# Chapter 6
# Reasoning in Knowledge Graphs: An Embeddings Spotlight

Luigi Bellomarini[1], Emanuel Sallinger[2,3], and Sahar Vahdati[3]

[1] Banca d'Italia, Italy
[2] TU Wien, Austria
[3] University of Oxford, United Kingdom

## 1 Introduction

**Abstract.** In this chapter we introduce the aspect of reasoning in Knowledge Graphs. As in Chapter 2, we will give a broad overview focusing on the multitude of reasoning techniques: spanning logic-based reasoning, embedding-based reasoning, neural network-based reasoning, etc. In particular, we will discuss three dimensions of reasoning in Knowledge Graphs. Complementing these dimensions, we will structure our exploration based on a pragmatic view of reasoning tasks and families of reasoning tasks: reasoning for knowledge integration, knowledge discovery and application services.

The notion of intelligence is closely intertwined with the ability to reason. In turn, this ability to reason plays a central role in AI algorithms. This is the case not only for the AI of today but for any form of knowledge representation, understanding and discovery, as stated by Leibniz in 1677: "It is obvious that if we could find characters or signs suited for expressing all our thoughts as clearly and as exactly as arithmetic expresses numbers or geometry expresses lines, we could do in all matters insofar as they are subject to reasoning all that we can do in arithmetic and geometry. For all investigations which depend on reasoning would be carried out by transposing these characters and by a species of calculus."[277].

Research in reasoning was carried out by mathematicians and logicians, and naturally adopted and also carried out by computer scientists later on. Concrete references of having knowledgeable machines date back to at least the 1940s – V. Bush talked about a machine able to think like a human in his influential essay in 1945 "As We May Think" [64]. Later in 1950, with Alan Turing's seminal work [430], the idea behind Artificial Intelligence and impressing thinking power to machines began with mathematically employed reasoning. The developments of symbolic reasoning continued towards providing mathematical semantics of logic programming languages [439, 301] and new forms of efficient reasoning foundations [72, 233]. Reasoning about facts of belief networks, as in today's Knowledge Graphs, is addressed in [347].

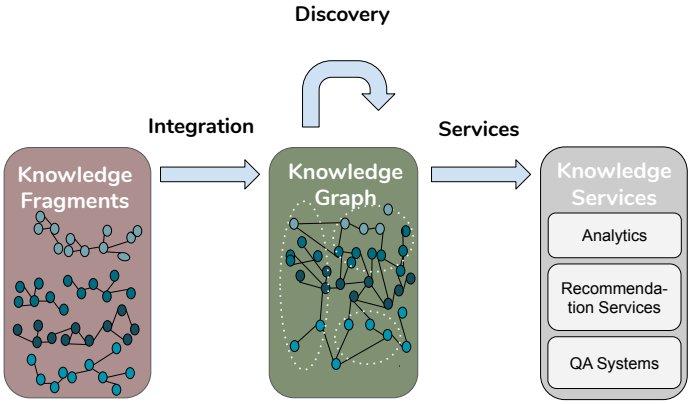


Fig. 1: A simplified life-cycle of Knowledge Graphs

However, at the scale at which they were envisioned, all of these approaches were simply not possible in practice without large-scale data management, processing, inference and retrieval. The last decade witnessed a technology boost for AI-driven technologies with the emergence of Big Data. This has created an incredible number of industrial-scale applications of Machine Learning approaches over data represented and managed in Knowledge Graphs. The technology behind KGs created a practical platform for the envisioned AI machines.

**Perspectives**. In Chapter 2, we introduced the *layered perspective* of Knowledge Graphs, and noted that the aspect of *reasoning* will be considered particularly in this chapter. It is clear that the requirements on reasoning are different between the three layers introduced in Chapter 2:

- At the bottom-most layer (**representation**), reasoning is an important design consideration to achieve a good balance between expressive power and computational complexity.
- At the middle layer (**management**), similar to a relational database management system, providing a general-purpose reasoning (or in a RDBMS: querying) service is of utmost importance.
- At the top layer (**application**), the specific reasoning service required or exposed by the application becomes the focus.

Given both the history of use of reasoning methods in computer science, as well as their concrete use in the construction and use of Knowledge Graphs, it would be tempting to divide them according to their use in the life-cycle of KGs. This is illustrated in Figure 1 where we see knowledge fragments being integrated into a Knowledge Graph, this KG being enriched using discovery, and finally services provided based on the Knowledge Graph:

- **Reasoning for Knowledge Integration**: where the focus is to use reasoning in order to deal with knowledge acquisition and integration from heterogeneous, interconnected and distributed data.
- **Reasoning for Knowledge Discovery**: where the focus is to use reasoning in order to identify new – and possible hidden – knowledge based on existing knowledge.
- **Reasoning for Application Services**: where the focus is to employ reasoning techniques to directly provide services at the application level of the Knowledge Graph.

The position that we will take in this chapter is that while these three phases of the life-cycle are clearly important, and many of the available reasoning techniques fall into one category or the other, many others as we shall see permeate these life-cycle phases. We thus refer to them rather as *dimensions*.

This chapter shall not be a survey of reasoning techniques, but for each of the three dimensions it shall give one or two prominent examples to give the reader an impression on the breadth and variation between reasoning techniques on Knowledge Graphs.

**Organization**. In Section 2, we will consider the dimension of integration; in Section 3, we consider the dimension of discovery; and in Section 4, we consider the dimension of application services. We will conclude with a summary.

## 2 Reasoning for Knowledge Integration

In recent years, a huge number of Knowledge Graphs has been built both in academia and industry. Knowledge Graph creation follows a set of steps for data acquisition and integration from heterogeneous resources. It requires a comprehensive domain conceptualization and a proper data representation model. In many cases, data transformation from the already existing formats formed the Knowledge Graph for many individual or enterprise agents. With post-processing stages, such Knowledge Graphs have been made usable by other approaches for further investigations.

Yet, considering the potential amount of information that could be mapped into such Knowledge Graphs from the real world, they are greatly incomplete. A number of manual and automated data curation, harvesting and integration techniques are being developed for data completion tasks already from decades ago. However, considering the characteristics of Knowledge Graphs, they became ideal for applying machine learning approaches to Knowledge Graph completion. Thus, KG completion tasks gain a new dimension meaning the coverage increase of knowledge. Therefore, new communities of research have been merged or revived such as knowledge embedding. Application of such models have been investigated with the objective of providing services for link predictions, resource classification and recommendation services.

Aforementioned representations are attempts to create a real world model where a lack of full coverage and information correctness problems will always

be present. Thus, proposing embedding models for Knowledge Graphs gained a lot of attention by giant companies and received great hype in research in recent years. Such models are probabilistic-based approaches to predict missing relations in a graph. Although there have already been proposals of using ML and such probabilistic link prediction models on top of data modeled in triples from the early 2000s, the application of such models has been practiced with the emergence of KGs. Three conflicting dimensions of challenges in the construction of such a Knowledge Graph have been mentioned [145] namely *freshness*, *coverage* and *correctness*.

### 2.1 Schema/Ontology Matching

Ontology matching in the meaning of finding semantic relationships between entities of one or several Knowledge Graphs plays an important role in KG integration and construction. Due to the heterogeneity of KGs, the process of KG integration and mapping ontologies end with high complexities. Therefore scalability is one of the main focal points in this regard. The approaches for providing light weighted ontology matching tools includes ontology partitioning [129], use of data and ontology structure [229, 381]. There are two main categories of approaches: logic-based and graph-based [3]. In the early years of the Semantic Web community [166, 167], some logic-based reasoning approaches, which are used to partition the relationships of an ontology, have been discussed.

Another set of approaches are ontology-based data access (OBDA) [354] approaches, which are well-known where ontologies are used to encode the domain knowledge, which enables new fact deduction. In [57], a datalog-based approach is proposed for KG completion tasks. A datalog is an ontology-based approach that is applied in question answering [287].

The proposed approach is a partitioning model that incorporates the ontology graph and the distribution of extractions. In a related work, reasoning by using ontology-based approaches is used to query probabilistic knowledge bases [58, 73]. The application of such ontology-based reasoning in relation to other inference tasks such as maximum a posteriori (MAP) computations and most probable explanations (MPE) corresponds to identifying tuples that contribute the most to the satisfaction of an observed query. The concept of common sense is introduced as a type of knowledge in [58] with regard to closed world or open world assumptions. With a closed world assumption, question-answering systems that are built on top of knowledge bases fail to answer anything that requires intuitive or deductive reasoning.

A logic-based scalable ontology matching system is introduced in [227] named LogMap. The ontology obtained by integrating LogMap's output mappings with the input ontologies is consistent. Although it belongs to the period before KGs were introduced, its capability in terms of dealing with semantically rich ontologies makes it considerable for application in KGs as well. Logical reasoning is also used in other works over the union of the source ontologies, e.g. in the medical domain [228].

In general, Knowledge Graph identification (KGI) is used as a reasoning technique in Knowledge Graph construction. For example, [360] deals with challenges in automation of KG creation from noisy extractions. In order to handle the scaling problems, partitioning extractions is an approach that allows parallel reasoning in carving valid KG from a collection of noisy information. KGIs uses logical constraints and entity resolution and the results can be used in classification and link prediction tasks. In a series of works [360, 359, 357], probabilistic soft logic (PSL) is used for running reasoning jointly with extraction of knowledge from a noisy collection of information. The proposed solution is based on an ontology-aware technique that uses universally quantified logical rules. It performs efficient reasoning on KGs with rich representation of ontologies and statements in Web Ontology Language (OWL). In the reasoning process, frequent patterns, constraints or paths are used to infer new knowledge.

The rules are defined to relate the uncertain information discovered in the extraction process. The extracted triples are labeled to be a candidate relation or a candidate label and a value is assigned which shows the probable truth of the triple. The model combines the weights from several sources and retrieves a list of classifications or predicted links. Ontological information such as domain and range constraints are used to further enrich the reasoning. The joint reasoning means that logical rules as well as entity resolution are used in parallel such that a) logical rules relate the ontological knowledge about the predicates of the constructed Knowledge Graph and b) entity resolution are injected in prediction.

F-OWL is another ontology matching the engine proposed in [487], and was originally designed for knowledge bases. It is a rule-based reasoning engine which also considers entity resolution for extracting hidden knowledge. Pellet, an open source OWL-DL reasoner [401], employs an incremental reasoning mechanism. Thus semantic expressively of such formalism for representing and querying probabilistic knowledge has gained significant importance in recent years. Another application of KG integration is given in [116], which explains a chain of processes in which domain knowledge about Chinese Intangible cultural heritage (ICH) was extracted from textual sources using Natural Language Processing (NLP) technology. The extracted knowledge is shaped as a knowledge base using on domain ontology and instances.

### 2.2 Entity Resolution

One of the techniques required for combining multiple Knowledge Graphs is using entity resolution. In some cases, this task turns to a pair-wise matching task between the target KGs for integration. This can bring a set of challenges caused by different ontologies used by KGs and additional complexity. In [358], a unified model for entity resolution is provided for KG integration tasks.

Some of these reasoning techniques are used for Knowledge Graph refinement after data integration processes. Several researchers of the KG domain (e.g., Paulheim, Dong) have been using the KG "Refinement" notion to define a range of technology application with the purpose of KG enrichment including completion and error detection. In some other views, refinement has seen

improvements in KGs by considering that ontology learning mainly deals with learning a concept-level description of a domain.

### 2.3 Data Exchange and Integration

While the focus of this chapter shall be on embedding-based reasoning, we do want to at least give a glimpse at the huge body of logic-based reasoning methods and techniques developed in the database and artificial intelligence area over basically the last decades, including large research organizations such as IBM research and others spearheading these kinds of developments.

Logical rules that play the role of knowledge in a Knowledge Graph, and are thus reasoned upon have been historically often called *schema mappings*. There exist countless papers in this area [18, 51, 127, 250, 432], a survey on reasoning about schema mappings can be found at [380]. Key formalisms in these area are *tuple-generating dependencies* (tgds), i.e., logical formulas of the form

$$\varphi(\bar{x}) \rightarrow \exists \bar{y}\, \psi(\bar{x}, \bar{y})$$

where $\varphi$ and $\psi$ are conjunctions of relational atoms and all free variables are universally quantified (which we will assume for all formulas presented in what follows by some abuse of notation), and *equality-generating dependencies* (egds), i.e., logical formulas of the form

$$\varphi(\bar{x}) \rightarrow x_i = x_j$$

These together can express a large amount of knowledge typically expressed in database constraints, and thus usable for data exchange and data integration, or simply as knowledge in Knowledge Graphs.

Research foci include the higher expressive power needed for particular reasoning tasks, including

- *second-order (SO) tgds* [128, 132, 133, 161, 163] for expressing ontological reasoning and composition, i.e., logical formulas that, in simplified form have the structure

  $$\exists \bar{f}((\varphi_1 \rightarrow \psi_1) \wedge \ldots \wedge (\varphi_n \rightarrow \psi_n))$$

  where $\bar{f}$ are function symbols.
- *nested tgds* [141, 251] for expressing reasoning on tree-like data, i.e., normal tgds of the form

  $$\chi = \varphi(\bar{x}) \rightarrow \exists \bar{y}\, \psi(\bar{x}, \bar{y})$$

  but with the extension that each conjunct of $\psi$ may in addition to a relational atom also be a formula of the form $\chi$ again, i.e., allow nesting.

A particularly important restriction is the study of reasoning with conjunctive queries (CQs), i.e., in the form of logical rules

$$\exists \bar{x}\, \varphi(\bar{x}, \bar{y}) \rightarrow Ans(\bar{y})$$

where *Ans* is an arbitrary predicate name representing the answer of a query. These CQs are at the core of almost all practical data processing systems, including of course databases and Knowledge Graph management systems that allow reasoning or querying of almost any level. Under the name of "projective views", reasoning on them has been studied intensively, for pointers see e.g. [173], but there are countless papers studying this formalism central to KGs.

While we will avoid making this section a full-blown survey on reasoning in data exchange and integration, we do want to give a (biased) selection of, in our opinion, particularly interesting reasoning problems in this area:

- *limits* [252]: like limits in the mathematical, it is particularly relevant for *approximating* data exchange and integration scenarios to also reason about limits in this context. Similarly to limits, other operators such as union and intersection are important [20, 349].
- *equivalence* [353]: equivalence is a fundamental reasoning problem for all other services building upon it, such as optimization, approximation, etc.
- *inconsistency* [19, 22, 351]: reasoning in an inconsistent state of data or knowledge is the standard case for Knowledge Graphs, and needs delicate handling.
- *representability* [21]: how can knowledge be represented in different parts of a Knowledge Graph?

Many other topics could have been mentioned here – and many more references given – as this is a particularly rich area of reasoning on this important sub-area of Knowledge Graphs. Bridging the gap towards our main focus in this chapter, embedding-based reasoning, we conclude by mentioning that substantial parts of the logic-based reasoning formalisms presented in this section can be *injected* into embedding-based reasoning methods to make them perform far better than they could have if no such knowledge were present in the Knowledge Graph.

## 3 Reasoning for Knowledge Discovery

In this section, we structure reasoning approaches for task-based AI challenges. There is a long list of possible approaches that could go in this category; however, we will focus on embedding-based reasoning for link prediction. Examples of other approaches could be Statistical Relational Learning (SLRs) which are well covered in several review articles [328, 483], Markov Logic Networks (MLN) [249, 371], and Probabilistic Graphical Models [8, 253, 315].

### 3.1 Link Prediction

The power of specific knowledge representation in Knowledge Graphs facilitates information systems in dealing with challenges of Big Data and supports solving challenges of data heterogeneity. However, KGs suffer from incompleteness,

inaccuracy and low data quality in terms of correctness [17, 324]. This highly affects the performance of AL-based approaches, which are used on top of KGs in order to provide effective services. Therefore, graph completing methods gained a lot of interest to be applied on KGs. One of the most popular methods is Knowledge Graph Embedding models, which obtain the vector representation for entities and/or relations to be used in downstream tasks such as Knowledge Graph Completion tasks. KGEs are a type of deductive reasoning in the vector space through discovery of new links.

For a Knowledge Graph with a set of triples in the form of *(h,r,t)* representing (head, relation, tail), KG embeddings aim at mapping entities and relations into a low-dimensional vector space. Then, the KGE model defines a score and loss functions to further optimize the vectors through a specific embedding representation. The embedding of entities and relations is generally learned over existing positive samples inside the KGs. A set of negative samples are also usually injected into the model in order to optimize the learning phase and help the KGE model gain strength. In these ways, the score function is trained over both the positive and negative samples and assigns a high score for positive samples and a low score to negative samples. Each embedding model also has a loss function that optimizes the scoring. Here we will look into the existing embedding models from the lens of their reasoning power in knowledge discovery. Knowledge Graph embedding models can be roughly divided into three main categories:

- **Translational and Rotational Based Models** A large number of KGE models are designed using mathematical transnational (plus) or rotational (Hadamard product). The score and loss function of these models optimize the vectors in a way that their plausibility is measured by the distance or degree of the entities with regard to the relation.
- **Semantic Matching Models** Some of the embedding models are designed based on element-wise multiplication. In this case, the similarity of the vectors is evaluated to define the plausibility of the entities an relations.
- **Neural Network-Based Models** A third category of the KGE models are the ones designed on top of neural networks. These models have two learning phases: one for calculating and creating the vectors and the second for evaluating the plausibility in a layer-based learning approach, which comes from NN.

**Translational and Rotational Models**. In this type of model, the plausibility of a triple is computed based on distance function (e.g. based on the Euclidean distance) [455]. In the following, we describe KGE models that are relevant in the context of this work; however, many others have been proposed.

**TransE** [56] is one of the early KGE models that is the base for several other families of models where the score function takes a relation $r$ as the translation from the head entity $h$ to the tail entity $t$:

$$h + r \approx t \tag{1}$$

To measure the plausibility of a triple, the following scoring function is defined:

$$f_r(h,t) = -\|h + r - t\| \tag{2}$$

The TransE model is extremely simple and computationally efficient. Therefore, it is one of the most common embedding models used on large-scale KGs with the purpose of reasoning for knowledge discovery. However, TransE is limited in modeling 1-N, N-1 and N-M relations. For this reason, several extensions have been proposed [455]. Due to this fact, encoding relations with reflexive and symmetric patterns becomes impossible, which is an important aspect in the inference of new knowledge. Therefore, several new models have tried to solve this problem, which will be discussed in the remainder of this section.

**TransH** [459] is en extension of TransE, which addresses the limitations of TransE in modeling N-M relations. It uses relation-specific entity representation to enable encoding of such relational patterns. This model uses an additional hyperplane to represent relations. Then, the translation from the head to the tail entity is performed in that relation-specific hyperplane. This method is called projecting head and tail entities into the relation-specific hyperplane. The formulation of this method is as follows:

$$h_\perp = h - w_r^\top h w_r \tag{3}$$

$$t_\perp = t - w_r^\top t w_r \tag{4}$$

where $w_r$ is the normal vector of the hyperplane. The plausibility of the triple *(h,r,t)* is computed:

$$f_r(h,t) = -\|h_\perp + d_r - t_\perp\|_2^2 \tag{5}$$

where $d_r$ is the relation-specific translation vector.

**TransR** is another KGE model that followed the basics from TranE as an extension of TransH with a difference that it encodes entities and relations in different vector spaces. This is a relation-specific solution in contrast to the hyperplanes of TrasnH where the translation happens in the specific space of each relation. Relations are in matrix representation of $M_r$ which takes entities projected into the relational specific space:

$$h_r = hM_r \tag{6}$$

$$t_r = tM_r \tag{7}$$

Based on this representation, the score function is designed as following:

$$f_r(h,t) = -\|h_r + r - t_r\|_2^2 \tag{8}$$

This model is capable of handling complex relations as it uses different spaces; however its computation is highly costly due to the high number of required parameters.

**TransD** [224] is an attempt to improve TransR by reducing the number of required parameters by removing the need for matrix vector multiplications. The core of this model is to use two vectors for representation of entities and relations. Assuming that $h, r, t$ encode the semantics, and $h_p, r_p, t_p$ constructs projection, the projection of entities in relation-specific spaces is defined as follows:

$$M_{rh} = r_p h_p^T + I^{m \times n} \tag{9}$$

$$M_{rt} = r_p t_p^T + I^{m \times n}, \tag{10}$$

In this definition, $I$ is a matrix where the values of the diagonal elements are 1 and 0 elsewhere. The head and tail entities are computed as:

$$h_\perp = M_{rh} h \tag{11}$$

$$t_\perp = M_{rt} t \tag{12}$$

The score of the triple (h,r,t) is then computed based on these projections:

$$f_r(h, t) = -\|h_\perp + r - t_\perp\|_2^2 \tag{13}$$

**RotatE** [415] is one of the early models which uses rotation than translation. The model is mainly designed with the objective of reasoning relational patterns, which was not mainly addressed by other translational models. RotatE is designed to reason new knowledge based on the Euler formula $e^{i\theta} = \cos(\theta) + i \sin(\theta)$. Based on its score function, for every correct triple $(h, r, t)$ there should be the relation of $h_j r_j = t_j$ which holds $\forall j \in \{0, \ldots, d\}$. $h_j, r_j, t_j$ are the $j$-th elements of the embedding vectors of $\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{C}^d$. Since it deals with complex space, $r_i$ is set to 1 i.e. $|r_j| = \sqrt{Re(r_j)^2 + Im(r_j)^2} = 1$. The model performs a rotation of the $j$-th element $h_j$ of the head vector $\mathbf{h}$ by the $j$-th element $r_j = e^{i\theta_{r_j}}$ of a relation vector $\mathbf{r}$ to get the $j$-th element $t_j$ of the tail vector $\mathbf{t}$, where $\theta_{r_j}$ is the phase of the relation $r$. Therefore, the score function of RotatE is designed as a rotation using $\circ$ which is a Hadmard product of two vectors:

$$f_{h,t}^r = \|\mathbf{h} \circ \mathbf{r} - \mathbf{t}\|, \tag{14}$$

In this way, the RotatE model becomes capable of encoding symmetric, inverse, and composition relation patterns. Due to this capability, its performance is high and due to the high quality of the newly discovered links in the reasoning process, it outperforms all the previous models.

**Semantic Matching Models**. As discussed before, the second category of embedding models in reasoning over KGs determines the plausibility of a triple by comparing the similarity of the latent features of the entities and relations. A number of KGE models fall into this category; we will discuss a few of the best performing ones.

**RESCAL** [325] is an embedding-based reasoning model that represents each entity as a vector and each relation as a matrix, $M_r$ to capture the latent semantics. The score of the triples is measured by the following formulation:

$$f_r(h,t) = h^T M_r t \tag{15}$$

where $M_r$ is a matrix associated with relations, which encodes pairwise interactions between the features of the head and tail entities.

**DistMult** is a model that focuses on capturing the relational semantics and the composition of relations as characterized by matrix multiplication [472]. This model considers learning representations of entities and relations within the underlying KG. DistMult [472] simplifies RESCAL by allowing only diagonal matrices as $diag(\boldsymbol{r})$. The score function of this model is designed in a way that triples are ranked through pair-wise interactions of the latent features:

$$f_r(h,t) = h^T diag(r) t \tag{16}$$

where $r \in R^d$ and $M_r = diag(r)$. The restriction to diagonal matrices makes DistMult more computationally efficient than RESCAL but less expressive.

**ComplEx** ComplEx [428] is an extension of DistMult into the complex space. Considering the scoring function of DistMult, it can be observed that it has a limitation in representing anti-symmetric relations since $h^T diag(r)t$ is equivalent to $t^T diag(r)h$. Equation 16 can be written in terms of the Hadamard product of $h, r, t$: $< h, r, t > = \sum_{i=1}^{d} h_i * r_i * t_i$, where $h, r, t \in R^d$. The scoring function of ComplEx uses the Hadamard product in the complex space, i.e. $h, r, t \in C^d$:

$$f_r(h,t) = \Re(\sum_{i=1}^{d} h_i * r_i * \overline{t_i}) \tag{17}$$

where $\Re(x)$ represents the real part of a complex number and $\overline{x}$ its conjugate. It is straightforward to show that $f_r(h,t) \neq f_r(t,h)$, i.e. ComplEx is capable of modeling anti-symmetric relations.

**Neural Network-Based Models**. As the last category of the embedding models that we will discuss here, we consider the ones which are built on top of Neural Networks. Such models inherit a second layer from NNs for the learning phase. This category is also known as Neural Link Predictors, which is in the downstream task level, the ultimate objective of such models. Such models contain a multi-layered learning approach with two main components: namely, encoding of the vectors and scoring of the vectors.

**ConvE** [106] is a multi-layer embedding model designed on top of the neural networks.

$$f(h,t) = g(\mathrm{Vec}(g([\bar{\mathbf{h}}; \bar{\mathbf{r}}] * \omega))\, W)\, \mathbf{t} \tag{18}$$

**Neural Tensor Network (NTN)** [406] is one of the earlier methods which includes textual information in the embedding. It learns the word vectors from

a corpus and initializes each entity by the average of vectors of words associated with the entity.

$$\vec{w}_r^T \tanh(\vec{h}^T W_r \vec{t} + W_r^{(1)} \vec{h} + W_r^{(2)} \vec{t} + \vec{b}_r) \tag{19}$$

**LogicENN** [321] is an NN-based model which performs reasoning on top of a KG through jointly learning embeddings of entities $(\mathbf{h}, \mathbf{t})$ and relations $(\beta_{\mathbf{r}})$ of the KG and the weights/biases $(\mathbf{w}/b)$ of the NN. Given a triple of $(h, r, t)$, the network passes the entity vectors $(\mathbf{h}, \mathbf{t})$ through a universally shared hidden layer with $L$ nodes to obtain the joint feature mapping of the entities $(h, t)$ i.e. $\Phi_{h,t}^T = [\phi_{h,t}(\mathbf{w}_1, b_1), \ldots, \phi_{h,t}(\mathbf{w}_L, b_L)] = [\phi(\langle \mathbf{w}_1, [\mathbf{h}, \mathbf{t}] + b_1 \rangle), \ldots, \phi(\langle \mathbf{w}_L, [\mathbf{h}, \mathbf{t}] + b_L \rangle)]$. The network considers the weights of the output nodes (i.e. $\beta_{\mathbf{r}}$) as the embedding of relation $r$. The score of the triple $(h, r, t)$ is computed by the inner product of $\Phi_{h,t}$ and $\beta_{\mathbf{r}}$ as follows

$$\begin{aligned} f(h, r, t) = \sum_{i=1}^{L} \phi(\langle \mathbf{w}_i, [\mathbf{h}, \mathbf{t}] + b_i \rangle) \beta_i^r = \sum_{i=1}^{L} \phi_{h,t}(\mathbf{w}_i, b_i) \beta_i^r \\ = \Phi_{h,t}^T \beta^r. \end{aligned} \tag{20}$$

Considering the formulation of the score function, the algebraic formulae (algebraic constraints) corresponding to each of the logical rules – namely symmetric, inverse, transitive, negation, implication, equivalence etc – are derived. The formulae are then used as penalty terms to be added to the loss function for optimization. This enables the injection of rules into the learning process of the network. Consequently, the performance of the model is improved.

Overall, the network has the following advantages:

- The model is proven to be capable of expressing any ground truth of a KG with $n$ facts.
- The network separates the spaces of entities $(\phi_{h,t})$ and relation $\beta_{\mathbf{r}}$. Therefore, the score-based algebraic constraints corresponding to the symmetric, inverse, implication and equivalence rules do not need the grounding of entities. This feature enables the model to better inject rules with a lower computational cost due to lifted groundings.
- The model has been shown to obtain state-of-the-art performance on several standard datasets.

**Summary**. So far we have given a detailed description of some highlighted methods in embedding-based reasoning methods. More information can be found in [324, 456]. Despite the fact that most of embeddings only consider the relation and entities of a KG, there are several types of complementary knowledge (e.g., text, logical rules, ontology, complementary KG) from which embedding models can be improved. In [326], ontological knowledge is introduced as complementary knowledge, which can be used in the factorization process of embedding models. In some of the more focused work, ontological knowledge such as entity types

is used as constraints [471, 263, 457, 200] which improves the performance of the embedding models. In recent years, logic-based reasoning and embedding-based reasoning have come together and attracted a great deal of academic attention. Some initial work is done using logical rules as a post-processing task after embedding [457, 462]. [373] optimizes the embeddings using first order logical rules to obtain entity pairs and relations. [201] provides a general framework to transfer information in logical rules to the weights of different types of neural networks.

## 4 Reasoning for Application Services

The ultimate goal of the aforementioned approaches is to provide better knowledge aware services such as smart analytics and recommendation and prediction services as well as to facilitate query answering. In many knowledge management tasks, learning and reasoning is an important component towards providing such services. There are also hybrid systems which integrate many such models consuming different learning representation and learning methods. Such methods are usually defined as high-level tasks where the purpose is to gain a certain practical step in KGs where it is ready for low-level tasks. This section includes some AI-driven applications with an underlying knowledge-aware learning and reasoning engine.

### 4.1 Recommendation Systems

In many of the high-level tasks related to Knowledge Graphs, learning and reasoning methods are considered to be well-suited to providing recommendation services. Recommendation services are typical applications of reasoning for knowledge discovery and link prediction approaches. Logic-based reasoning provides explainable recommendations while embedding-based reasoning mostly explores the interlinks within a knowledge graph. The learning phase in both of these approaches is mostly about analysis of the connectivity between entities and relations in order to discover possible news paths. This can be facilitated with rich and complementary information. These approaches reveal the semantics of entities and relations and facilitate recommendation services to comprehend ultimate user interests.

In the domain application level, such approaches can be applied for any graph-based scenario. For example, KGs of social networks [454] are one of the most interesting application domains on which learning frameworks are applied. Item recommendation in online shopping is a typical application for link prediction. Such problems are usually formulated as ML-based problems in KGs and employ link prediction approaches. Another typical example is link prediction between co-authors in scholarly Knowledge Graphs. The plausibility of such recommendations is prediction-based for the future and might not happen. Adding a temporal feature for such recommendations by making Knowledge Graphs time-aware makes such applications more interesting.

### 4.2 Question Answering

A number of reasoning-based applications for which intelligent systems are built goes under the umbrella of question answering systems (QA). In addition to normal search engines and query-based systems, into this category falls conversational AI systems, speech assistants, and chat-bots. Example of such systems are Apple's Siri, Microsoft's Cortana, and Amazon's Alexa, for which the source of knowledge is an underlying KG. Despite the huge success in building such systems, the possible incorrect answers as well as their limits in retrieving a certain level of knowledge queries is not avoidable. There are multiple reasons for this, such as KG incompleteness or other quality issues on the data side, which cause minimal semantic understanding. However, for the complete part of the data in practice, any simple question has the potential to require complex queries and thus complex reasoning over multiple computational steps [275]. Therefore, all of these systems are facilitated with reasoning and inference techniques in order to retrieve hidden information.

In recent years, one of the hyped applications of reasoning for question answering is on Knowledge Graphs with diverse modality of data. This is because, by nature, Knowledge Graphs contain different types of information ranging from images, text, numerical data or even videos and many more. The main challenge is that, on the application side, most of the learning approaches are mainly considered with one modal. While there has been a lot of progress from computer vision communities in audio and video processing, such multidisciplinary research is still at an early stage. Such KGs are known as *Multimodal Knowledge Graphs (MKGs)* and have fundamental differences with other visual-relational resources. There are recent works on construction of *Multi-Modal Knowledge Graphs* and application of ML-related models on top of such KGs. Visual QA systems are designated specifically for MKGs [65].

Due to the explainability power of rule-based reasoning techniques, they are an important part of QA systems as well. In the case of complex questions with a need for multiple steps, it is easier to provide explainable and certain statements. Multi-hop reasoning is a solution for such cases, which is elevated by end-to-end differentiable (deep learning) models [461, 107].

## 5 Challenges and Opportunities

In this chapter, we considered reasoning in Knowledge Graphs in multiple dimensions: namely that of integration, discovery and application. For each of these, we picked some techniques that showcase some of the diversity of reasoning techniques encountered in Knowledge Graphs. As a grand challenge, we see the integration of multiple reasoning techniques, such as logic-based and embedding-based reasoning techniques, and similarly neural network-based reasoning and other reasoning techniques. Clearly, also each individual reasoning problem that we introduced in this chapter would allow for challenges and opportunities to be listed, but that would go beyond the scope of this chapter.

**Acknowledgements**. E. Sallinger acknowledges the support of the Vienna Science and Technology (WWTF) grant VRG18-013 and the EPSRC programme grant EP/M025268/1.