy of composition as well as direction for recursive neural network to combine each word's embedding vector.

Socher et al. (2012) proposed a recursive matrix-vector model (MV-RNN) which captures constituent parsing tree structure information by assigning a matrix-vector representation for each constituent. The vector captures the meaning of constituent itself and the matrix represents how it modifies the meaning of the word it combines with. Suppose we have two children components l, r and their father component p, the composition can be formulated as follows:

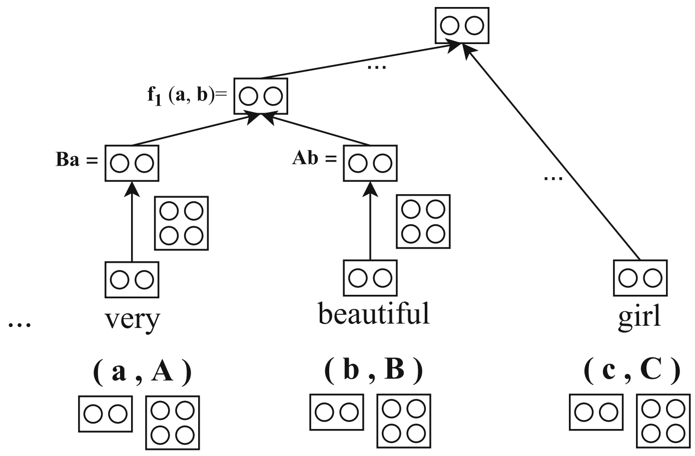


Fig. 5.7 The architecture of matrix-vector recursive encoder

$$\mathbf{p} = f_1(l, r) = g\left(\mathbf{W}_1 \begin{bmatrix} \mathbf{Ba} \\ \mathbf{Ab} \end{bmatrix}\right) \tag{5.7}$$

$$\mathbf{P} = f_2(l, r) = \mathbf{W}_2 \begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix}, \tag{5.8}$$

where $\mathbf{a}, \mathbf{b}, \mathbf{p}$ are embedding vectors for each components and $\mathbf{A}, \mathbf{B}, \mathbf{P}$ are matrices, $\mathbf{W}_1$ is a matrix that maps transformed words into another semantic space, the element-wise function g is an activation function, and $\mathbf{W}_2$ is a matrix that maps two matrices into one combined matrix P with the same dimension. The whole process is illustrated in Fig. 5.7. And then, MV-RNN selects the highest node of the path in the parse tree between the two target entities to represent the input sentence.

In fact, the RNN unit here can be replaced by LSTM units or GRU units. Tai et al. (2015) propose two types of tree-structured LSTMs including the Child-Sum Tree-LSTM and the N-ary Tree-LSTM to capture constituent or dependency parsing tree structure information. For the Child-Sum Tree-LSTM, given a tree, let $C(t)$ denote the set of children of node t. Its transition equations are defined as follows:

$$\hat{\mathbf{h}}_t = \sum_{k \in C(t)} \mathbf{h}_k, \tag{5.9}$$

$$\mathbf{i}_t = \sigma(\mathbf{W}^{(i)}\mathbf{w}_t + \mathbf{U}^i\hat{\mathbf{h}}_t + \mathbf{b}^{(i)}), \tag{5.10}$$

$$\mathbf{f}_{tk} = \sigma(\mathbf{W}^{(f)}\mathbf{w}_t + \mathbf{U}^f\mathbf{h}_k + \mathbf{b}^{(f)}) \ \ (k \in C(t)), \tag{5.11}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}^{(o)}\mathbf{w}_t + \mathbf{U}^o\hat{\mathbf{h}}_t + \mathbf{b}^{(o)}), \tag{5.12}$$

$$\mathbf{u}_t = \tanh(\mathbf{W}^{(u)}\mathbf{w}_t + \mathbf{U}^u\hat{\mathbf{h}}_t + \mathbf{b}^{(u)}), \tag{5.13}$$

$$\mathbf{c}_t = \mathbf{i}_t \odot \mathbf{u}_t + \sum_{k \in C(t)} \mathbf{f}_{tk} \odot \mathbf{c}_{t-1}, \tag{5.14}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t). \tag{5.15}$$

The N-ary Tree-LSTM has similar transition equations with the Child-Sum Tree-LSTM. The only difference is that it limits the tree structures has at most N branches.

5.3.1.3 Relation Classifier

Finally, when obtaining the representation $\mathbf{x}$ of the input sentence, relation classifier calculates the conditional probability $p(r|x, e_1, e_2)$ via a softmax layer as follows:

$$p(r|x, e_1, e_2) = \text{softmax}(\mathbf{M}\mathbf{x} + \mathbf{b}), \tag{5.16}$$

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

5.3.2 Document-Level NRE

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

Therefore, document-level NRE aims to predict the semantic relations between an entity pair using all involved sentences. Given the input sentence set S which consists of n sentences $S = (x_1, x_2, \ldots, x_n)$ and its corresponding entity pair e_1 and e_2 as inputs, document-level NRE wants to obtain the conditional probability $p(r|S, e_1, e_2)$ of relation r $(r \in \mathbb{R})$ via a neural network, which can be formalized as

$$p(r|S, e_1, e_2) = p(r|S, e_1, e_2, \theta). \tag{5.17}$$

A basic form of document-level NRE consists of four components: (a) an input encoder similar to sentence-level NRE, (b) a sentence encoder similar to sentence-level NRE, (c) a document encoder which computes either vector representing all related sentences, and (d) a relation classifier similar to sentence-level NRE which takes document vector as input instead of sentence vector. In this next, we will introduce the document encoder and in detail.

5.3.2.1 Document Encoder

The document encodes all sentence vectors into either single vector **S**. We will introduce the different document encoders in the following.

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

$$\mathbf{S} = \mathbf{x}_i \ (i = 1, 2, \ldots, n), \tag{5.18}$$

where $\mathbf{x}_i$ indicates the sentence representation of x_i and i is a random index.

Max Encoder. In fact, as introduced above, not all sentences containing two target entities can express their relations. For example, the sentence "New York City is the premier gateway for legal immigration to the United States" does not express the relation `city_of`. Hence, in (Zeng et al. 2015), they follow the at-least-one assumption which assumes that at least one sentence that contains these two target entities can express their relations, and select the sentence with highest probability for the relation to represent the document. Formally, the document representation is defined as

$$\mathbf{S} = \mathbf{x}_i \ (i = \text{argmax}_i \, p(r|x_i, e_1, e_2)). \tag{5.19}$$

Average Encoder. Both random encoder or max encoder use only one sentence to represent the document, which ignores the rich information of different sentences. To exploit the information of all sentences, Lin et al. (2016) believe that the representation **S** of the document depends on all sentences' representations $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$. Each sentence representation $\mathbf{x}_i$ can give the relation information about two entities for input sentence x_i. The average encoder assumes that all sentences contribute equally to the representation of the document. It means the embedding **S** of the document is the average of all the sentence vectors:

$$\mathbf{S} = \sum_i \frac{1}{n} \mathbf{x}_i. \tag{5.20}$$

Attentive Encoder. Due to the wrong label issue brought by distant supervision assumption inevitably, the performance of average encoder will be influenced by those sentences that contain no related information. To address this issue, Lin et al. (2016) further propose to employ a selective attention to de-emphasize those noisy sentence. Formally, the document representation is defined as a weighted sum of sentence vectors:

$$\mathbf{S} = \sum_i \alpha_i \mathbf{x}_i, \tag{5.21}$$

where α_i is defined as

$$\alpha_i = \frac{\exp(\mathbf{x}_i \mathbf{A} \mathbf{r})}{\sum_j \exp(\mathbf{x}_j \mathbf{A} \mathbf{r})}, \tag{5.22}$$

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

5.3.2.2 Relation Classifier

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

$$p(r|S, e_1, e_2) = \text{softmax}(\mathbf{M}'\mathbf{S} + \mathbf{b}'), \tag{5.23}$$

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

5.4 Bridging Knowledge with Text: Entity Linking

Knowledge Graph contains rich knowledge about the world's entities, their attributes, and semantic relations between different entities. Bridging Knowledge Graph with textual data can facilitate many different tasks, such as information extraction, text classification, and question answering. For example, it is helpful for understanding "*Jobs leaves Apple*" if we knew "*Steve Jobs* is CEO of *Apple Inc.*".

Currently, the main research issue in bridging Knowledge Graph with textual data is *entity linking (EL)* (Ji et al. 2010). Given a set of name mentions $M = \{m_1, m_2, \ldots, m_k\}$ in a document d, and a Knowledge Graph KB containing a set of entities $E = \{e_1, e_2, \ldots, e_n\}$, an entity linking system is a function $\delta : M \rightarrow E$ which maps name mentions to their referent entities in KB. Fig. 5.8 shows an example, where an EL system will identify the referent entities of the three entity mentions *WWDC, Apple, and Lion* correspondingly are *Apple Worldwide Developers Conference, Apple Inc.* and*, Mac OS X Lion*. Based on the entity linking results, all knowledge about these entities in KB can be used to understand the text, for example, we can classify the given document into *IT* category, rather than into *Animal* category based on the knowledge "*Lion is an Operation System*".

The main challenges for entity linking are the *name ambiguity problem* and the *name variation problem*. The name ambiguity problem is related to the fact that a name may refer to different entities in different contexts. For example, the name *Apple* can refer to more than 20 entities in Wikipedia, such as *fruit Apple, the IT company Apple Inc.*, and *the Apple Bank*. The name variation problem means that an entity can be mentioned in different ways, such as its full name, aliases, acronyms, and misspellings. For example, the IBM company can be mentioned using more than 10 names, such as *IBM, International Business Machine*, and its nickname *Big Blue*.

To solve the name ambiguity problem and the name variation problem, many approaches have been proposed for entity linking (Milne and Witten 2008; Kulkarni et al. 2009; Ratinov et al. 2011; Han and Sun 2011; Han et al. 2011; Han and Sun 2012). In the following, we first describe a general framework for entity linking, and then we introduce how deep learning techniques can be used to enhance EL performance.

5.4.1 The Entity Linking Framework

Given a document d and a Knowledge Graph KB, an entity linking system links the name mentions in the document as follows.

Name Mention Identification. In this step, all name mentions in a document will be identified for entity linking. For example, an EL system should identify three mentions {*WWDC, Apple, Lion*} from the document in Fig. 5.8. Currently, most EL systems employ two techniques for this task. One is the classical named entity recognition (NER) technique (Nadeau and Sekine 2007), which can recognize names of *Person, Location*, and *Organization* in a document, and then these entity names will be used as name mentions for entity linking. The main drawback of NER technique is that it can only identify limited types of entities, while ignores many

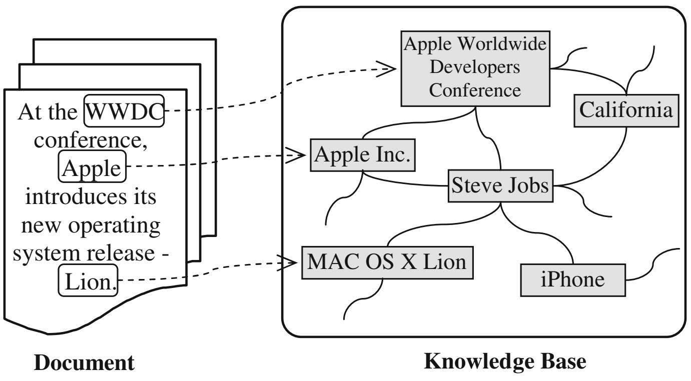

At the WWDC conference, Apple introduces its new operating system release - Lion.
Apple Worldwide Developers Conference
California
Apple Inc.
Steve Jobs
MAC OS X Lion
iPhone
Document
Knowledge Base

Fig. 5.8 A demo of entity linking

commonly used entities such as *Music, Film*, and *Book*. The other technique for name mention detection is dictionary-based matching, which first constructs a name dictionary for all entities in a Knowledge Graph (e.g., collected from anchor texts in Wikipedia Mihalcea and Csomai 2007), and then all names matched in a document will be used as name mentions. The main drawback of dictionary-based matching is that it may match many noisy name mentions, e.g., even the stop words *is* and *an* are used as entity names in Wikipedia. To resolve this problem, many techniques (Mihalcea and Csomai 2007; Milne and Witten 2008) have been proposed to filter out noisy name mentions.

Candidate Entity Selection. In this step, an EL system selects candidate entities for each name mention detected in Step 1. For example, a system may identify {*Apple(fruit), Apple Inc., Apple Bank*} as the possible referents for name *Apple*. Due to the name variation problem, most EL systems rely on a reference table for candidate entity selection. Specifically, a reference table records all possible referents of a name using (*name, entity*) pairs, and reference tables can be collected from Wikipedia anchor texts (Milne and Witten 2008), web (Bollegala et al. 2008), or query log (Silvestri et al. 2009).

Local Compatibility Computation. Given a name mention m in document d and its candidate referent entities $E = \{e_1, e_2, \ldots, e_n\}$, a critical step of EL systems is to compute the local compatibility $sim(m, e)$ between mention m and entity e, i.e., estimate how likely the mention m will be linked to the entity e. Based on the local compatibility scores, a name mention m will be linked to the entity which has the largest compatibility score with it:

$$e^* = \text{argmax}_e \quad \text{sim}(m, e). \tag{5.24}$$

For example, to determine the referent entity of the name *apple* in the following sentence:

The ***apple*** *tree is a deciduous tree in the rose family*

we need to compute its compatibility with entities *Apple(fruit)* and *Apple Inc.*, and finally link *apple* with *Apple(fruit)* based on the contextual words "*tree*", "*rose family*", etc.

Currently, many approaches have been proposed for local compatibility computation (Milne and Witten 2008; Mihalcea and Csomai 2007; Han and Sun 2011). The essential idea is to extract discriminative features (e.g., important words, frequent co-occur entities, attribute values) from the mention's context and the description of a specific entity (e.g., the Wikipedia page of the entity), and then the compatibility is determined by their shared common features.

Global Inference. It has long been proven that global inference can significantly increase the performance of entity linking. The underlying assumption of global inference is the topic coherence assumption, i.e., *all entities in a document should semantically related to the document's main topics*. Based on this assumption, a referent entity should not only compatible with its local context but also should coherent with other referent entities in the same document. For example, if we know the referent entity of the name mention *Lion* is *Mac OSX(Lion)* in Fig. 5.8, we can easily determine the referent entity of *Apple* is *Apple Inc.* using the semantic relation Product-of(*Apple Inc., Mac OSX(Lion)*). These examples strongly suggest that the entity linking performance could be improved by resolving the entity linking problems in the same document jointly, rather than independently.

Formally, given all mentions $M = \{m_1, m_2, \ldots, m_k\}$ in a document d, a global inference algorithm aims to find the optimal referent entities which will maximize the global coherence score:

$$[e_1^*, \ldots, e_k^*] = \text{argmax}\left(\sum_i \text{sim}(m_i, e_i) + \text{Coherence}(e_1, e_2, \ldots, e_k)\right). \quad (5.25)$$

In recent years, many global inference algorithms have been proposed for entity linking, including graph-based algorithms (Han et al. 2011; Chen and Ji 2011), topic model-based methods (Ganea et al. 2016; Han and Sun 2012), and optimization-based algorithms (Ratinov et al. 2011; Kulkarni et al. 2009). These methods differ with each other in how their model the document coherence, and how they infer the global optimal EL decisions. For example, Han et al. (2011) model the coherence as the sum of semantic relatedness between all referent entities:

$$\text{Coherence}(e_1, e_2, \ldots, e_k) = \sum_{(i,j)} \text{SemanticRelatedness}(e_i, e_j) \quad (5.26)$$

then the global optimal decisions are obtained through a graph random walk algorithm. By contrast, Han and Sun (2012) propose an entity-topic model, where the coherence is modeled as the probability of generating all referent entities from a document's main topics, and the global optimal decisions are obtained through a Gibbs sampling algorithm.

5.4.2 Deep Learning for Entity Linking

In this section, we introduce how to employ deep learning techniques for entity linking. As introduced above, one main problem of EL is the name ambiguity problem; thus, the key challenge is how to compute the compatibility between a name mention and an entity by effectively using contextual evidences.

It has been observed that the performance of entity linking heavily depend on the local compatibility model. Existing studies typically use handcrafted features to represent different types of contextual evidences (e.g., mention, context, and entity description), and measure the local compatibility using heuristic similarity measures (Milne and Witten 2008; Mihalcea and Csomai 2007; Han and Sun 2011). These feature-engineering-based approaches, however, have the following drawbacks:

- Feature engineering is labor-intensive, and it is difficult to manually design discriminative features. For example, it is challenging to design features which can capture the semantic similarity between the words *cat* and *dog*.
- The contextual evidences for entity linking are usually heterogeneous and may be at different granularities. The modeling and exploitation of heterogeneous evidences are not straightforward using handcrafted features. Till now, many different kinds of contextual evidences have been used for entity linking, including entity name, entity category, entity description, entity popularity, semantic relations between entities, mention name, mention context, mention document, etc. It is hard to design features which can project all these evidences into the same feature space, or to summarize all these evidences into a uniform framework for EL decisions.
- Finally, traditional entity linking methods usually define the compatibility between a mention and an entity heuristically, which is weak in discovering and capturing all useful factors for entity linking decisions.

To resolve the above drawbacks of feature-engineering-based approaches, in recent years many deep learning techniques have been employed for entity linking (He et al. 2013; Sun et al. 2015; Francis-Landau et al. 2016; Tsai and Roth 2016). In following, we first describe how to represent heterogeneous evidences via neural networks, then we introduce how to model the semantic interactions between different types of contextual evidences, and finally, we describe how to optimize local compatibility measures for entity linking using deep learning techniques.

5.4.2.1 Representing Heterogeneous Evidences via Neural Networks

One main advantage of neural network is it can learn good representations automatically from different types of raw inputs, such as text, image, and video (Bengio 2009). In entity linking, neural networks have been exploited to represent heterogeneous contextual evidences, such as mention name, mention context and entity description. By encoding all contextual evidences in the continuous vector space which are suitable for entity linking, neural networks avoid the need of designing handcrafted features. In following, we introduce how to represent different types of contextual evidences in detail.

Name Mention Representation. A mention $m = [m_1, m_2, ...]$ is typically composed of one to three words, such as *Apple Inc., President Obama*. Previous methods mostly represent a mention as the average of embeddings of the words it contains

$$\mathbf{v}_m = \text{average}(\mathbf{e}_{m_1}, \mathbf{e}_{m_2}, \ldots), \tag{5.27}$$

where $\mathbf{e}_{m_i}$ is the embeddings of word m_i, which can be learned using CBOW or Skip-Gram models (Mikolov et al. 2013).

The above embedding average representation fails to take the importance and the position of a word into consideration. To resolve this problem, some methods employ

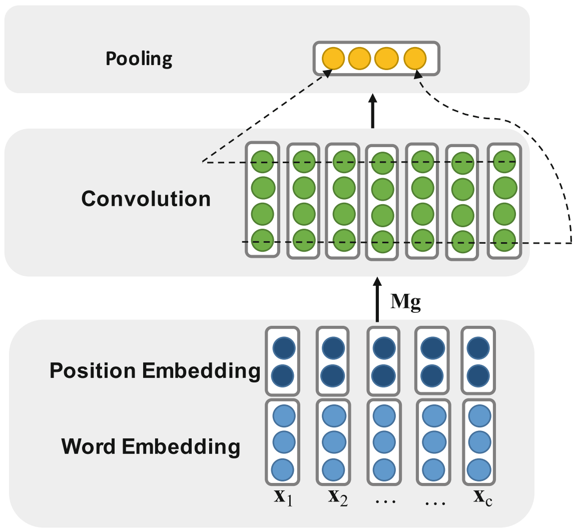


Fig. 5.9 Representing local context via convolutional neural network

convolutional neural networks (CNN) (Francis-Landau et al. 2016) to represent a mention, which provides more flexible ability to represent name mentions.

Local Context Representation. The local context around a mention provides critical information for entity linking decisions. For example, the context words {*tree, deciduous, rose family*} in "*The **apple** tree is a deciduous tree in the rose family*" provide critical information for linking the name mention *apple*. Sun et al. (2015) propose to represent local context using CNN, where the representation of a context is composed of the words it contains, by taking both the semantics of words and their relative positions to the mention into consideration.

Figure 5.9 demonstrates how to represent local context using CNN. Formally, given the words in a context $c = [w_1, w_2, \ldots, w_{|c|}]$, we represent each word w as $\mathbf{x} = [\mathbf{e}_w, \mathbf{e}_p]$, where $\mathbf{e}_w$ is the embeddings of word w and $\mathbf{e}_p$ is the position embeddings of word w, with d_w and d_p are the dimensions of word vector and position vector. A word w_i's position is its distance to the mention in the local context.

To represent the context c, we first concatenate all vectors of its words as

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{|c|}] \tag{5.28}$$

then a convolution operation is applied to $\mathbf{X}$, and the output of convolution layer is

$$\mathbf{Z} = [\mathbf{M_g}\mathbf{X}_{[1,K+1]}, \mathbf{M_g}\mathbf{X}_{[2,K+2]}, \ldots, \mathbf{M_g}\mathbf{X}_{[|c|-K,|c|]}], \tag{5.29}$$

where $\mathbf{M_g} \in \mathbb{R}^{n_1 \times n_2}$ is the linear transformation matrix, and K is the context size of convolution layer.

Since the local context is of variable length, and in order to determine the most useful feature in each dimension of the feature vector, we perform a max-pooling operation(or other pooling operations) to the output of the convolution layer as

$$m_i = \max \quad \mathbf{Z}(i, .) \quad 0 \leq i \leq |c|. \tag{5.30}$$

Finally, we use the vector $\mathbf{m}_c = [m_1, m_2, \ldots]$ to represent the local context c of mention m.

Document Representation. As described in previous researches (He et al. 2013; Francis-Landau et al. 2016; Sun et al. 2015), the document and the local context of a name mention provide information at different granularities for entity linking. For example, a document usually captures larger topic information than local context. Based on this observation, most entity linking systems treat document and local context as two different evidences, and learn their representations individually.

Currently, two types of neural networks have been exploited for document representation in entity linking. The first is the convolutional neural network (Francis-Landau et al. 2016; Sun et al. 2015), which is the same as we introduced in local context representation. The second is denoising autoencoder (DA) (Vincent et al. 2008), which seeks to learn a compact document representation which can retain maximum information in original document d. Specifically, a document is first represented as a binary bag-of-words vector $\mathbf{x}_d$ (He et al. 2013), where each dimension of $\mathbf{x}$ indicates whether word w_i is appeared. Given the document representation $\mathbf{x}$, a denoising autoencoder seeks to learn a model which can reconstruct $\mathbf{x}$ given a random corruption $\mathbf{x}'$ of $\mathbf{x}$ through the following process: (1) randomly corrupt $\mathbf{x}$ by applying masking noise(randomly mask 1 or 0) to the original $\mathbf{x}$; (2) encode $\mathbf{x}$ into a compact representation $h(\mathbf{x})$ through an encoding process; (3) reconstruct $\mathbf{x}$ from $h(\mathbf{x})$ through a decoding process $g(h(\mathbf{x}))$. The learning goal of DA is to minimize the reconstruction error $L(x, g(h(\mathbf{x})))$. Figure 5.10 demonstrates the encoding and decoding process of DA.

DA has several advantages for document representation (He et al. 2013). First, the autoencoder tries to learn a compact representation of a document, and therefore can group similar words into clusters. Second, by randomly corrupting original inputs, DA can capture general topics and ignore meaningless words, such as function words

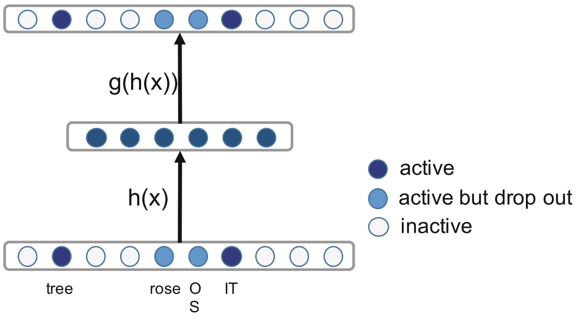


Fig. 5.10 DA and reconstruction sampling

is, and, or, etc. Third, autoencoder can be repeatedly stacked on top of previous learned $h(\mathbf{x})$; therefore, DA can learn multiple levels of representation of a document.

Entity Knowledge Representation. Currently, most entity linking systems use Wikipedia (or Knowledge Bases derived from Wikipedia, such as Yago, DBPedia, etc.) as its target Knowledge Base. Wikipedia contains rich knowledge about entities, such as title, description, infobox containing its important attributes, semantic categories, and sometimes its relations with other entities. For example, Fig. 5.11 shows *Apple Inc.*'s knowledge contained in Wikipedia. In following, we describe how to represent the evidence from entity knowledge using neural networks.

- **Entity Title Representation**. As the same with name mention, an entity title is typically composed of one to three words; therefore, most entity linking systems employ the same neural networks as in name mention representation to represent entity titles, i.e., average of word embeddings or CNN.
- **Entity Description**. Currently, most entity linking systems model entity description as a plain document, and learn its representation as the same with document representation, i.e., via CNN or DA.

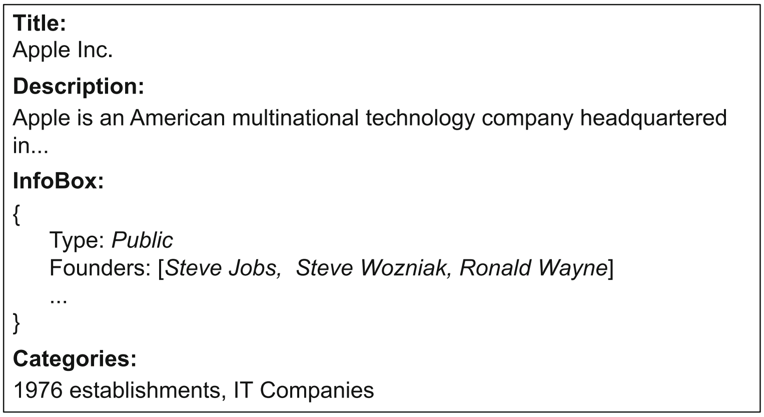
From the above introduction, deep learning techniques propose a family of neural networks for representing contextual evidences, from word embeddings, denoising auto-encoder, to convolutional neural networks. These neural networks can effectively learn the representations of contextual evidences, without the need of handcrafted features.

In recent years, many other types of evidences have also been exploited for entity linking. For instance, entity popularity which tells the likelihood of an entity appearing in a document, semantic relations which capture the semantic association/relation between different entities (e.g., CEO-of(*Steve Jobs, Apple Inc.*) and Employee-of(*Michael I. Jordan, UC Berkeley*)), categories which provide key generalization information for an entity(e.g., *apple ISA fruit, Steve Jobs is a Businessman, Michael Jeffery Jordan ISA NBA player*). The representation of these contextual evidences using neural networks is still not straightforward. For future work, it may be helpful to design other neural networks which can effectively represent these contextual evidences.

5.4.2.2 Modeling Semantic Interactions Between Contextual Evidences

As shown in above, there exist many types of contextual evidences for entity linking. To make accurate EL decision, an EL system needs to take all different types of contextual evidences into consideration. Furthermore, in recent years, the task of cross-lingual entity linking makes it essential to compare contextual evidences in different languages. For example, an EL system needs to compare the Chinese name mention "***pingguo****(Apple) fabu(released) xin(new) iPhone*" with the English description of "*Apple Inc.*" in Wikipedia for Chinese-to-English entity linking.

To take all contextual evidences into consideration, recent studies have employed neural networks to model the semantic interactions between different context evi-



Title:
Apple Inc.

Description:
Apple is an American multinational technology company headquartered in...

InfoBox:
{
    Type: *Public*
    Founders: [*Steve Jobs, Steve Wozniak, Ronald Wayne*]
    ...
}

Categories:
1976 establishments, IT Companies

Fig. 5.11 The information of *Apple Inc.* in Wikipedia

dences. Generally, two strategies have been used to model the semantic interactions between different contextual evidences:

- The first is to map different types of contextual evidences to the same continuous feature space via neural networks, and then the semantic interactions between contextual evidences can be captured using the similarities (mostly the cosine similarity) between their representations.
- The second is to learn a new representation which can summarize information from different contextual evidences, and then to make entity linking decisions based on the new representation.

In following, we describe how these two strategies are used in entity linking systems.

In Francis-Landau et al. (2016), it learns convolutional neural networks to project name mention, mention's local context, source document, entity title, and entity description into the same continuous feature space; then, the semantic interactions between different evidences are modeled as the similarities between their representations. Specifically, given the continuous vector representations learned by CNN, Francis-Landau et al. (2016) capture the semantic interactions between a mention and an entity as

$$\mathbf{f}(c, e) = [\cos(s_d, e_n), \cos(s_c, e_n), \cos(s_m, e_n), \cos(s_d, e_d), \cos(s_c, e_d), \cos(s_m, e_d)], \tag{5.31}$$

where s_d, s_m, and s_c correspondingly are the learned vectors of mention's document, context, and name, and e_n and e_d are correspondingly the learned vectors of entity's name and description. Finally, the above semantic similarities are combined with other signals such as link counts to predict the local compatibility.

In Sun et al. (2015), it learns a new representation for every mention, which consists of evidences from mention's name and local context based on their representations. Specifically, the new representation uses neural tensor network to compose mention vector($\mathbf{v}_m$) and context vector($\mathbf{v}_c$):

$$\mathbf{v}_{mc} = [\mathbf{v}_m, \mathbf{v}_c]^T [M_i^{appr}]^{[1,L]} [\mathbf{v}_m, \mathbf{v}_c]. \tag{5.32}$$

In this way, the semantic interactions between different contextual evidences are summarized into the new feature vector $\mathbf{v}_{mc}$. Sun et al. (2015) also learn a new representation for each entity by composing its entity name representation and entity category representation. Finally, the local compatibility between a mention and an entity is calculated as the cosine similarity between their new representations.

In Tsai and Roth (2016), it proposes a multilingual embedding method for cross-lingual entity linking. Cross-lingual entity linking aims to ground mentions written in non-English documents to entries in the English Wikipedia. Tsai and Roth (2016) project words and entity names in both the foreign language and in English into a new continuous vector space, and then the similarity between a foreign language mention and English Wikipedia entries can be effectively calculated for entity linking. Specifically, given the embeddings of the aligned English and foreign language titles $\mathbf{A}_{en} \in \mathbb{R}^{a \times k_1}$ and $\mathbf{A}_f \in \mathbb{R}^{a \times k_2}$, where a is the aligned title number, k_1 and k_2 correspondingly are the embedding dimensions of English and foreign language, Tsai and Roth (2016) apply a canonical correlation analysis (CCA) to these two matrices:

$$[\mathbf{P}_{en}, \mathbf{P}_f] = \text{CCA}(\mathbf{A}_{en}, \mathbf{A}_f). \tag{5.33}$$

Then, the English embeddings and the foreign language embeddings are projected into a new feature space as

$$\mathbf{E}'_{en} = \mathbf{E}_{en}\mathbf{P}_{en}, \tag{5.34}$$

$$\mathbf{E}'_f = \mathbf{E}_f\mathbf{P}_f, \tag{5.35}$$

where $\mathbf{E}_{en}$ and $\mathbf{E}_f$ is the original embeddings of all words in English and foreign language, and $\mathbf{E}'_{en}$ and $\mathbf{E}'_f$ is the new embeddings of all words in English and foreign language.

5.4.2.3 Learning Local Compatibility Measures

Both the contextual evidence representation learning and the semantic interaction modeling rely on a large set of parameters for good performance. Deep learning techniques provide an end-to-end framework, which can effectively optimize all parameters using back-propagation algorithm and gradient-based optimization algorithms. In Fig. 5.12, we show a commonly used architecture for local compatibility learning. We can see that mention's evidence and entity's evidence will be first encoded into a continuous feature space using contextual evidence representation neural networks, then compatibility signals between mention and entity will be computed using semantic interaction modeling neural networks, and finally, all these signals will be summarized into the local compatibility score.

To learn the above neural network for local compatibility, we need to collect entity linking annotations (d, e, m) from different resources, e.g., from Wikipedia hyperlinks. Then, the training objective is to minimize the ranking loss:

$$L = \sum_{(m,e)} L(m, e), \tag{5.36}$$

where $L(m, e) = \max\{0, 1 - sim(m, e) + sim(m, e')\}$ is the pairwise ranking criterion for each training instance (m, e), which gives a penalize if the top 1 ranked entity e' is not the true referent entity e.

We can see that, in the above learning process, deep learning techniques can optimize the similarity measure by fine-tuning the mention representation and entity representation, and learning the weights for different compatibility signals. In this way, it usually can achieve better performance than heuristically designed similarity measures.

5.5 Summary

Knowledge Graph is a fundamental knowledge repository for natural language understanding and commonsense reasoning, which contains rich knowledge about the world's entities, their attributes, and semantic relations between entities.

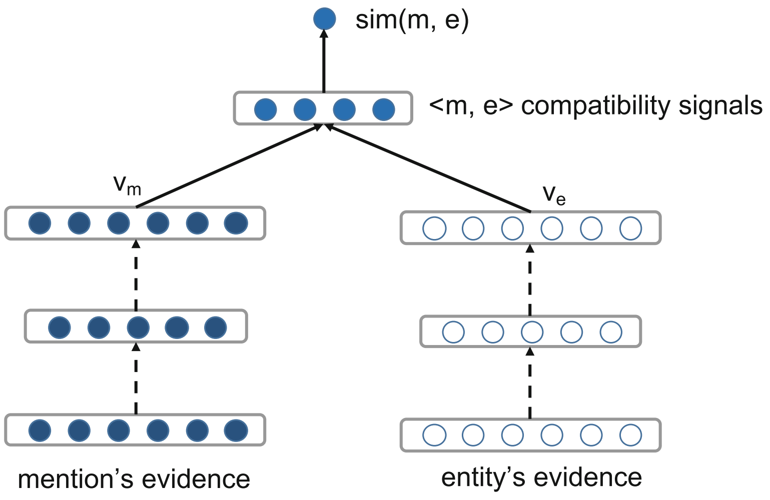


Fig. 5.12 A general framework for local compatibility learning

In this chapter, we introduce several important Knowledge Graphs, including DBPedia, Freebase, Wikidata, Yago, and HowNet. Afterwards, we introduce three important tasks for Knowledge Graph and describe how deep learning techniques can be applied to these issues: the first is representation learning, which can be used to embed entities, relations into a continuous feature space; the second is neural relation extraction, which shows how to construct Knowledge Graph by extracting knowledge from web pages and texts; the third is entity linking, which can be used to bridge knowledge with text. The deep learning techniques are used to embed entities and relations for Knowledge Graph representation, and to represent relation instances in relation extraction for Knowledge Graph construction, and to represent heterogeneous evidences for entity linking. The above techniques will provide a solid foundation for understanding, representing, constructing, and utilizing KGs in different tasks, e.g., question answering, text understanding and commonsense reasoning.

Besides benefiting KG construction, knowledge representation learning provides us an exciting approach for the application of KGs. In future, it will be important to explore how to better take KGs into consideration of deep learning models for natural language understanding and generation, and develop knowledgeable neural models for natural language processing.

References

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv:1409.0473.

Bengio, Y. (2009). Learning deep architectures for AI. Foundations and trends®. *Machine Learning*, *2*(1), 1–127.

Bollegala, D., Honma, T., Matsuo, Y., & Ishizuka, M. (2008). Mining for personal name aliases on the web. In *Proceedings of the 17th International Conference on World Wide Web* (pp. 1107–1108). New York: ACM.

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., & Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. In *Proceedings of NIPS* (pp. 2787–2795).

Chen, Z., & Ji, H. (2011). Collaborative ranking: A case study on entity linking. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (pp. 771–781). Association for Computational Linguistics.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv:1406.1078.

Dong, Z. & Dong, Q. (2003). Hownet-a hybrid language and knowledge resource. In *2003 International Conference on Natural Language Processing and Knowledge Engineering, 2003. Proceedings* (pp. 820–824). IEEE.

Francis-Landau, M., Durrett, G., & Klein, D. (2016). Capturing semantic similarity for entity linking with convolutional neural networks. In *Proceedings of NAACL-HLT* (pp. 1256–1261).

Ganea, O.-E., Ganea, M., Lucchi, A., Eickhoff, C., & Hofmann, T. (2016). Probabilistic bag-of-hyperlinks model for entity linking. In *Proceedings of the 25th International Conference on World Wide Web* (pp. 927–938). International World Wide Web Conferences Steering Committee.

Garcıa-Durán, A., Bordes, A., & Usunier, N. (2015). Composing relationships with translations. *Proceedings of EMNLP*.

Gu, K., Miller, J., & Liang, P. (2015). Traversing knowledge graphs in vector space. *Proceedings of EMNLP*.

Han, X., & Sun, L. (2011). A generative entity-mention model for linking entities with knowledge base. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies* (Vol. 1, pp. 945–954). Association for Computational Linguistics.

Han, X., & Sun, L. (2012). An entity-topic model for entity linking. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning* (pp. 105–115). Association for Computational Linguistics.

Han, X., Sun, L., & Zhao, J. (2011). Collective entity linking in web text: A graph-based method. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval* (pp. 765–774). New York: ACM.

He, S., Liu, K., Ji, G., & Zhao, J. (2015). Learning to represent knowledge graphs with Gaussian embedding. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management* (pp. 623–632). New York: ACM.

He, Z., Liu, S., Li, M., Zhou, M., Zhang, L., & Wang, H. (2013). Learning entity representation for entity disambiguation. *ACL*, *2*, 30–34.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Ji, H., Grishman, R., Dang, H. T., Griffitt, K., & Ellis, J. (2010). Overview of the TAC 2010 knowledge base population track. In *Third Text Analysis Conference (TAC 2010)* (p. 3).

Ji, G., He, S., Xu, L., Liu, K., & Zhao, J. (2015). Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of ACL* (pp. 687–696).

Ji, G., Liu, K., He, S., & Zhao, J. (2016). Knowledge graph completion with adaptive sparse transfer matrix.

Krompaß, D., Baier, S., & Tresp, V. (2015). Type-constrained representation learning in knowledge graphs. In *Proceedings of the 13th International Semantic Web Conference (ISWC)*.

Kulkarni, S., Singh, A., Ramakrishnan, G., & Chakrabarti, S. (2009). Collective annotation of Wikipedia entities in web text. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 457–466). New York: ACM.

Lin, Y., Liu, Z., & Sun, M. (2015a). Modeling relation paths for representation learning of knowledge bases. *Proceedings of EMNLP*.

Lin, Y., Liu, Z., Sun, M., Liu, Y., & Zhu, X. (2015b). Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of AAAI* (pp. 2181–2187).

Lin, Y., Shen, S., Liu, Z., Luan, H., & Sun, M. (2016). Neural relation extraction with selective attention over instances. *Proceedings of ACL*, *1*, 2124–2133.

Mihalcea, R., & Csomai, A. (2007). Wikify!: linking documents to encyclopedic knowledge. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management* (pp. 233–242). New York: ACM.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv:1301.3781.

Milne, D., & Witten, I. H. (2008). Learning to link with Wikipedia. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management* (pp. 509–518). New York: ACM.

Miwa, M., & Bansal, M. (2016). End-to-end relation extraction using LSTMs on sequences and tree structures. arXiv:1601.00770.

Nadeau, D., & Sekine, S. (2007). A survey of named entity recognition and classification. *Lingvisticae Investigationes*, *30*(1), 3–26.

Ratinov, L., Roth, D., Downey, D., & Anderson, M. (2011). Local and global algorithms for disambiguation to Wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies* (Vol. 1, pp. 1375–1384). Association for Computational Linguistics.

Silvestri, F., et al. (2009). Mining query logs: Turning search usage data into knowledge. *Foundations and Trends in Information Retrieval*, *4*(1–2), 1–174.

Socher, R., Huval, B., Manning, C. D., & Ng, A. Y. (2012). Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of EMNLP* (pp. 1201–1211).

Sun, Y., Lin, L., Tang, D., Yang, N., Ji, Z., & Wang, X. (2015). Modeling mention, context and entity with neural networks for entity disambiguation. In *IJCAI* (pp. 1333–1339).

Tai, K. S., Socher, R., & Manning, C. D. (2015). Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of ACL* (pp. 1556–1566).

Tsai, C.-T., & Roth, D. (2016). Cross-lingual wikification using multilingual embeddings. In *Proceedings of NAACL-HLT* (pp. 589–598).

Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning* (pp. 1096–1103). New York: ACM.

Wang, Z., Zhang, J., Feng, J., & Chen, Z. (2014a). Knowledge graph and text jointly embedding. In *Proceedings of EMNLP* (pp. 1591–1601).

Wang, Z., Zhang, J., Feng, J., & Chen, Z. (2014b). Knowledge graph embedding by translating on hyperplanes. In *Proceedings of AAAI* (pp. 1112–1119).

Xiao, H., Huang, M., & Zhu, X. (2016). From one point to a manifold: Orbit models for knowledge graph embedding. In *Proceedings of IJCAI* (pp. 1315–1321).

Xiao, H., Huang, M., Hao, Y., & Zhu, X. (2015). Transg: A generative mixture model for knowledge graph embedding. arXiv:1509.05488.

Xie, R., Liu, Z., & Sun, M. (2016c). Representation learning of knowledge graphs with hierarchical. In *Proceedings of IJCAI*.

Xie, R., Liu, Z., Chua, T.-s., Luan, H., & Sun, M. (2016a). Image-embodied knowledge representation learning. arXiv:1609.07028.

Xie, R., Liu, Z., Jia, J., Luan, H., & Sun, M. (2016b). Representation learning of knowledge graphs with entity descriptions. In *Proceedings of AAAI*.

Xu, K., Feng, Y., Huang, S., & Zhao, D. (2015). Semantic relation classification via convolutional neural networks with simple negative sampling. arXiv:1506.07650.

Zeng, D., Liu, K., Chen, Y., & Zhao, J. (2015). Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of EMNLP*.

Zeng, D., Liu, K., Lai, S., Zhou, G., & Zhao, J. (2014). Relation classification via convolutional deep neural network. In *Proceedings of COLING* (pp. 2335–2344).

Zhang, D., & Wang, D. (2015). Relation classification via recurrent neural network. arXiv:1508.01006.

Zhong, H., Zhang, J., Wang, Z., Wan, H., & Chen, Z. (2015). Aligning knowledge and text embeddings by entity descriptions. In *Proceedings of EMNLP* (pp. 267–272).