



**HAL**  
open science

## Combining Embeddings and Rules for Fact Prediction

Armand Boschin, Nitisha Jain, Gurami Keretchashvili, Fabian M. Suchanek

► **To cite this version:**

Armand Boschin, Nitisha Jain, Gurami Keretchashvili, Fabian M. Suchanek. Combining Embeddings and Rules for Fact Prediction. International Research School in Artificial Intelligence in Bergen, 2022, Bergen (NO), Norway. 10.4230/OASIs.AIB.2022.4 . hal-04462010

**HAL Id: hal-04462010**

**<https://telecom-paris.hal.science/hal-04462010v1>**

Submitted on 16 Feb 2024

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Combining Embeddings and Rules for Fact Prediction

Armand Boschin ✉

Télécom Paris

Institut Polytechnique de Paris, France

Nitisha Jain ✉

Hasso Plattner Institute

University of Potsdam, Germany

Gurami Keretchashvili ✉

Télécom Paris

Institut Polytechnique de Paris, France

Fabian Suchanek ✉🏠

Télécom Paris

Institut Polytechnique de Paris, France

---

## Abstract

Knowledge bases are typically incomplete, meaning that they are missing information that we would expect to be there. Recent years have seen two main approaches to guess missing facts: Rule Mining and Knowledge Graph Embeddings. The first approach is symbolic, and finds rules such as “If two people are married, they most likely live in the same city”. These rules can then be used to predict missing statements. Knowledge Graph Embeddings, on the other hand, are trained to predict missing facts for a knowledge base by mapping entities to a vector space. Each of these approaches has their strengths and weaknesses, and this article provides a survey of neuro-symbolic works that combine embeddings and rule mining approaches for fact prediction.

**2012 ACM Subject Classification** Information systems → Information systems applications

**Keywords and phrases** Rule Mining, Embeddings, Knowledge Bases, Deep Learning

**Digital Object Identifier** 10.4230/OASICS.AIB.2022.4

**Category** Invited Paper

**Acknowledgements** This work was partially funded by ANR-20-CHIA-0012-01 (“NoRDF”).

## 1 Introduction

A knowledge base (KB) is a computer-processable collection of knowledge about the world. KBs typically contain real-world entities (such as organizations, people, movies, or locations) and their relationships (who was born where, which movie plays where, etc.). Thousands of such KBs are publicly available, including, e.g., Wikidata [60], DBpedia [4], and YAGO [53]. These KBs contain millions of entities and relationships between them, saying, e.g., who was born in which city, which actor acted in which movie, or which city is located in which country. Such KBs are used for question answering, Web search, text understanding, personal assistants, and other AI applications [66].

KBs are usually never complete; there are always facts that are missing from the KB. This is due to the way in which KBs are constructed: Some of them are constructed automatically by extracting facts from Web sources. Such an extraction may fail to extract all information, and the underlying sources can be incomplete themselves. Other KBs are fed by a community, and may be incomplete simply because not all facts have yet been added. *Fact prediction* is the task of predicting facts that are true in the real world, but missing in the KB. Although



© Armand Boschin and Nitisha Jain and Gurami Keretchashvili and Fabian Suchanek; licensed under Creative Commons License CC-BY 4.0

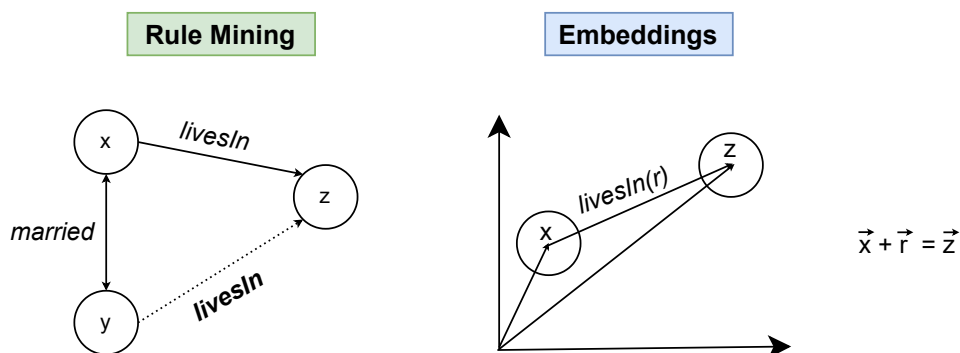
International Research School in Artificial Intelligence in Bergen (AIB 2022).

Editors: Camille Bourgaux, Ana Ozaki, and Rafael Peñaloza; Article No. 4; pp. 4:1–4:30

OpenAccess Series in Informatics



OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



■ **Figure 1** Rule Mining and Embeddings

44 this may never make the KB complete, it will at least add facts that were missing. There are  
 45 two major approaches to this end: Rule Mining and Knowledge Graph Embeddings. *Rule*  
 46 *mining* is a symbolic approach. It finds rules such as the following in a KB:

$$47 \quad \text{married}(x, y) \wedge \text{livesIn}(x, z) \Rightarrow \text{livesIn}(y, z)$$

48 This rule means that if some person  $x$  is married to some person  $y$ , and  $x$  lives in a city  $z$ ,  
 49 then  $y$  also lives in that city. Such rules are usually not true in all instances, and typically  
 50 come with a confidence score. Modern systems [30, 34, 40] can find such rules automatically  
 51 on KBs of millions of entities. These rules can then be used to predict missing facts: If we  
 52 know that some person lives in some city, but we do not know the place of residence of their  
 53 spouse, we can use the rule to predict that, with high likelihood, the spouse lives in the same  
 54 city.

55 The other methods to predict missing facts are *embedding-based methods*. These methods  
 56 are a gift of the renaissance of neural networks in the 2010's. They project the entities  
 57 and facts of a KB into a vector space. In its simplest variant, an entity  $x$  is mapped to  
 58 its embedding, the vector  $\vec{x}$ . A relationship  $r$ , likewise, is mapped to a vector  $\vec{r}$ . These  
 59 embeddings have the following property: If  $\vec{r}$  is the vector for the *livesIn* relationship, then we  
 60 can walk from the embedding  $\vec{x}$  of a person  $x$  to the embedding  $\vec{z}$  of their place of residence  
 61  $z$  by computing  $\vec{z} = \vec{x} + \vec{r}$ . This gives us another way of guessing the place of residence for  
 62 some person  $y$ : We just find the city whose embedding is closest to  $\vec{y} + \vec{r}$ .

63 Each of these methods has its advantages and disadvantages: While rules are easy to  
 64 understand for humans (and embeddings are less intuitively accessible), embeddings can  
 65 take into account signals from all facts in which an entity occurs (and not just the ones  
 66 mentioned in the rule, which are typically few). Therefore, recent years have seen fruitful  
 67 endeavors to combine neural methods with symbolic methods. Both rule mining techniques  
 68 and embedding techniques have been surveyed in recent articles [62, 9, 46, 73], among which  
 69 is our own previous tutorial article [54]. Hence, in this tutorial, we survey approaches that  
 70 combine both techniques.

71 The article is structured as follows: Section 2 introduces knowledge bases, rule mining  
 72 techniques, and embedding techniques, following largely [54]. Section 3 discusses embeddings  
 73 in more detail. Section 4 discusses embedding techniques that use rule mining techniques.  
 74 Section 5, vice versa, discusses rule mining techniques that use embedding techniques. We  
 75 conclude in Section 6.

## 2 Preliminaries

### 2.1 Knowledge Bases

**Knowledge Bases.** To define a knowledge base [54], we need a set  $\mathcal{I}$  of *entities*. An entity is anything that can be an object of thought [67]. General-purpose KBs are typically concerned with entities such as places (e.g., *Paris*, or *India*), people (such as politicians, scientists, or actors), organizations (such as companies or associations), or artworks (such as movies, books, etc.). But knowledge bases can also be concerned with biomedical entities, geological formations, scientific articles, or any other type of entities.

In what follows, we assume a set  $\mathcal{R}$  of binary *relation names* (also called *relations*, *relationships*, or *predicates*). For example, the relation *locatedIn* holds between a city and a country; the relation *actedIn* holds between an actor and a movie; and the relation *president-Of* holds between a person and a country. Finally, we need a set  $\mathcal{L}$  of *literals*. These are strings or numbers. A *fact* (or an *assertion*, *triple*, or *statement*) is then of the form  $\langle s, r, o \rangle$  with a *subject*  $s \in \mathcal{I}$ , a *relation*  $r \in \mathcal{R}$  and an *object*  $o \in \mathcal{I} \cup \mathcal{L}$  [30]<sup>1</sup>. An example of a fact is  $\langle \textit{Paris}, \textit{locatedIn}, \textit{France} \rangle$ . The *inverse* of a relation  $r$  is a relation  $r^-$ , so that  $\langle x, r, y \rangle$  holds if and only if  $\langle y, r^-, x \rangle$  holds. For example, the inverse of *hasNationality* is *hasCitizen*. A *knowledge base*  $\mathcal{K}$  over the sets  $\mathcal{I}, \mathcal{R}, \mathcal{L}$  is then a set of facts over these sets. Whenever  $\mathcal{K}$  is clear from the context, we write  $\langle s, r, o \rangle$  to mean  $\langle s, r, o \rangle \in \mathcal{K}$ .

**Taxonomies.** Knowledge bases typically also define *classes*. Intuitively, a class can be understood as a set of entities, its *instances*. For example, the class of capital cities contains the city of Paris, the city of Beijing, etc. Many formalisms use unary predicates to express class membership, stating, e.g., *city(Paris)*. If every instance of some class  $y$  is also an instance of some class  $y'$ , then  $y$  is called a *subclass* of  $y'$ . For example, the class *capitalCity* is a subclass of the class *city*, which is itself a subclass of *geographicLocation*. This gives us a hierarchy of classes – the *taxonomy*. Figure 2 shows an example of a taxonomy of classes.

Many KBs express the taxonomy by binary relations. To say that an entity  $x$  belongs to a class  $y$ , the KB adds the triple  $\langle x, \textit{type}, y \rangle$ . To say that a class  $y$  is a subclass of a class  $y'$ , we add  $\langle y, \textit{subclassOf}, y' \rangle$ . However, a taxonomy has an inherent semantics that is different from other facts that hold between entities, and therefore, one is usually ill-advised to treat the link  $\langle \textit{Paris}, \textit{type}, \textit{city} \rangle$  in the same way as  $\langle \textit{Paris}, \textit{locatedIn}, \textit{France} \rangle$ .

**Axioms.** KBs typically come with a set of logical constraints. For example, we can impose that if  $x$  is an instance of a class  $y$ , and if  $y$  is a subclass of the class  $y'$ , then  $x$  must also be an instance of  $y'$ :

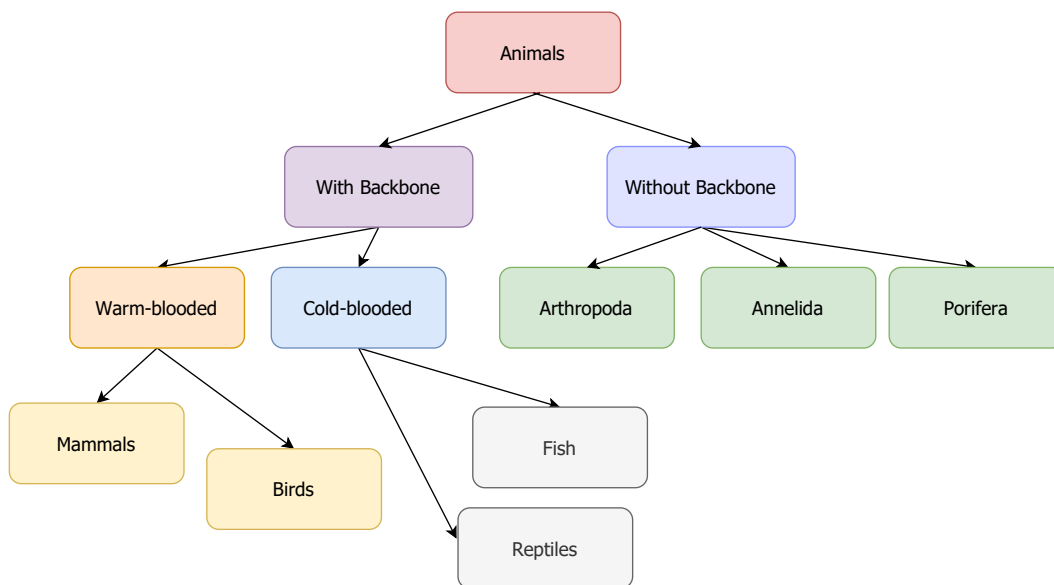
$$\langle x, \textit{type}, y \rangle \wedge \langle y, \textit{subclassOf}, y' \rangle \Rightarrow \langle x, \textit{type}, y' \rangle$$

Typical axioms are the following:

- **Domain and Range Constraints** say that the subject (resp. object) of a relation must belong to a certain class, as in “People are born in places (and not, say, in organizations)”.
- **Cardinality Constraints** say that the number of objects per subject for a certain relation is restricted, as in “People can have at most one birth place”.
- **Symmetry, transitivity, and inverse constraints** say that a relation is symmetric, transitive, or the inverse of another relationship.
- **Disjointness constraints** say that two classes cannot have instances in common, e.g., places and people.

<sup>1</sup> For our purpose, in line with the other works [17, 18, 40], we do not consider blank nodes.

## 4:4 Combining Embeddings and Rules for Fact Prediction



■ **Figure 2** Taxonomy Example

119 Such axioms exist in packages of different complexity: The Resource Description Framework  
120 Schema *RDFS* is a system of basic axioms that are concerned mainly with class membership.  
121 The axioms are so basic that they cannot result in contradictions. The Web Ontology  
122 Language *OWL* is a system of axioms that exists in several flavors – from the simple to  
123 the undecidable [54]. Such packages of axioms, together with the taxonomy, are sometimes  
124 called *ontology* or *schema*. Automated reasoners can be used to (1) predict facts that follow  
125 logically from these axioms and (2) determine whether a KB is inconsistent with respect to  
126 these axioms.

127 **Fact Prediction.** In what follows, we will assume an ideal knowledge base  $\mathcal{K}^*$ , which  
128 contains all facts of the real world (see [44] for a discussion of such a KB). One typically  
129 assumes that all facts in some given KB  $\mathcal{K}$  are also true in the real world, i.e.,  $\mathcal{K} \subseteq \mathcal{K}^*$ .  
130 However, the KBs are typically incomplete, i.e., there are facts in the real world that are not  
131 in the KB (i.e.,  $\mathcal{K} \subsetneq \mathcal{K}^*$ ). Predicting a fact  $f$  that is true in the real world, but not yet in  
132 the KB, is called the problem of *fact prediction*.

133 **World Assumptions.** Fact prediction is complicated by the fact that the KBs typically do  
134 not store negative information [43]. That is: while a KB may store that Elvis Presley has  
135 sung the song “All Shook Up”, it will not store the fact that he did not sing the song “The  
136 Winner Takes It All”. This raises the question what we should do if the KB does not contain  
137 certain statements (e.g., the KB does not contain the fact that Elvis sang “Always on my  
138 mind”, which is true in the real world). In a database, one would assume that any fact that  
139 does not appear in our data is not true in the real world – an assumption known as the  
140 *Closed World Assumption*. This assumption, however, is usually false for KBs, as KBs are  
141 highly incomplete and miss many facts from the real world. Hence, it is more appropriate to  
142 make the *Open World Assumption*, which says that if an assertion is not in the KB, it may  
143 or may not be true in the real world. Thus, in our example, if the KB does not contain the  
144 assertion that Elvis sang “Always on my mind”, we would not be entitled to conclude that  
145 this assertion would be false in the real world (which it is indeed not).

146 **Negative assertions.** A negative assertion is a statement that is known to be false. Such

147 statements are essential as counter-examples in rule mining and fact prediction, so as to  
 148 avoid an over-generalization. For example, Woody Allen married his step-daughter. If we  
 149 find 10 other people who married their step-daughter, and no person who is *not* married to  
 150 their step-daughter, we would conclude that people in general marry their step-daughters.  
 151 The problem is now that KBs do not contain negative assertions. No KB tells us that Elvis  
 152 Presley was *not* married to his step-daughter. And the Open World Assumption prevents us  
 153 from assuming this negative assertion from the facts that are in the KB. This means that we  
 154 have, in theory, no way to generate counter-examples for rule mining and fact prediction.  
 155 Hence, we could mine the rule “Everybody is married to their step-daughter” without any  
 156 obstruction.

157 Several remedies have been proposed. One is the *Partial Completeness Assumption*, or  
 158 *Local Closed World Assumption* [17]. It says that if a KB contains the facts  $\langle s, r, o_1 \rangle, \dots,$   
 159  $\langle s, r, o_n \rangle$ , then any fact  $\langle s, r, o' \rangle$  with  $o' \notin \{o_1, \dots, o_n\}$  must be false in the real world. The  
 160 rationale is that if some contributor made the effort to add the objects  $o_1, \dots, o_n$ , they would  
 161 for sure also have added any remaining object  $o'$ . It can be shown that this assumption is  
 162 generally true for relations that have few objects, such as *hasBirthDate* or *hasNationality* [18].  
 163 Indeed, in most KBs, the relations are designed in such a way that the average number of  
 164 objects per subject is lower than the average number of subjects per object [18]. For example,  
 165 a KB is more likely to contain the relation *hasNationality* (one person has few nationalities)  
 166 rather than *hasCitizen* (one country has millions of citizens). A relation that has a higher  
 167 average number of objects per subject than subjects per object can simply be replaced by its  
 168 inverse [18]. With this, the PCA works generally well.

169 The method can be used as follows to generate a large number of negative examples: take  
 170 any fact  $\langle s, r, o \rangle$  from the KB, replace  $o$  by a randomly chosen object  $o'$  such that  $\langle s, r, o' \rangle$  is  
 171 not in the KB, and assume that  $\langle s, r, o' \rangle$  is a negative assertion. The assertion  $\langle s, r, o' \rangle$  is  
 172 called a *corrupted* variant of  $\langle s, r, o \rangle$ . The method is also often applied in the same way to  
 173 the subjects of the triples. This, however, creates a problem: Since relations generally have  
 174 more subjects per object than vice versa, the PCA is much less plausible in this setting. For  
 175 example, while it is, under the PCA, safe to assume that if some person Mary is American,  
 176 she is not French, it is not safe to assume there are no more Americans than those in the  
 177 KB. This is why the original PCA is applied only to the objects.

## 178 2.2 Rule Mining

179 **Rules and Axioms.** We have already seen that KBs can come with axioms, such as the  
 180 symmetry of a relation. These axioms are usually defined manually, and they allow no  
 181 exceptions. In what follows, we will be concerned with *rules*. These also express constraints  
 182 on the data, but different from axioms, they are not imposed on the data, but automatically  
 183 mined from the data. As such, they also allow for exceptions. For example, we can find that  
 184 *marriedTo* is “usually” symmetric in the data of a given KB, meaning that for most couples,  
 185 the *marriedTo* fact holds in both directions – although there are some couples for which the  
 186 relation holds only in one direction, presumably because of missing data. This is why such  
 187 rules are also called *soft rules* (as opposed to the “hard” axioms). Let us now make this idea  
 188 more formal.

189 **Atoms and Rules.** An *atom* is an expression of the form  $\langle \alpha, r, \beta \rangle$ , where  $r$  is a relation  
 190 and  $\alpha, \beta$  are either entities or variables [30] (we write variables in lower case, and entities in  
 191 upper case). For example,  $\langle x, \textit{livesIn}, \textit{Berlin} \rangle$  is an atom with one variable,  $x$ . An atom is  
 192 *instantiated* if at least one of its arguments is an entity. If both arguments are entities, the atom  
 193 is *grounded* and tantamount to a fact. A *conjunction* of atoms  $B_1, \dots, B_n$  is of the form  $B_1 \wedge \dots \wedge$

194  $B_n$ . For example, we can build the conjunction  $\langle x, \text{livesIn}, \text{Paris} \rangle \wedge \langle x, \text{wasBornIn}, \text{Berlin} \rangle$ ,  
 195 which, intuitively, designates all people  $x$  who were born in Berlin and live in Paris. To make  
 196 this intuition more formal, we need the notion of a substitution. A *substitution*  $\sigma$  is a partial  
 197 mapping from variables to entities. Substitutions can be straightforwardly extended to atoms  
 198 and conjunctions. For example, the substitution  $\sigma = \{x \rightarrow \text{Mary}\}$  can be applied to our  
 199 conjunction above, and it yields  $\langle \text{Mary}, \text{livesIn}, \text{Paris} \rangle \wedge \langle \text{Mary}, \text{wasBornIn}, \text{Berlin} \rangle$ .

200 A (Horn) *rule* is a formula of the form  $B_1 \wedge \dots \wedge B_n \Rightarrow H$ , where the  $B_1 \wedge \dots \wedge B_n$  is a  
 201 conjunction of *body atoms*, and  $H$  is the *head atom*. An example for a rule is

$$202 \quad \langle x, \text{married}, y \rangle \wedge \langle x, \text{livesIn}, z \rangle \Rightarrow \langle y, \text{livesIn}, z \rangle$$

203 Let us call this rule  $R^*$  in what follows. Two atoms  $A, A'$  are *connected* if they have common  
 204 variables. It is common [17, 18, 40] to impose that all atoms in a rule are transitively  
 205 connected and that rules are closed. A rule is *closed* if every variable in the head appears in  
 206 at least one atom in the body. A rule is *grounded* if all of its atoms are grounded.

207 **Predictions.** Given a rule  $R = B_1 \wedge \dots \wedge B_n \Rightarrow H$  and a substitution  $\sigma$ , we can apply  $\sigma$  to both  
 208 the body and the head of  $R$ , and obtain an *instantiation* of  $R$ , which we denote by  $\sigma(R)$ . In our  
 209 example, we could instantiate the above rule  $R^*$  by  $\sigma = \{x \rightarrow \text{Mary}, y \rightarrow \text{Bob}, z \rightarrow \text{Paris}\}$ ,  
 210 and obtain  $\sigma(R^*)$  as

$$211 \quad \langle \text{Mary}, \text{married}, \text{Bob} \rangle \wedge \langle \text{Mary}, \text{livesIn}, \text{Paris} \rangle \Rightarrow \langle \text{Bob}, \text{livesIn}, \text{Paris} \rangle$$

212 If  $\sigma(B_i) \in \mathcal{K} \forall i \in \{1, \dots, n\}$ , we call  $\sigma(H)$  a *prediction* of  $R$  from  $\mathcal{K}$ , and we write  $\mathcal{K} \wedge R \models \sigma(H)$ .  
 213 Suppose, e.g., that we have a KB  $\mathcal{K} = \{\langle \text{Paris}, \text{locatedIn}, \text{France} \rangle, \langle \text{Mary}, \text{married}, \text{Bob} \rangle,$   
 214  $\langle \text{Mary}, \text{livesIn}, \text{Paris} \rangle\}$ . Here, our example rule  $R^*$  can be instantiated as before by  
 215  $\sigma = \{x \rightarrow \text{Mary}, y \rightarrow \text{Bob}, z \rightarrow \text{Paris}\}$ . Then, all body atoms of the instantiated rule  $\sigma(R^*)$   
 216 appear in  $\mathcal{K}$ . Hence, the rule predicts the head atom of  $\sigma(R^*)$ , which is  $\langle \text{Bob}, \text{livesIn}, \text{Paris} \rangle$ .  
 217 Hence, we write  $\mathcal{K} \wedge R^* \models \langle \text{Bob}, \text{livesIn}, \text{Paris} \rangle$ .

218 **Mining Rules.** Inductive Logic Programming (ILP) is the task of finding rules automatic-  
 219 ally [54]. Typically, one provides a set of *positive examples* (i.e., facts that the rules shall  
 220 predict), and a set of *negative examples* (facts that the rules must not predict). In the  
 221 context of KBs, ILP faces several challenges: First, KBs usually do not provide negative  
 222 examples. We have discussed a method to generate negative examples above, the Partial  
 223 Completeness Assumption (Section 2.1). Another challenge is that a strict application of  
 224 the definition of ILP to rule mining would find only rules that are true in all instantiations.  
 225 However, in real-world KBs, there can be exceptions to rules, e.g., due to faulty or missing  
 226 data. Hence, rule mining typically aims for rules that have a high *support* (the number of  
 227 positive examples predicted by the rule), and a high *confidence* (the proportion of examples  
 228 it predicts that are positive). In this way, the methods can find rules even if they do not  
 229 apply in all instances, such as “If two people are married, then the children of one of them  
 230 are also the children of the other”.

231 AMIE [17] was one of the first rule mining systems for large KBs under the Open World  
 232 Assumption. It starts with the most general rules (such as “everybody is married with  
 233 each other”), and refines them until their confidence is high enough (e.g., “if two people are  
 234 parents of the same children, they are most likely married”). This relies on the observation  
 235 that the support of a rule decreases monotonically when a rule is made more specific. The  
 236 RuDiK system [40] can mine logical rules like AMIE, but brings a number of improvements:  
 237 First, RuDiK can also mine negative rules, such as “If two people are siblings, they are not  
 238 married”. Second, RuDiK can mine relations between literals, such as “Someone’s birth date  
 239 is always before someone’s death date”. Finally, RuDiK removes facts that have been covered

240 by a rule, so that subsequent rules are forced to predict facts that have not already been  
 241 predicted. This allows not just for some optimizations of the mining algorithm, but also to  
 242 mine rules that predict more unknown facts correctly.

243 The AnyBURL system [34] is a bottom-up rule mining system: It starts with path rules  
 244 that are specific to one instance, and generalizes them to achieve good support. A particular  
 245 advantage of the system is that the user can trade running time for rule quality, i.e., get  
 246 better rules by waiting longer.

247 The DRUM system [49] is a linear formulation of the rule mining problem using one-  
 248 hot-encoding vectors for entities and adjacency matrices for relations. As it is linear, the  
 249 problem is fully differentiable and can then be solved using gradient descent techniques. This  
 250 solving approach proved to be very good for predictions involving previously unseen entities  
 251 or relations.

252 Let us now turn to the second family of methods that can be used to predict missing  
 253 facts: Knowledge Graph Embeddings.

## 254 2.3 Embeddings

255 **Embeddings.** An embedding for a group of objects (e.g. words, relations, or entities) is an  
 256 injective function that maps each object to a real-valued vector, so that the intrinsic relations  
 257 between the objects are maintained [54]. In the case of KBs, we are looking to embed entities  
 258 and relations. In particular, given a KB, we would want the entities that are semantically  
 259 similar in the KB to be mapped to vectors that are close to each other in the vector space.

260 The most basic embeddings [7] are designed so that, for a fact  $\langle s, r, o \rangle$ , we have  $\vec{s} + \vec{r} \approx \vec{o}$ ,  
 261 where  $\vec{v}$  is the embedding vector of the underlying entity or relation. For example, if we know  
 262  $\langle \text{Elvis}, \text{marriedTo}, \text{Priscilla} \rangle$ , then we would want the vector  $\vec{\text{Elvis}} + \vec{\text{marriedTo}}$  to be close  
 263 to the vector  $\vec{\text{Priscilla}}$ . An embedding with these properties has several advantages: First,  
 264 the embedding allows us to feed entities and relations into machine learning methods that  
 265 work on vectors (e.g., classification algorithms). The vectors are typically low in dimension  
 266 (e.g., a few hundred), which makes them particularly suited for such applications. Second,  
 267 the embedding provides a natural way of grouping together similar entities, so that given  
 268 one entity, we can find its peers by scanning the vector space. In our example, we would  
 269 expect Elvis to be close in the vector space to other singers. Finally, the embeddings allow  
 270 for link prediction: If we do not know the spouse of Elvis, we can just compute the vector  
 271  $\vec{\text{Elvis}} + \vec{\text{marriedTo}}$  and propose that the person that we find there is the spouse. If the  
 272 embedding is well designed, that would actually work.

273 **Terminology.** In the literature about KB embeddings, the KB is often called a *knowledge*  
 274 *graph* (KG) instead of a *knowledge base*. This is because embedding approaches typically  
 275 project away literals and facts with literals. Consequently, fact prediction is known as *link*  
 276 *prediction* in this scenario. Furthermore, the approaches typically do not deal with classes,  
 277 taxonomies, or axioms. What remains is then a graph where the nodes are entities, and the  
 278 edges are relations. In this scenario, facts are usually called *triples*, the subject is called the  
 279 *head* of the triple, and the object is called the *tail*.

280 **Link prediction with embeddings.** Knowledge graph embeddings are created by trainable  
 281 machine-learning models, typically neural networks. We will discuss these methods in detail  
 282 in Section 3. All of these models take as input a fact  $\langle h, r, t \rangle$ , and output a score of its  
 283 likelihood of being true: the higher the score, the more likely the model believes the fact  
 284 to be. This score is typically denoted by  $f(\langle h, r, t \rangle)$  or  $f_{\vec{r}}(\vec{h}, \vec{t})$ . To train such a model,  
 285 we need a KB of true facts. We train the model to give a high score to these facts. To  
 286 avoid over-generalization, we also have to train the model with counter-examples. These are



287 typically generated by corrupting the facts from the KB (Section 2.1), i.e., by taking a fact  
 288  $\langle h, r, t \rangle$  from the KB and replacing the tail by a random entity  $t'$ . The model is then trained  
 289 to give the true triples from the input KB a higher score than the corrupted triples.

290 We can then use the models for link prediction as follows: We take a partially-filled triple  
 291 for which we would like to know the head or tail entity, e.g.,  $\langle \text{Elvis}, \text{marriedTo}, ? \rangle$ . We try  
 292 out all possible tail entities from the KB, and score the resulting triple using the scoring  
 293 function. The predicted entity is intuitively the one with the highest resulting score. All  
 294 entities can be sorted according to the scores of their triple. Each entity is then associated  
 295 to its *prediction rank*, i.e., to the position that it has in the ranked list of predictions.

296 In the supervised setting, we often know the true answer (*Priscilla*), and we can compute  
 297 its prediction rank  $PR_{\langle \text{Elvis}, \text{marriedTo}, ? \rangle}(\text{Priscilla})$ . Several metrics are computed from the  
 298 prediction ranks of head and tail entities. If  $\mathcal{T}$  the set of known true facts, the metrics are  
 299 the following:

- Mean Rank (MR): the average prediction ranks of the correct entities

$$MR = \frac{1}{2 |\mathcal{T}|} \left( \sum_{(h,r,t) \in \mathcal{T}} PR_{\langle ?, r, t \rangle}(h) + PR_{\langle h, r, ? \rangle}(t) \right)$$

- Mean Reciprocal Rank (MRR): the average of the inverse of the prediction ranks

$$MRR = \frac{1}{2 |\mathcal{T}|} \left( \sum_{(h,r,t) \in \mathcal{T}} \frac{1}{PR_{\langle ?, r, t \rangle}(h)} + \frac{1}{PR_{\langle h, r, ? \rangle}(t)} \right)$$

- Hit at  $k$  (Hit@ $k$ ): proportion of the tests in which the prediction rank is better than  $k$   
 (typical values for  $k$  are 1, 3 and 10)

$$Hit@k = \frac{1}{2 |\mathcal{T}|} \left( \sum_{(h,r,t) \in \mathcal{T}} \mathbb{1}\{PR_{\langle ?, r, t \rangle}(h) \leq k\} + \mathbb{1}\{PR_{\langle h, r, ? \rangle}(t) \leq k\} \right)$$

300 Both MRR and Hit@ $k$  have values between 0 and 1, higher values indicate better results. In  
 301 some cases, multiple entities can be correct answers (e.g. for 1-N relations) and the model  
 302 should not be penalized for predicting another true answer that is simply more likely than  
 303 the one at hand. Those metrics are usually computed in a *filtered* setting in which prediction  
 304 ranks are computed by removing the other true entities ranked better than the one at hand.

### 305 **3 Embedding Models**

306 In the last decade, numerous methods for computing knowledge graph embeddings have been  
 307 proposed. The methods differ from one another in terms of how they relate the entities and  
 308 relations of the KG in the latent space. The existing models can be categorized as geometric,  
 309 tensor-based or convolutional. In this section, we introduce and discuss a few popular models  
 310 from each category.

311 In the following, let us consider a KG with  $n$  entities  $\mathcal{E} = \{e_1, \dots, e_n\}$  and  $m$  relations  
 312  $\mathcal{R} = \{r_1 \dots r_m\}$  that is to be embedded in a  $d$ -dimensional vector space.  $\mathbb{R}$  (resp.  $\mathbb{C}$ ) is the  
 313 field of real (resp. complex) numbers.

#### 314 **3.1 Geometric models**

315 Geometric models interpret relations as geometric operations in the vector space. The earliest  
 316 of these models is TransE, which we now describe in detail.

317 **TransE** [7] is a *translation-based model*, i.e., it uses a geometric distance to measure the  
 318 similarity of the entities. Given a fact  $\langle h, r, t \rangle$ , its goal is to find vectors  $\vec{h}, \vec{r}, \vec{t}$ , so that  
 319  $\vec{h} + \vec{r} \approx \vec{t}$ .

320 One way to do that is to design a neural network [54]. We first create a *vocabulary*,  
 321 i.e., an ordered list of all entities in the KG. Then we create, for each entity, its *one-hot*  
 322 *encoding*. This is simply a vector that has as many dimensions as there are entities. Every  
 323 element of the vector is set to zero, and only the  $i^{\text{th}}$  element is set to one, where  $i$  is the  
 324 position of the entity in the vocabulary. The same is done for the relations. Then we design  
 325 a network as follows: The input is the one-hot encoding of the head, the one-hot encoding of  
 326 the relation, and the one-hot encoding of the tail of a given fact from the KB. That is, if  
 327  $n$  is the number of entities, and  $m$  is the number of relations, the network has  $m + 2 \times n$   
 328 input neurons. The first hidden layer of the network then maps each of these vectors to a  
 329  $d$ -dimensional real vector in  $\mathbb{R}^d$ . An entity  $e$  is mapped to a vector  $\vec{e}$ , and a relation  $r$  is  
 330 mapped to  $\vec{r}$ . The further layers then reduce these vectors to a single output that scores the  
 331 input assertion. More precisely, the network computes, for an input fact  $\langle h, r, t \rangle$  from the  
 332 KB, the function  $f_{\vec{r}}(\vec{h}, \vec{t}) = -\|\vec{h} + \vec{r} - \vec{t}\|$  (where  $\|\cdot\|$  is either the 1-norm or the 2-norm).  
 333 The network is then trained with facts from the KB to maximize this score for these facts. It  
 334 is trained with negative assertions to minimize this score. This leads to embeddings that  
 335 verify the simple arithmetic equation  $\vec{h} + \vec{r} \approx \vec{t}$  [54]. The important thing here is that the  
 336 later hidden layers take their decision based solely on the output of the first hidden layer.  
 337 The vectors computed by the first layer thus contain all the necessary information to assess  
 338 the truth value of an assertion – and this is what we want from a good embedding. Thus, we  
 339 will use the vectors that the first layer outputs as the embeddings of the input entities.

340 One limitation of TransE is the inability to model symmetric relationships [65]: if  $r$   
 341 is symmetric (i.e.  $\langle h, r, t \rangle$  true implies  $\langle t, r, h \rangle$  to be true as well), then  $r$  tends to have  
 342 an embedding vector close to  $\vec{0}$  because minimizing both  $\|\vec{h} + \vec{r} - \vec{t}\|_2$  and  $\|\vec{t} + \vec{r} - \vec{h}\|_2$   
 343 simultaneously happens if and only if  $\vec{r} = \vec{0}$ . Another problem appears with one-to-many  
 344 relations. Consider for example the facts  $\langle \text{ElonMusk}, \text{founderOf}, \text{SpaceX} \rangle$  and  $\langle \text{ElonMusk},$   
 345  $\text{founderOf}, \text{Tesla} \rangle$ . TransE would give very similar embeddings to both *SpaceX* and *Tesla*,  
 346 and thus fail to differentiate between the two companies. TransE also has problems modeling  
 347 many-to-one, reflexive, and transitive relations, and to capture multiple semantics of a  
 348 relation.

349 **TransH** [65] tries to alleviate some limitations of TransE by allowing an entity to have  
 350 different representations in the embedding space depending on the relation it is involved  
 351 with. Each relation  $r$  is represented not only by a vector  $\vec{r}$ , but also by an hyperplane (i.e. a  
 352 sub-space of one dimension less than the embedding space). Algebraically an hyperplane can  
 353 be defined by a single vector, namely the vector that is orthogonal to it. Thus, each relation  
 354  $r$  is associated with a set of two vectors:  $\vec{r}$  for the relation itself, and  $\vec{h}_r$  for its hyperplane.

355 To compute the score of a triple  $\langle h, r, t \rangle$ , the embeddings  $\vec{h}, \vec{t}$  of the entities are first  
 356 projected onto the hyperplane defined by  $\vec{h}_r$ , and they are then connected by the translation  
 357 vector  $\vec{r}$  of the relation. Given a relation  $r$ , let  $p_r$  be the linear orthogonal projection  
 358 on the hyperplane defined by  $\vec{h}_r$ . Then the loss function of TransH can be written as  
 359  $f(\langle h, r, t \rangle) = f_{\vec{r}}(\vec{h}, \vec{t}) = -\|p_r(\vec{h}) + \vec{r} - p_r(\vec{t})\|_2^2$ .

360 This is designed to solve the limitations of TransE: a reflexive relation  $r$  can have a  
 361 translation vector  $\vec{r}$  close to  $\vec{0}$ , since all information is contained in  $\vec{h}_r$ . For relations with  
 362 several objects, likewise, the objects can be embedded in the same place in the hyperplane  
 363 only for that specific relation.

364 **TransR** [31] extends the idea of sub-space projection of TransH by proposing that the

365 projection step is now done on any sub-space of a given dimension. Let  $d$  be the dimension of  
 366 the embedding space and  $d'$  the dimension of the relation-specific sub-spaces. Algebraically a  
 367 linear projection from a vector space of dimension  $d$  into one of its sub-spaces of dimension  $d'$   
 368 is simply represented by a matrix of dimension  $d \times d'$ . Each relation  $r$  is then represented by  
 369 a vector  $\vec{r}$  and a projection matrix  $M_r$ . Thus, TransR is simply an evolution of TransH that  
 370 increases the expressiveness of the model by increasing the number of parameters. Intuitively,  
 371 this should allow the model to *learn* a greater amount of useful information from the known  
 372 facts it is trained on. CTransR [31] is an extension of TransR, which operates by clustering  
 373 diverse head-tail entity pairs into groups and learning distinct relation vectors for each group.  
 374 **TransD** [25] in turn proposes to keep the idea of projecting on any possible sub-space but  
 375 reduces the number of parameters compared to TransD in order to limit the risk of overfitting.  
 376 This is done by allowing only the sub-space projections that are defined by a *low-rank* matrix:  
 377 that is a matrix that can be decomposed as a product of vectors.

378 Several other improvements have also been proposed in the direction of translation  
 379 embedding methods, including TransG [68], TransF [16], and KG2E [21]. Other geometric  
 380 models perform rotation-like transformations in the vector space instead of pure translations.  
 381 Its most prominent examples are RotatE [55] and HAKE [72].

382 **RotatE** [55] aims to be particularly suited for relations that are symmetric, anti-symmetric,  
 383 inverses of each other, and compositions of each other, which are typical for KGs. For instance,  
 384 the relation *marriedTo* is a symmetric relation:  $\langle x, \text{marriedTo}, y \rangle$  implies  $\langle y, \text{marriedTo},$   
 385  $x \rangle$ . Further, many relations such as familial relations are compositional. For example,  $\langle x,$   
 386  $\text{hasParent}, y \rangle$  and  $\langle y, \text{hasParent}, z \rangle$  imply  $\langle x, \text{hasGrandParent}, z \rangle$ . RotatE captures these  
 387 relation patterns by defining each relation as a rotation from the head entity to the tail entity  
 388 in the vector space. Specifically entities and relations are now embedded in  $\mathbb{C}^d$  and for any  
 389 relation  $r$ , the modulus of each component  $\vec{r}_i$  is 1. For a triple  $\langle x, r, y \rangle$ , the model then tries  
 390 to achieve  $\vec{y} \approx \vec{x} \circ \vec{r}$ , where  $\circ$  is the element-wise product. Intuitively, a relation  $r$  applies a  
 391 coordinate-wise rotation on the head entity so as to come close to the tail entity. The score  
 392 function is then  $\|\vec{x} \circ \vec{r} - \vec{y}\|$ . A relation is symmetric if and only if its embedding belongs to  
 393  $\{-1, +1\}^d$  (i.e. coordinate-wise rotations of 0 or  $\pi$  radians),  $r_1$  and  $r_2$  are symmetric if  
 394 and only if their embeddings are complex conjugates, and a relation  $r_3$  is the composition of  
 395 two relations  $r_1$  and  $r_2$  if and only if  $\vec{r}_3 = \vec{r}_1 \circ \vec{r}_2$  (i.e., the coordinate-wise rotations of  $r_3$  are  
 396 the successive rotations of  $r_1$  and  $r_2$ ).

397 **HAKE** [72] extends the RotatE embeddings by taking into account and preserving the  
 398 semantic hierarchies of the entities in the KGs. For example, the entity *Paris* is part of  
 399 *France*, which is a part of the EU. Such hierarchies between entities are quite common in  
 400 most KGs such as Yago and Freebase. To model these relations between entities, HAKE  
 401 represents an entity  $e$  (and a relation  $r$ ) in the vector space in two parts: as  $\vec{e}_m$  and  $\vec{r}_m$   
 402 in the modulus part and as  $\vec{e}_p$  and  $\vec{r}_p$  in the phase part. The modulus part is aimed at  
 403 differentiating entities at different hierarchies from each other, such as *Paris* from *France*,  
 404 while the phase part distinguishes the different entities at the same hierarchy level, e.g. *Paris*  
 405 and *Lyon*. In this manner, HAKE is able to represent the semantic hierarchies associated  
 406 with KGs, and outperform other techniques by learning better embeddings.

### 407 3.2 Semantic Matching models

408 Another common category of embedding methods compares the vector of the subject and  
 409 the vector of the object directly in order to assess how likely the fact is to be true.

410 **RESCAL** [38] is the simplest model in this category. Entities are represented as vectors  
 411 and relations become bilinear functions (simply represented as square matrices). A triple

412  $\langle h, r, t \rangle$  is then scored by the application of the relation-specific bilinear function to the entity  
 413 embeddings:  $f(\langle h, r, t \rangle) = \vec{h}^t \cdot M_r \cdot \vec{t}$ , where  $\vec{h}$  (resp.  $\vec{t}$ ) is the embedding of  $h$  (resp.  $t$ ) and  
 414  $M_r \in \mathbb{R}^{d \times d}$  is the representing matrix of  $r$ . Intuitively, this bi-linear scoring function can  
 415 be interpreted as some sort of *scalar product* between the entities in some relation-specific  
 416 distortion of the embedding space. This is simply an intuition as no sufficient constraints  
 417 are imposed on the relation matrices to make them scalar products. Precisely, they are not  
 418 forced to be symmetric nor positive definite.

419 **DistMult** [69] is a variation of the RESCAL models where the relation matrices are all  
 420 forced to be diagonal. This simplifies the computations, and reduces the parameter space.  
 421 As a drawback, DistMult gives the same score for the triples  $\langle h, r, t \rangle$  and  $\langle t, r, h \rangle$ . Thus, it is  
 422 unable to model asymmetric relations such as *sonOf*, *actedIn* etc. Despite these limitations,  
 423 DistMult has been recently shown to perform as well as many recently proposed models,  
 424 presumably due to its simplicity and scalability [48].

425 **Complex** [58] improves upon the DistMult model by using the same diagonal constraint, but  
 426 with complex-valued embedding vectors. Entities and relations are then simply represented  
 427 as vectors in  $\mathbb{C}^d$  and the Hermitian product is used instead of the bi-linear product in the  
 428 scoring function. This allows the approach to take into account asymmetric relations in the  
 429 KGs, as in the triple  $\langle \textit{Paris}, \textit{capitalOf}, \textit{France} \rangle$  (where France is not the capital of Paris).  
 430 The scoring function is defined as  $f(\langle h, r, t \rangle) = \text{Re}(\vec{h} \cdot M_r \cdot \vec{t})$  where  $\text{Re}(c)$  is the real part of  
 431  $c \in \mathbb{C}$  and  $M_r$  is the diagonal matrix with  $\vec{r}$  on its diagonal. The fact that the Hermitian  
 432 product is not commutative solves the problem of representing asymmetric relations and  
 433 switching to complex vector space doubles the number of parameters thus increasing the  
 434 expressiveness of the model.

435 **Simple** [27] proposes to extend one of the most generic multiplicative methods: Canonical  
 436 Polyadic (CP) decomposition [22]. This method is used for decomposing tensors into a sum of  
 437 products. It can be applied to KG embeddings because a KG with  $n$  entities and  $m$  relations  
 438 is simply represented as a 3-dimensional adjacency tensor  $\mathcal{T} \in \{0, 1\}^{n \times n \times m}$ :  $\mathcal{T}[i, j, k] = 1$   
 439 if  $\langle e_i, r_k, e_j \rangle$  is true and 0 else. As explained in [27], CP decomposition represents entities  
 440  $e$  with two vectors  $(\vec{h}_e, \vec{t}_e) \in (\mathbb{R}^d)^2$  and relations  $r$  with a vector  $\vec{r} \in \mathbb{R}^d$  where  $d$  is the  
 441 dimension of the embedding. These vectors are learned in order to be able to reconstruct  
 442 the tensor  $\mathcal{T}$  by estimating  $\hat{\mathcal{T}}[i, j, k] = \langle \vec{h}_{e_i}, \vec{t}_{e_j}, \vec{r}_k \rangle = \sum_{\ell=1..d} \vec{h}_{e_i}[\ell] \times \vec{t}_{e_j}[\ell] \times \vec{r}_k[\ell]$ . This  
 443 estimation is used in the case of KG embeddings as a scoring function of triples. Simple  
 444 just proposes to represent relations  $r$  with two vectors  $\vec{r}$  and  $r^{-1}$ , the scoring function being  
 445 now  $f(e_i, r, e_j) = \frac{1}{2}(\langle \vec{h}_{e_i}, \vec{t}_{e_j}, \vec{r} \rangle + \langle \vec{h}_{e_j}, \vec{t}_{e_i}, r^{-1} \rangle)$ . The authors show that their model is fully  
 446 expressive, meaning that if given enough embedding dimensions it can exactly represent any  
 447 KG. It is then argued that simple logical constraints can be implemented in the model by  
 448 applying constraints on the relation embeddings. We will later see one such application in  
 449 Section 4.3.

### 450 3.3 Deep Models

451 Deeper neural architectures have also been introduced for KB embeddings, with the hope that  
 452 hidden layers can capture more complex interaction patterns between entities and relations  
 453 (and then estimate more complex scoring functions). In such models, the first part of the  
 454 network (which, in shallow networks, just maps facts to their embeddings or their projections)  
 455 now adds additional layers (possibly numerous) that receive as inputs the embeddings, and  
 456 produce as outputs some extracted features. The second part of the network now computes  
 457 the scoring function from the features extracted by the first part of the network, and not  
 458 directly from the embedding (or its projection) as in shallow models. The scoring function

459 also becomes a parameter of the model (to be trained) and is not defined a priori anymore.  
 460 This often entails that we lose the interpretability of the scoring function [54]. There are  
 461 many deep neural network based models that have been proposed over the years, early  
 462 examples of such models are SME, NTN and MLP [6, 52, 14].

463 **NTN** [52] was introduced by Socher et al. as a generalization of the RESCAL model. It  
 464 employs a combination of linear transformations and nonlinear activation functions to obtain  
 465 head and tail embeddings. As such, while this is a more expressive model, it is also quite  
 466 complex with a large number of parameters that are harder to train. Better and lightweight  
 467 architectures have been since proposed, such as MLP, where the parameters are shared  
 468 among all the relations.

469 **ConvE** and **ConvKB** [12, 10] are popular examples of models that are based on convolu-  
 470 tional neural networks (CNN). These can learn complex nonlinear features of the entities  
 471 and relations with fewer parameters by using 2D convolutions over embeddings. ConvE has  
 472 been shown to be particularly effective for complex graph with nodes having a high number  
 473 of incoming edges. The model introduced the 1-N scoring scheme where for a given triple  
 474  $\langle h, r, t \rangle$  where  $t$  is to be predicted, the matching is performed with all the tail entities at the  
 475 same time, leading to speedier training. ConvE has proven to be a competitive embedding  
 476 model and a popular baseline for more recent deep learning approaches.

477 **Graph Convolutional Networks (GCNs)** have recently gained popularity for performing  
 478 link prediction in knowledge graphs in tandem with standard embedding techniques. GCNs  
 479 are a form of message passing multi-layer neural networks, first introduced by [28] for semi-  
 480 supervised node classification on graph structured datasets. One layer of GCN encodes  
 481 information about the immediate neighbours of a node in feature vector, and  $k$  layers stacked  
 482 on top of each other can encode the information of the neighbourhood  $k$  hops away. GCNs  
 483 can overcome the limitations of knowledge graph embedding models in terms of neglecting  
 484 the attributes of the entities and ignoring the graph structure by encoding the entities based  
 485 on their neighbours in the graph. Several extensions of GCNs have been suggested for  
 486 multi-relational knowledge graphs.

487 **Relational GCNs (R-GCNs)** [50] are GCNs for graphs with a large number of relations,  
 488 which makes them particularly suitable for knowledge graphs. Link prediction is essentially  
 489 an auto-encoder framework: An encoder creates the feature representations for the entities  
 490 from its neighbours (these features are generated from relation-specific transformations that  
 491 are dependent on the type and direction of the relations). A decoder (in this case, DistMult  
 492 factorization) is a scoring function to predict the labelled edges. R-GCNs show improvements  
 493 compared to DistMult, HolE and ComplEx for the link prediction task. While R-GCN  
 494 extended the GCN models on knowledge graphs by including the different types of relations  
 495 during the generation of entity representations, they do not represent relations themselves.

496 **VR-GCN** [70] is an extension of the R-GCN model that generates both entity and relation  
 497 embeddings explicitly. It ensures relation representation and different entity roles (head or  
 498 tail in different triples), and it conforms to the translation representation from translational  
 499 embeddings where  $\vec{h} + \vec{r} \approx \vec{t}$ . While the primary goal of this technique is to enable graph  
 500 alignment, the performance of VR-GCN is also discussed in terms of the link prediction task,  
 501 with VR-GCN acting as the encoder and DistMult as the decoder for scoring the triples.

502 **SACN** [51] leverages a variant of the existing knowledge graph embeddings ConvE and  
 503 TransE as decoder along with a variant of GCN (weighted GCN) as encoder. The weighted  
 504 GCN encoder learns representations for the entities in the graph by utilizing the graph  
 505 structure, node attributes and the associated relations while weighing different relations  
 506 differently and learning these weights during the training. The entity representations are

Dataset	Number of entities	Number of relations	Number of training facts	Number of evaluation facts	Number of test facts
FB15k	14,951	1,345	483,142	50,000	59,071
FB15k-237	14,541	237	272,115	17,535	20,466
WN18	40,943	18	141,442	5,000	5,000
WN18RR	40,943	11	86,835	3,034	3,134
Yago3-10	123,182	37	1,079,040	5,000	5,000

■ **Table 1** Details on the various KBs used for embedding evaluation.

507 given to the decoder, which is a combination of ConvE and TransE (based on ConvE but  
 508 having the translational property of TransE model) that performs better than the ConvE  
 509 model. The encoder and decoder are trained jointly to learn the entity representations and  
 510 score triples to verify and improve the representations.

511 **CompGCN** [59] generalizes previous GCN methods by jointly learning the representation  
 512 of the nodes and the relations in the multi-relational KGs while leveraging composition  
 513 functions from embedding approaches. CompGCN is able to scale well with the increasing  
 514 number of relations and outperforms several previous models including TransE, DistMult,  
 515 ComplEx, R-GCN and SACN.

## 516 3.4 Evaluation of Embedding Methods

### 517 3.4.1 Evaluation Protocol

518 **Evaluation.** Rule methods are typically evaluated under the *open world assumption*, i.e.,  
 519 any fact that is predicted is manually evaluated to see whether it holds in the real world  
 520 or not. Thus, even a fact that does not appear in the KB can be counted as correct. This  
 521 evaluation is obviously very labor-intensive, but it targets what rule mining is interested  
 522 in: the prediction of yet-unknown facts. KB embedding models, in contrast, are typically  
 523 evaluated under the closed world assumption: Given a KB, one removes a certain portion of  
 524 it to obtain a training KB. One then trains the embeddings on this reduced KB, and uses the  
 525 embeddings to predict facts. If these facts appear in the original KB, they count as correct,  
 526 otherwise they count as incorrect.

527 **Datasets.** Three very common KBs for evaluating embedding approaches are FB15k [7],  
 528 WN18 [6] and Yago3-10 [12]. FB15k and WN18 were both proposed by Bordes et al.  
 529 respectively in 2013 and 2014. FB15k is an extraction from Freebase where entities were  
 530 selected based on the number of citations in the original KB. WN18 is a subset of Wordnet  
 531 in which entities are *synsets*, that is semantic senses of words (selected on their popularity in  
 532 the KB) and predicates are lexical relations between those senses. Yago3-10 was proposed by  
 533 Dettmers et al. in 2018 as a subset of Yago3 [33] in which most of the facts describe people  
 534 (e.g., by citizenship, gender, and profession). The three KBs have been initially randomly  
 535 split into training, validation and test subsets and those splits always stay the same. Table 1  
 536 shows some statistics about these datasets.

### 537 3.4.2 Shortcomings of Benchmarks

538 While embedding models have gained popularity for the link prediction task and obtained  
 539 state-of-the-art results, several studies have recently taken a critical look at the performance  
 540 and evaluation aspects of these models. The benchmark datasets on which the embedding

541 models are trained have also been scrutinized. **Toutanova et al.** [57] were the first to  
542 find data leakage issues in the FB15k dataset. More precisely, the authors noted that, for  
543 certain relations  $r$ , the inverse relation  $r^-$  was also present in the data. This makes the  
544 prediction of a fact  $\langle x, r, y \rangle$  trivial if the fact  $\langle y, r^-, x \rangle$  is already there. As a remedy, the  
545 authors constructed the dataset FB15k-237 by removing the inverse triples and keeping only  
546 one relation out of the reverse relations. Dettmers et al. [12] similarly found issues with the  
547 WN18 dataset and created the WN18RR dataset. Table 1 shows the statistics about these  
548 datasets. With the introduction of these new datasets and their adoption for the evaluation  
549 of newer embedding models, it could be ensured that the models are not just learning trivial  
550 entailment, but learning to correctly predict non-trivial facts that require actual inference.  
551 However, most papers still showed the results for the evaluation of new models on both the  
552 old and new version of the datasets.

553 **Akrami et al.** [2] conducted a further detailed study questioning the performance of  
554 embedding models in the presence of data leakage and data redundancy. The study found  
555 a sizeable percentage of inverse, duplicate, and Cartesian product relations in the popular  
556 datasets FB15k, WNRR and Yago3-10. Duplicate relations are relations with different  
557 names that share the same facts (e.g., *hasCitizenship* and *hasNationality*). Cartesian product  
558 relations are relations that hold between all instances of a class (e.g., *sameSpeciesAs*). Such  
559 relations can be predicted trivially. Hence, the authors argued, the performance of these  
560 models would be significantly worse for link prediction on actual unseen data in realistic  
561 settings. Their experiments analysed various popular embeddings models including TransE,  
562 TransH, TransR, TransD, DistMult, ComplEx, ConvE, Tucker, and RotatE and showed  
563 substantial drops in performance with different datasets after removing the unrealistic triples,  
564 so much so that simple rule based techniques could achieve better accuracy than complex  
565 embedding techniques. The authors therefore strongly advocated the need to re-evaluate  
566 existing embedding approaches to find an effective solution for the link prediction task.

567 **Rossi et al.** [47] take a critical look at the properties of the entities in the benchmark datasets  
568 that are used to evaluate link prediction performance of embedding models. They focused  
569 on the Freebase and Wordnet datasets and performed a detailed experimental analysis of the  
570 features of these datasets and their limiting effect on the performance of embedding models.  
571 For instance, the authors showed that embedding models perform artificially better for the  
572 most frequent entities in the dataset. In FB15k, the entity *United States* appears in a lot of  
573 triples, and therefore, the TransE and DistMult models show better scores while predicting  
574 this entity as the missing entity. If the most frequent entities were removed from these  
575 datasets, the model performance (counter-intuitively) improved, indicating the over-fitting of  
576 the models on the most representative entities. Therefore, the authors advocated that better  
577 benchmarking practices and metrics are needed to determine the capability and fairness of  
578 the models.

579 **Pujara et al.** [42] performed an interesting study on the effect of sparsity and unreliable  
580 data on the performance of embeddings. Existing curated KGs like Wordnet and Freebase  
581 were modified in different experiments to introduce sparsity (in terms of relations or entities)  
582 and unreliable or corrupted triples, so that they resemble real-world KGs derived from text  
583 (such as NELL [8]). The authors found that performance is closely linked with sparsity,  
584 i.e. embeddings work well for relations and entities that have a dense representation and  
585 sparsity adversely affects their performance. Experiments showed that unreliable triples also  
586 degraded the performance. However, the authors made an interesting conclusion, namely  
587 that corrupted triples still improved embeddings marginally, therefore it is better to have a  
588 large noisy KG rather than a small set of very high quality triples.

589 These studies helped in bringing into focus the flaws that are inherent in all the popular  
590 benchmark datasets due to which global metrics for the evaluation of embedding models are  
591 proved to be insufficient and misleading. Thus, there is a need for careful and fine-grained  
592 evaluation of the performance of embedding models for their application in realistic use cases.

### 593 3.4.3 Shortcomings of the protocol

594 Several works have studied the shortcomings of the evaluation protocol for KB embed-  
595 dings. **Pezeshkpour et al.** [41] focused on the evaluation metrics and pointed out the  
596 need and importance of calibration of the embedding models before they can be deployed  
597 in real-world scenarios. For example, if the model says with 0.5 confidence that a triple  
598 is true, then the actual probability of the triples with this confidence should also be 0.5.  
599 In particular, they found that the model calibration as well as the ranking metrics were  
600 highly susceptible to the choice of negative sampling during training, with random re-  
601 placement of subject or object entity (*Random-N*) leading to worst results. In order to  
602 improve the evaluation techniques, the authors proposed the *CarefulN* method to select  
603 negative samples. Here, the highest scoring negative sample having an entity type which  
604 is different from the target entity type is selected as a negative sample. E.g. given a triple  
605  $\langle \text{Barack Obama}, \text{presidentOf}, \text{USA} \rangle$ , if *USA* is the target entity to be predicted, and the  
606 ranked list of predicted entities is  $(\text{USA}, \text{Hawaii}, \text{United Nations}, \text{Michelle Obama}, \dots)$ ,  
607 then we choose  $\langle \text{Barack Obama}, \text{presidentOf}, \text{Michelle Obama} \rangle$  as the negative sample since  
608 the type for *Michelle Obama* is different from *USA*. This technique explicitly ensures that the  
609 negative sample being generated is a true negative. Following this technique, they derived a  
610 new benchmark dataset Yago3-TC for evaluating KG completion that consists of both true  
611 and false facts for facilitating the correct measurement of triple classification accuracy.

612 **Sun et al.** [56] looked into the very specific issue of the recent neural-networks based  
613 embedding models showing inconsistent performance gains across different datasets such  
614 as FB15k-237 and WNRR18. They investigated in detail the models ConvKB [10] and  
615 CapsE [61] and found an unusual score distribution to be the reason for this discrepancy. For  
616 instance, many negatively sampled triples were given the same score as the correct triple. To  
617 break ties for the triples with the same score, they proposed a RANDOM evaluation protocol,  
618 i.e. if multiple triples have the same score, one of them is chosen randomly. Experiments  
619 demonstrated that recent deep models such as ConvKB, and CapsE were indeed affected by  
620 different evaluation protocols (unlike other models like ConvE and RotatE) and this could  
621 be detected with the proposed RANDOM protocol.

622 **Kadlec et al.** [26] were among the first to question the performance gains achieved by  
623 the newly proposed model architectures. The authors were able to perform suitable fine  
624 tuning the hyper-parameters for DistMult, one of the first embedding models proposed, and  
625 outperform several new models. This raised concerns on the performance gains by the newer  
626 models, advocating for a closer inspection of the training practices and objectives.

627 **Ruffinelli et al.** [48] re-implemented several existing models and performed extensive  
628 analysis of the performance of these models to compare them on a common ground. Going  
629 beyond previous works such as [36] (which studied the loss functions) and [29] (which looked  
630 into the negative sampling strategies), this paper performed a comprehensive and empirical  
631 evaluation of the effect of different training strategies and parameters such as the regularizer,  
632 optimizer and loss functions on a number of new and old embedding models. Their analysis  
633 indicated that the training parameters play a major role in the embedding performance.  
634 With a systematic fine tuning of these parameters, even the older model architectures such  
635 as RESCAL can match or outperform the recently introduced improved models. This work



636 makes a strong point of the need for re-assessing the individual benefits claimed by recent  
 637 and newer embedding models in light of the older models. The authors caution that the  
 638 performance gain reported by newer models could be mitigated by merely fine tuning of the  
 639 training strategies and therefore, this warrants close inspection.

640 **Jain et al.** [23] raised questions regarding the very semantic representation learned by  
 641 the embeddings models in the first place. They performed classification and clustering  
 642 experiments on the embeddings in order to analyse their semantic capability. The authors  
 643 challenge the common notion that entities having similar meaning (i.e. belonging to the  
 644 same class or type) such as *politicians*, *actors* etc. would be represented by similar vectors.  
 645 They constructed a dataset with entities belonging to different levels of the taxonomy  
 646 for Yago3-10 and DBpedia datasets, such as from *person* to *artist* to *painter*. Detailed  
 647 experiments demonstrated that both clustering and classification showed poor results for  
 648 entities having fine-grained classes. This means that embeddings are unable to capture the  
 649 semantics of entities beyond the top level classes (*person*, *organization*, *places* in Yago).  
 650 These surprising results indicated that though embeddings might show good performance on  
 651 the link prediction task, their utility for other semantic tasks such as reasoning etc. should  
 652 be carefully examined.

653 **Wang et al.** [64] inspected the evaluation protocol of the embedding techniques for the KB  
 654 completion task. They argue that the Entity Ranking (ER) protocol, where the missing head  
 655 or tail entity is predicted for a triple, is more suitable for evaluating a question answering  
 656 task, but not for the KB completion task. This is due to the fact that the context of the  
 657 missing information in terms of a head or tail entity would not be available when attempting  
 658 to find new missing triples of the form  $\langle ?, r, ? \rangle$ . With the ER protocol, the models may not  
 659 be penalized for ranking certain incorrect triples higher since they are not encountered at all.  
 660 The paper instead proposes a Pairwise Ranking (PR) protocol where all possible entity pairs  
 661 are considered and ranked with respect to a particular relation. Extensive experiments show  
 662 that popular embedding models provide worse performance with PR protocol than with the  
 663 ER protocol, even on seemingly easy datasets.

664 These studies have emphasized the need for better evaluation protocols and a critical  
 665 look at the training strategies of the embedding models for the task of KB completion in  
 666 realistic settings.

## 667 **4 Embedding Methods with Logical Components**

### 668 **4.1 Rationale**

669 Rule mining methods and embedding methods are complementary for the purpose of link  
 670 prediction:

- 671 ■ Rule mining produces patterns that can be understood by humans. Thus, their predictions  
 672 can be explained and justified.
- 673 ■ Rule mining can, in principle, work together with the schema of the KB, axioms, and  
 674 other types of logical constraints.
- 675 ■ Rule mining methods can deal with literals and numerical values, while embedding  
 676 methods typically project these away. Rule mining can find, e.g., that the death date is  
 677 always later than the birth date.
- 678 ■ Rule mining is typically evaluated under the open world assumption: it is explicitly  
 679 designed to predict facts that are not yet in the KB. Embedding methods, in turn, are  
 680 typically tuned to predict what is already there.

- 681 ■ On the flip-side, rule mining methods typically predict based on a single rule; it is harder  
 682 to predict facts with several rules that reinforce or contradict each other.
- 683 ■ Rule mining methods do not have a holistic view on an entity, with all its relations; they  
 684 are restricted to the relations that appear in the rules.
- 685 ■ Rule mining methods often generate rules with a low confidence, i.e., with a high rate of  
 686 false predictions.
- 687 For these reasons, several works have taken to combine symbolic and embedding methods.  
 688 Here, the symbolic component does not necessarily come from Rule Mining, but can also  
 689 come from the logical axioms of the ontology of the input KB. The existing approaches fall  
 690 into three classes:
- 691 ■ **Adding simple axioms.** Some approaches constrain the embeddings by simple ax-  
 692 ioms, which concern inverse, symmetric, or equivalent relations. We discuss them in  
 693 Sections 4.2 [13], 4.3 [15], and 4.4 [35].
- 694 ■ **Complex constraints.** Other approaches support more general and complex logical  
 695 constraints on the embeddings. We discuss them in Section 4.5 [23], 4.6 [11], and 4.7 [63].
- 696 ■ **Joint learning.** Finally, a number of approaches jointly embeds embeddings and confidences for  
 697 rules. We discuss them in Section 4.8 [19] and 4.9 [20, 71].

## 698 4.2 Improving Knowledge Graph Embeddings Using Simple Constraints

699 A first simple combination of logical rules and embeddings is presented by Ding et al. [13].  
 700 The authors focus on relation entailments, i.e., rules of the form  $\langle x, r, y \rangle \Rightarrow \langle x, r', y \rangle$  that can  
 701 also be denoted as  $r \Rightarrow r'$ . For example, if two people are married, then they also know each  
 702 other: *marriedTo*  $\Rightarrow$  *knows*. Such entailments can either be axioms from the ontology, or  
 703 they can be soft rules, i.e., rules with a confidence score that do not hold in all instantiations.  
 704 For example, a soft rule can be: If a person is born in a country, then the person probably  
 705 has the citizenship of that country. This is very often the case though not always. Such soft  
 706 rules can be mined by a rule mining system, and from now on, a set of such entailments is  
 707 assumed to be available.

708 If it is known that a relation  $r$  entails a relation  $r'$  and that  $\langle x, r, y \rangle$  holds for some entities  
 709  $x, y$ , then  $\langle x, r', y \rangle$  holds. Thus, the score that an embedding model gives to the fact  $\langle x, r, y \rangle$   
 710 should not be larger than the score it gives to  $\langle x, r', y \rangle$ :

$$711 \quad f(\langle x, r, y \rangle) \leq f(\langle x, r', y \rangle) \quad (1)$$

712 The authors enforce this condition on the ComplEx model (see Section 3) by imposing that  
 713 for a given entity  $x$ , all the real parts of the components of the embedding vector  $\vec{x} \in \mathbb{C}^d$   
 714 have to be non-negative, and all the imaginary parts have to be smaller than or equal to one.  
 715 Given an entity  $x$ ,  $d$  the embedding dimension, the constraints are formalized in Equation 2  
 716 where  $Re(\cdot)$  (resp.  $Im(\cdot)$ ) returns the real (resp. imaginary) part of a complex number and  
 717  $\vec{x}_i$  is the  $i^{th}$  component of the vector  $\vec{x}$ .

$$718 \quad \forall i \in \{1, \dots, d\} : Re(\vec{x}_i) \geq 0 \wedge Im(\vec{x}_i) \leq 1 \quad (2)$$

719 This constraint can be intuitively justified by seeing each component of  $\vec{x}$  as a feature, whose  
 720 value is zero if the feature does not apply to the entity  $x$ , and greater than 0 if the feature  
 721 applies to  $x$ , but never below zero. The constraint on the imaginary component serves as a  
 722 kind of normalization. With this non-negativity constraint, the desideratum of Equation 1  
 723 can be achieved by requiring:

$$724 \quad \forall i \in \{1, \dots, d\} : Re(\vec{r}_i) \leq Re(\vec{r}'_i) \wedge Im(\vec{r}_i) = Im(\vec{r}'_i) \quad (3)$$

725 If the entailment does not hold strictly, but only with a certain confidence, the condition can  
 726 be relaxed by introducing a real-valued confidence level  $\lambda$  and vector slack variables  $\vec{\alpha}, \vec{\beta}$ ,  
 727 which turn Equation 3 into

$$728 \quad \forall i \in \{1, \dots, d\} : \lambda \times (Re(\vec{r}_i) - Re(\vec{r}'_i)) \leq \vec{\alpha}_i, \lambda \times (Im(\vec{r}_i) - Im(\vec{r}'_i))^2 \leq \vec{\beta}_i \quad (4)$$

729 The larger the confidence level  $\lambda$ , and the smaller the slack variables  $\vec{\alpha}$  and  $\vec{\beta}$ , the more  
 730 Equation 4 resembles the hard constraint of Equation 3.

731 When these constraints are imposed on the ComplEx model, then the model is forced to  
 732 give a high score to facts that are logically entailed by other facts to which it gave a high  
 733 score. The authors then show that this improves the performance of link prediction over the  
 734 original model.

### 735 4.3 Improved Knowledge Graph Embedding Using Background 736 Taxonomic Information

737 Fatemi et al [15] introduce another way to improve knowledge graph embeddings, which uses  
 738 the taxonomy of the KB. For example, a knowledge base might contain the information that  
 739 *Emmanuel Macron* is a *president*, but it does not contain information that he is a *mammal*,  
 740 because it is implied by taxonomical knowledge. With this knowledge, if we know that  
 741 mammals are warm-blooded, we can conclude that *Emmanuel Macron* is warm-blooded as  
 742 well, without having explicit facts about this relation in the KG. Going one step further than  
 743 relation entailment, this work leverages the subsumption property of the relations as well as  
 744 the classes in KG. For example, the relation *presidentOf* is a sub-property of *managerOf*,  
 745 which in turn is a sub-property of *employedBy*. Formally, if a relation  $r_1$  is a sub-property of  
 746 a relation  $r_2$ , then  $\forall x, y : \langle x, r_1, y \rangle \Rightarrow \langle x, r_2, y \rangle$ . To represent class subsumption, the authors  
 747 model the entities as the characteristic functions of the class they belong to. This means that  
 748 if entity  $e$  is in class  $C$  i.e.  $\langle e, type, C \rangle$ , then the characteristic function between  $e$  and  $C$  is  
 749 true – written as  $\langle e, C, true \rangle$ . Hence, class subsumption can be expressed as a special case  
 750 of relation subsumption. For instance, if *president* is subclass of *mammal* in the taxonomy,  
 751 then  $\langle EmmanuelMacron, president, true \rangle \Rightarrow \langle EmmanuelMacron, mammal, true \rangle$ .

752 The proposed framework is a modification of the SimpleE [27] embedding model (see  
 753 Section 3.2), which makes use of these axioms. SimpleE considers two embeddings for each  
 754 relation: one embedding  $r^+$  for relation itself and another  $r^-$  for its inverse relation. Similarly,  
 755 there are two embeddings for each entity: one as a head entity  $e^+$ , and another as the tail  
 756 entity  $e^-$ . These embeddings are concatenated to obtain the final embedding for a relation  
 757 or entity. The proposed modification of this model is restricting the entity embeddings to be  
 758 element-wise non-negative.

759 In order to enforce the axiom that a relation  $r$  is a sub-relation of a relation  $s$  ( $\forall x, y :$   
 760  $\langle x, r, y \rangle \Rightarrow \langle x, s, y \rangle$ ), the model adds an equality constraint as  $\vec{r} = \vec{s} - \vec{\delta}_r$  where  $\vec{\delta}_r$  is a  
 761 non-negative vector, which expresses how  $r$  is different from  $s$ . This vector is learned during  
 762 training. With this, the function  $\mu$  (that maps embeddings to the probability of a triple)  
 763 obeys the constraint  $\mu\langle x, s, y \rangle \geq \mu\langle x, r, y \rangle$ .

764 Thus, the resulting *SimpleE*<sup>+</sup> model is able to enforce subsumption properties for entities  
 765 and relations and therefore, incorporate taxonomic knowledge in the embeddings to learn  
 766 more interpretable representation for words [37]. The experimental evaluation shows that  
 767 the proposed model is able to outperform traditional SimpleE for the KG completion task  
 768 and also has a faster convergence rate when taxonomic information is available.

#### 4.4 Regularizing Knowledge Graph Embeddings via Equivalence and Inversion Axioms

We have so far seen approaches that concentrate on subproperty axioms. We shall now look into two other types of axioms [35]: Given two relations  $r_1$  and  $r_2$ , an equivalence axiom  $r_1 \equiv r_2$  means that  $r_1$  and  $r_2$  are semantically equivalent though distinct in the KB (e.g., *part of* and *component of*). An inverse axiom  $r_2 \equiv \bar{r}_1$  means that  $r_1$  is the inverse predicate of  $r_2$  (e.g., *part of* and *has part*). The approach assumes that these axioms are defined in the ontology of the input KB.

Given the two sets of equivalence and inversion axioms, constraints are enforced in the training of the models. Let  $r_1 \equiv r_2$  be an equivalence (resp. inversion) axiom. This means that relations  $r_1$  and  $r_2$  are equivalent (resp. inverse) and then the scoring function  $f$  of an embedding model should verify  $f(\langle h, r, t \rangle) = f(\langle h, r_2, t \rangle)$  (resp.  $f(\langle h, r, t \rangle) = f(\langle t, r_2, h \rangle)$ ) given any entities  $h$  and  $t$ .

In the case of equivalence, this is simply implemented by forcing the embeddings of  $r_1$  and  $r_2$  to be the identical. In the case of an inversion axiom, the constraint has to be specified for each model in the form of a model-dependent function  $\Phi$  such that the constraint  $\bar{r}_2 = \Phi(r_1)$  results in  $f(\langle h, r_1, t \rangle) = f(\langle t, r_2, h \rangle)$ . For example, in the case of TransE [7], using  $\Phi : r_2 \mapsto -r_1$ , one gets  $f(\langle h, r_1, t \rangle) = \|\vec{h} + \vec{r}_1 - \vec{t}\| = \|-\vec{h} - \vec{r}_1 + \vec{t}\|$  (by homogeneity of the norm) and then  $f(\langle h, r_1, t \rangle) = \|\vec{t} + \Phi(\vec{r}_1) - \vec{h}\| = f(\langle t, r_2, h \rangle)$ . Note that  $\Phi$  needs to be an involution, i.e.,  $\forall r, \Phi(\Phi(r)) = r$ .

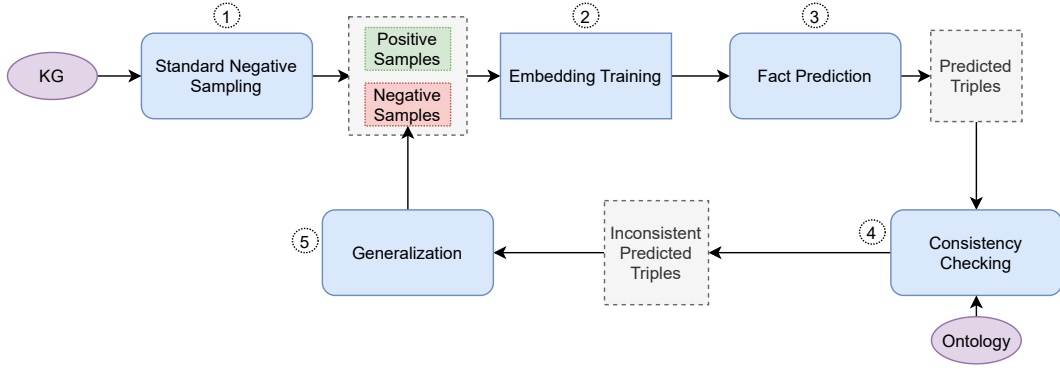
These constraints are called *hard constraints* because they entirely determine some embeddings. Another possibility is to use soft constraints in order to enforce axioms that are not entirely true. For example *married with* and *partner of* are not entirely semantic equivalents but their embeddings are similar to one another. Intuitively the objective of soft constraints is to nudge the model to adopt some desired properties rather than enforce hard-coded requirements. This is done by adding two weighted terms to the usual training loss:  $\hat{\mathcal{L}} = \mathcal{L} + \lambda [\sum_{r_1 \equiv r_2} \|\vec{r}_1 - \vec{r}_2\|_2^2 + \sum_{r_1 \equiv \bar{r}_2} \|\vec{r}_2 - \Phi(\vec{r}_1)\|_2^2]$  where  $\lambda$  is an hyper-parameter that needs to be determined during training.

#### 4.5 Improving Knowledge Graph Embeddings with Ontological Reasoning

Until now, we have concentrated mainly on very simple types of axioms to improve embeddings. ReasonKGE [24] is a method that can use complex constraints as well. The idea is to use symbolic reasoning to find predictions by the model that are logically inconsistent, and to feed these as negative samples into a retraining step. Traditionally, embedding methods generate negative triples by randomly replacing the head entity or tail entity in a triple from the KB (Section 2). This method, however, has two problems: First, as we have already discussed, it does not work as well for the head entities (Section 2.1). Furthermore, traditional methods do not necessarily create negative statements that violate domain and range constraints. For example, a triple such as  $\langle \text{Elvis}, \text{hasNationality}, \text{Priscilla} \rangle$  cannot be true since *hasNationality* requires a country as object. If such triples are not generated as negative examples, the model may produce them as predictions. Therefore, ReasonKGE sets out to generate negative examples by axioms – inspired by the NELL system [8], which also uses axioms for the generation of examples.

The framework of the proposed method is shown in Figure 3. The inputs of the framework are the KG and its ontology, whereas the outputs are negative samples, which can then be used for training the model in the next iteration. The first iteration simply generates the

## 4:20 Combining Embeddings and Rules for Fact Prediction



■ **Figure 3** ReasonKGE Framework

baseline model with a default sampling method. Here, traditional sampling methods are used to generate the negative facts, and the model is trained based on positive and negative facts to obtain the predictions. The predicted triples are checked for inconsistencies with respect to the underlying ontology with the help of a reasoner. The inconsistent triples are then generalized to other semantically similar triples which would also cause the same inconsistencies. Lastly, all the generated negative samples are fed back to the model for the next iteration of training. With each round of training, the model learns to identify inconsistencies and therefore make more consistent predictions.

The **consistency checking procedure**(step 4) is one of the main steps, that detects which predictions made by embedding model are inconsistent with the original KG ( $\mathcal{G}$ ) and ontology  $\mathcal{O}$ . For computational purposes, this check is done only on the subset of relevant facts. The *relevant set* is defined as follows:

$$Relv(\alpha, \mathcal{G}) = \{\alpha\} \cup \{\beta \in \mathcal{G} \mid Ent(\beta) \cap Ent(\alpha) \neq \emptyset\} \quad (5)$$

Here,  $\alpha$  is the predicted triple and  $\beta$  are triples in the KG. For example, consider  $\alpha = \langle Samsung, locatedIn, Emmanuel Macron \rangle$ . For this prediction the relevant set could consist of the following triples:

$$Relv(\alpha, \mathcal{G}) = \{ \langle Emmanuel Macon, livesIn, France \rangle, \\ \langle Emmanuel Macron, spouse, Brigitte Macron \rangle, \\ \langle Emmanuel Macron, type, person \rangle, \\ \langle Samsung, type, company \rangle \}$$

It is sufficient to perform consistency checking on  $Relv(\alpha, \mathcal{G}) \cup \mathcal{O}$ , instead of  $\alpha \cup \mathcal{G} \cup \mathcal{O}$ . In our example, a reasoner would flag this prediction as inconsistent, because the ontology tells us that *locatedIn* requires a location, and hence  $\langle Samsung, locatedIn, Emmanuel Macron \rangle$  implies that the type of *Emmanuel Macron* is *location*. That contradicts the fact  $\langle Emmanuel Macron, type, person \rangle$ , together with the axiom that states that people and locations are disjoint. Thus, this triple can serve as a negative sample.

Further, in Step 5, the negative samples obtained via consistency checking are fed to a **generalization** module to obtain multiple similar inconsistent facts that have a similar structure within KG. This is beneficial in 2 ways: firstly, by generating several negative samples that cause the same inconsistency, the model would be able to learn the inconsistency pattern and thus, the prediction of similar incorrect triples in next training iterations would

847 be avoided. Secondly, it enables us to obtain sufficient number of negative samples for a given  
 848 triple during the training of the model. The generalization of inconsistent predictions is done  
 849 in the following way: in an inconsistent predicted fact  $\langle h, r, t \rangle$ ,  $t$  can be replaced with another  
 850 entity  $k$  that has similar triples. For example, if  $\alpha = \langle Samsung, locatedIn, Emmanuel Macron \rangle$   
 851 is a predicted inconsistent triple, then we can take  $\alpha = \langle Samsung, locatedIn, Joe Biden \rangle$  as  
 852 another negative example if *Joe Biden* has the same neighbour triples as *Emmanuel Macron*,  
 853 i.e.  $\langle Joe Biden, type, person \rangle$ .

854 The authors show experimentally that *ReasonKGE* achieved better results on the link  
 855 prediction task as compared to traditional methods (TransE, ComplEx). Experiments  
 856 conducted on the Yago3-10 dataset were particularly significant, as the model achieved  
 857 more than 10% improvement for all the measures as compared to TransE. Additionally,  
 858 *ReasonKGE* reduced the ratio of inconsistent predictions over the test set when compared  
 859 to other models that employ static or random sampling techniques. A limitation of this  
 860 method is the use of DL-Lite [3] ontologies, due to which, theoretically, not all possible  
 861 similar negative samples will be obtained based on a given inconsistent prediction in the  
 862 generalization step.

## 863 4.6 Injecting Background Knowledge into Embedding Models for 864 Predictive Tasks on Knowledge Graphs

865 Similar to *ReasonKGE*, this paper proposes to improve KG embeddings by injecting available  
 866 background knowledge in the form of ontological axioms [11]. The authors propose *TransOWL*  
 867 and *TransROWL* models, as improved versions of the traditional embedding methods TransE  
 868 and TransR respectively.

869 The injection of background knowledge during the training phase involves two main  
 870 components - *reasoning* to add negative samples and *Background Knowledge (BK) injection*  
 871 to add constraints on the scoring function.

872 ■ During **reasoning**, negative samples are generated by leveraging the ontological properties  
 873 such as *domain*, *range*, *disjointWith*, *functionalProperty* with the help of the Apache Jena  
 874 framework. For example, if a particular entity type (or concept in ontology terminology),  
 875 let's say *location* is disjoint with another type e.g. *person*, then negative samples are  
 876 generated by replacing the person entity in a triple with all location entities present in the  
 877 KG. Thus, for a triple  $\langle Samsung, locatedIn, South Korea \rangle$ , a list of negative samples can  
 878 be generated by replacing *South Korea* with *Joe Biden*, *Barack Obama*, *John Smith*  
 879 and so on.

880 ■ During **BK injection**, ontological properties such as *equivalentClass*, *equivalentProp-*  
 881 *erty*, *inverseOf* and *subClassOf* are applied for the definition of additional constraints  
 882 on the scoring function such that resulting embedding vectors can reflect these prop-  
 883 erties. New triples corresponding to these properties are generated and added to the  
 884 training set of the model. For example, for the *equivalentClass* property, if class  $A$   
 885 is equivalent to class  $B$ , then for a triple  $\langle entity1, type, A \rangle$ , it is possible to generate  
 886 another triple  $\langle entity1, type, B \rangle$  as well. Similarly this is performed for other properties as  
 887 well and a considerable number of additional triples is generated before training the model.  
 888

889 The basic loss function for TransE is defined as

$$890 \sum_{\langle h,r,t \rangle \in \Delta, \langle h',r,t' \rangle \in \Delta'} [\gamma + f_r(t, h) - f_r(t', h')] \quad (6)$$

891 here  $\gamma \geq 0$  is the hyperparameter *margin*. For TransOWL, this loss function is more complex  
 892 due to the additional constraints from the axioms. For example, the addition of the the  
 893 *inverseOf* axiom would add a term to the loss function as

$$894 \sum_{\langle t,s,h \rangle \in \Delta, \langle t',s,h' \rangle \in \Delta'} [\gamma + f_s(t,h) - f_q(t',h')] \quad (7)$$

895 where  $f$  is the scoring function,  $\Delta$  refers to the set of additional triples generated by a reasoner  
 896 and  $s$  is the inverse relation of  $r$ . Similarly, the constraints are added in the loss function for  
 897 the other axioms as well. Experimental evaluation shows that the models generated through  
 898 this procedure show improvement for link prediction as well as triple classification in KGs as  
 899 compared to the original TransE and TransR models.

## 900 4.7 Knowledge Base Completion Using Embeddings and Rules

901 In [63], the authors propose to constrain knowledge graph embeddings by an altogether  
 902 different type of axioms: cardinality axioms. This is done by an Integer Linear Programming  
 903 problem: the objective function is computed using the scoring function of an embedding  
 904 model under the constraints from the symbolic axioms.

Let the  $\mathcal{E} = \{e_1, e_2, \dots, e_n\}$  and  $\mathcal{R} = \{r_1, r_2, \dots, r_m\}$  be the sets of entities and relations  
 in a KG at hand. The linear problem is defined with  $mn^2$  decision variables  $\{x_{i,j}^k, 1 \leq i, j \leq$   
 $n, 1 \leq k \leq m\}$  such that  $x_{i,j}^k$  indicates whether the fact  $\langle e_i, r_k, e_j \rangle$  is true or false. The weight  
 of a triple is computed using the scoring function  $f$  of an embedding model. This results in  
 an objective function of the form:

$$\max_{x_{i,j}^k} \sum_i \sum_j \sum_k f(\langle e_i, r_k, e_j \rangle) \cdot x_{i,j}^k$$

905 The constraints of this optimization problem are derived from four types of rules:

- 906 ■ **Type 1: noisy observation.** Observed triples are very likely to be true but KBs are  
 907 prone to noise. In order to take into account the rare cases in which an observed fact is  
 908 false, slack variables  $\epsilon_{i,j}^k$  are introduced for each observed triple and the R1 constraint is  
 909 added along with a penalization term in the objective function. This is a classical method  
 910 in linear programming, which allows the easy identification of noisy triples.
- 911 ■ **Type 2: argument type expectation.** Some predicate-specific type constraints should  
 912 be respected by the head and tail entities. This results in the R2 constraint in which  $\mathcal{S}_k$   
 913 (resp.  $\mathcal{O}_k$ ) contain the indexes of the entities that have the type of the head (resp. tail)  
 914 of the relation  $r_k$ .
- 915 ■ **Type 3: at-most-one restraint.** Some relations can handle at most one head per  
 916 tail (many-to-one) or one tail per head (one-to-many). For example, the relation *city-*  
 917 *LocatedInCountry* is a one-to-many relation meaning that a city can be located in at most  
 918 one country. Other relations are one-to-one. Those three types of relations result in three  
 919 constraints R3.1, R3.2 and R3.3 in which  $\mathcal{R}_{1-M}$ ,  $\mathcal{R}_{M-1}$  and  $\mathcal{R}_{1-1}$  are respectively the  
 920 sets of one-to-many, many-to-one and one-to-one relations.
- 921 ■ **Type 4: simple implication.** A relation  $r_1$  can imply another relation  $r_2$ , if  $\langle x, r_1, y \rangle \Rightarrow$   
 922  $\langle x, r_2, y \rangle$  for any entities  $x$  and  $y$ . It is denoted  $r_1 \Rightarrow r_2$ . This gives us the constraint R4.

923 With this, the final Integer Logic Program is:

$$\begin{aligned}
924 \quad & \max_{x_{i,j}^k} \sum_i \sum_j \sum_k f(\langle e_i, r_k, e_j \rangle) \cdot x_{i,j}^k \\
925 \quad & (R1) \ x_{i,j}^k + \epsilon_{i,j}^k = 1, \forall (i, j, k) : \langle e_i, r_k, e_k \rangle \text{ is observed} \\
926 \quad & (R2) \ x_{i,j}^k = 0, \forall k, \forall i : e_i \notin S_k, \forall j : e_j \notin O_k \\
927 \quad & (R3.1) \ \sum_i x_{i,j}^k \leq 1, \forall k : r_k \in \mathcal{R}_{1-M}, \forall j \\
928 \quad & (R3.2) \ \sum_j x_{i,j}^k \leq 1, \forall k : r_k \in \mathcal{R}_{M-1}, \forall i \\
929 \quad & (R3.3) \ \sum_i x_{i,j}^k \leq 1, \sum_j x_{i,j}^k \leq 1, \forall k : r_k \in \mathcal{R}_{1-1}, \forall i, \forall j \\
930 \quad & (R4) \ x_{i,j}^{k_1} \leq x_{i,j}^{k_2}, \forall k_1, k_2 \text{ s.t. } r_{k_1} \Rightarrow r_{k_2} \\
931 \quad & \text{where } x_{i,j}^k \in \{0, 1\}, \forall i, j, k; \ \epsilon_{i,j}^k \in \{0, 1\}, \forall (i, j, k) : \langle e_i, r_k, e_k \rangle \text{ is observed}
\end{aligned}$$

933 In spite of their promising results, the authors highlight two main limitations to this approach.  
934 First, constraints do not take into account the possible many-to-many relations, possibly  
935 missing out some ontology information. Second, solving the integer linear programming  
936 problem is time consuming and the approach then lacks scalability. In this regard, the  
937 authors propose a divide-and-conquer strategy for future work.

## 938 4.8 Jointly embedding KGs and Rules

939 So far, we have constrained embeddings by axioms. There are, however, also approaches that  
940 use soft rules instead of axioms, and that learn embeddings jointly with confidence scores  
941 for these soft rules. The first of these [19] improves the training procedure of the TransE  
942 model [7] by a new training loss that integrates both observed triples and groundings of some  
943 logical rules. The method focuses on rules of only two shapes:  $\forall x, y, \langle x, r_1, y \rangle \Rightarrow \langle x, r_2, y \rangle$   
944 and  $\forall x, y, \langle x, r_1, y \rangle \cap \langle y, r_2, z \rangle \Rightarrow \langle x, r_3, z \rangle$ , where  $r_1, r_2$  and  $r_3$  are relations from the graph.

Following Rocktäschel et al. [45], the truth value of a grounded rule is computed from the truth values of the constituent triples and t-norm logic principles. For this, the truth value of a single triple is first defined as:

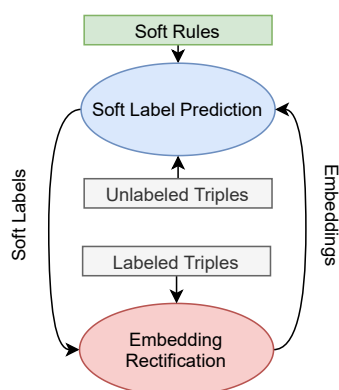
$$f(\langle x, r, y \rangle) = 1 - \frac{1}{3\sqrt{d}} \|\vec{x} + \vec{r} - \vec{y}\|$$

945 This is simply a normalization of the TransE [7] scoring function. To compute the truth  
946 value of more complicated logical formulae, this definition has to be broadened to negation,  
947 conjunction, and disjunction. The truth value of a negated triple  $\neg p$  is simply  $1 - f(p)$ .  
948 The truth value of a conjunction is given by a t-norm, i.e., a function that is commutative,  
949 associative, and monotonous, and that has 1 as the identity element. The work of [19] uses  
950 simply the product as the t-norm, i.e.,  $f(p \wedge q) = f(p) \times f(q)$ . With this, the truth value  $f$   
951 of an implication is

$$\begin{aligned}
952 \quad & f(p \Rightarrow q) = f(\neg p \vee q) = f(\neg(p \wedge \neg q)) \\
953 \quad & = 1 - f(p) \times (1 - f(q)) \\
954 \quad & = 1 - f(p) + f(p) \times f(q)
\end{aligned}$$

956 The only question left is how to generate the logical rules that are taken as input of this  
957 improved training procedure. A natural method could be to run a logical approach such





■ Figure 4 RUGE

958 as AMIE or RuDiK [17, 40]. The authors of [19] have a different approach that uses their  
 959 method of scoring rule groundings to select the best ranking rules in a greedy manner.

#### 960 4.9 Knowledge Graph Embedding with Iterative Guidance from Soft 961 Rules (RUGE)

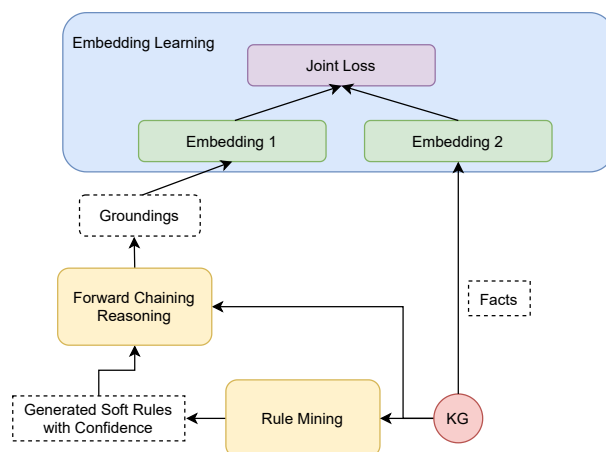
962 The Rule-Guided Embedding (RUGE) algorithm [20] is another method that learns embed-  
 963 dings jointly with confidence scores for logical rules. Its main steps are shown in Figure 4.  
 964 The system starts out with soft rules (top of the figure), mined by the AMIE system [17].  
 965 These are instantiated to make predictions (see Section 2.1) – each with a confidence. The  
 966 Embedding Step (bottom of the figure) takes as input the predictions of the rules as well as  
 967 labeled triples from the KB. The embedding is trained on these two sources. This allows  
 968 the prediction of new facts, which will in turn predict new facts by help of the rules. This  
 969 process is iterated, thereby amplifying automatically the number of labeled examples.

970 The rule mining system gives each rule a confidence. However, this confidence concerns  
 971 the rule as a whole, not an individual grounded variant of the rule, where all variables are  
 972 instantiated. To compute the confidence of an individual grounded rule, the approach uses  
 973 the scores  $\phi(\cdot)$  of the embeddings of the facts that appear in the rule, as well as the score  
 974  $s(\cdot)$  of the fact that the rule predicts, and proceeds according to the definitions of t-norm  
 975 based fuzzy logics, in much the same way as [19].

976 The approach then aims to find a scoring function  $s(\cdot)$  that is as close as possible to  
 977 the current scoring function  $\phi(\cdot)$ , while at the same time making the confidences  $\pi(\cdot)$  of  
 978 all grounded rules as close as possible to 1 (the maximum). This is done by solving an  
 979 optimization problem. This yields scores  $s(\cdot)$  for facts that are predicted by the rules.

980 In the second step, the approach then corrects its embeddings  $\phi(\cdot)$  so that they mirror (1)  
 981 the truth value of facts that appear already in the KB and (2) the score  $s(\cdot)$  for facts that  
 982 do not appear in the KB, but were predicted by the rules. This updated embedding is then  
 983 fed again into the rules, and the process is iterated. Experiments show that this method  
 984 achieves significant improvements in link prediction task on Freebase and YAGO.

985 A variant of this approach is the SoLE system [71] (“Soft Logical rules enhanced Embed-  
 986 dings”), whose architecture is shown in Figure 5. Like RUGE, SoLE takes as input a KB  
 987 and rules. It uses the rules to predict new facts in an iterative manner until no more facts  
 988 can be predicted (a technique called *forward chaining*). The rules are then grounded, and a  
 989 confidence score is computed for each grounded rule, not unlike this is done in RUGE as well.



■ **Figure 5** SoLE Architecture (stage 1 in yellow, stage 2 in blue)

990 Different from RUGE, SoLE then minimizes a joint loss so as to find embeddings that can  
 991 (1) predict the labels of triples contained the KB, while also (2) imitating the confidences of  
 992 rules.

## 993 5 Rule Mining with embedding techniques

994 In the previous section, we have discussed several embedding methods that use logical  
 995 techniques to improve their performance. The other direction is much less common: there are  
 996 few methods that use embedding models in order to improve logical rule mining techniques.  
 997 We will now present the most prominent ones.

### 998 5.1 ILP Rule Mining

999 A first small application of embeddings for rule mining is presented in an extension of  
 1000 RuDiK [40] by Ahmadi et al. [1]. The new system can also mine rules about class membership,  
 1001 such as “Politicians are not married to officeholders of a different party”:

$$\begin{aligned}
 1002 \quad & [\langle x, party, x_p \rangle \wedge \langle y, party, y_p \rangle \wedge x_p \neq y_p \\
 1003 \quad & \wedge \langle x, type, Politician \rangle \wedge \langle y, type, Office Holder \rangle] \Rightarrow \neg \langle x, spouse, y \rangle \\
 1004
 \end{aligned}$$

1005 The question is now what classes should be considered in such rules. Considering all classes  
 1006 may lead to rules that are too fine-grained. It would also be inefficient. Using only the  
 1007 top-level classes, in contrast, may miss out on useful rules that hold in a subclass.

1008 RuDiK therefore clusters the instances of the KB. The method of choice here are entity  
 1009 embedding methods. The authors observe that the clusters obtained this way are more  
 1010 uniform in what concerns the structural similarity of entities (i.e., the outgoing relations that  
 1011 they share) than class membership. This is because two entities with different relations can  
 1012 belong to the same class, and entities with the same relations can belong to different classes.  
 1013 The embedding, in contrast, groups entities by their relations, which is more amenable to  
 1014 the rule mining.

1015 It turns out that entities with popular classes, such as *Person*, can be spread across  
 1016 multiple clusters, but classes with finer granularity, such as *Politician* and *OfficeHolder* are  
 1017 grouped together. For each cluster, RuDiK then determines a class (e.g., the class that most

1018 entities in the cluster belong to). This class is then used for mining rules such as the one  
1019 above.

## 1020 5.2 Few-shot learning for label propagation

1021 As stated in Section 2.1, a recurrent problem when working on KBs is the lack of negative  
1022 statements. That makes it difficult to classify a prediction of any model. In an ideal situation,  
1023 an operator would be available during training in order to manually tag generated facts as  
1024 positive or negative. This is rarely the case because it is very costly but it could be very  
1025 useful in the generation of false statements for example. This problem of manually tagging  
1026 samples (here triples) is not specific to KB processing and it has given birth to a field of  
1027 research called few-shot learning. This is the study of learning algorithms that work on a  
1028 very small number of samples. It often applies in fields where the creation of supervision labels  
1029 is costly, for example computer vision.

1030 In [32] the authors propose a few-shot rule-based knowledge validation framework that  
1031 uses an embedding model (HypER [5]) in order to propagate the decisions of a human  
1032 operator to whom triples to tag are submitted. The goal of the method is to enrich the KB  
1033 with positive and negative examples that allow a better evaluation of a set of rules. The  
1034 proposed *propagation* method relies on a measure of similarity between facts. To compute  
1035 the similarity, a vector representing each triple is computed by concatenating the embeddings  
1036 of the entities. The propagation of manual labels is done locally to triples sharing the same  
1037 relation, and so their embedding is omitted in the concatenation. For example, let's say that  
1038 an operator labeled the fact  $\langle \text{Barack Obama}, \text{marriedTo}, \text{Sasha Obama} \rangle$  as false, this label  
1039 is going to be propagated to triples involving *Barack Obama* and *MarriedTo* or *MarriedTo*  
1040 and *Sasha Obama* that are similar enough to the initial one. Eventually the set of manually  
1041 labeled triples along with the automatically labeled ones improve the evaluation process of  
1042 the rules. The authors apply their method to rules mined with AMIE [17] and RuDiK [40].  
1043 The proposed method uses HypER as embedding model but the authors insist on the fact  
1044 that any model can be used for this task.

## 1045 5.3 Approximate algorithms

1046 AMIE [17] is an exhaustive rule mining system, i.e., it finds all rules above user-specified  
1047 confidence and support thresholds. This makes AMIE quite heavy to run on large knowledge  
1048 bases. AMIE+ [18] improved the runtime by approximating the computation of the confidence  
1049 value of rules. This comes at a minor cost in the precision of the algorithm but allows  
1050 reducing the computation time by several orders of magnitude.

1051 Another way to speed up the rule mining is by *sampling*. The underlying intuition is  
1052 that a rule of the form  $\langle e, r_1, e_1 \rangle \wedge \langle e_1, r_2, e_2 \rangle \wedge \dots \wedge \langle e_n, r_{n+1}, e' \rangle \Rightarrow \langle e, r, e' \rangle$  can be seen as  
1053 the co-occurrence in the knowledge graph (KG) of two paths from  $e$  to  $e'$ : one of length 1  
1054 (passing through the relation  $r$ ), and one of length  $2n + 1$  (through the entities  $e_1, e_2, \dots,$   
1055  $e_n$  and the relations  $r_1, r_2, \dots, r_{n+1}$ ). Exploring the possible rules then comes down to  
1056 finding possible paths from one entity to another. This graph exploration is computationally  
1057 expensive, and so the authors of [39] propose a two-step acceleration of the graph exploration  
1058 and of the evaluation of the rules.

1059 ■ First, the size of the graph is reduced by *sampling* the KG. Given a relation  $r$  that should  
1060 appear in the head atom of the rule and a maximum length  $l \geq 2$ , the neighborhood of  $r$   
1061 is computed iteratively. We start from a set  $E_0$ , which includes any entity involved in a  
1062 fact with  $r$ . We then compute  $E_i$  for  $1 \leq i \leq l - 2$  by including entities linked to some

entity of  $E_{i-1}$  by any predicate. The neighborhood of  $r$  is then defined as  $\mathcal{N}(r) = \cup_{i=0}^{l-2} E_i$  and it includes all entities relevant to find paths of maximum length  $l$  and then rules involving  $l$  atoms in the body and  $p$  in the head.

Subsequently, instead of exhaustively exploring all the possible paths in the neighborhood of the relation  $r$ , the authors suggest to use a bilinear embedding model to learn matrix representations of relations (see Section 3.2). A relation path  $r_1, r_2, \dots, r_l$  in the graph can then be represented as the product of the matrices of the relations  $M_{r_1} \cdot M_{r_2} \cdot \dots \cdot M_{r_l}$ . The similarity between the path (corresponding to the body of the potential rule) and  $r$  is computed using the matrix Frobenius norm  $\text{sim}(r, [r_1, r_2, \dots, r_l]) = \exp(-\|M_r - M_{r_1} \cdot M_{r_2} \cdot \dots \cdot M_{r_l}\|_F)$ .

The authors compare their approach to AMIE+, and show that the new approach mines more rules, and rules of better quality in terms of confidence. Furthermore, the process is much faster for rules that have the shape of paths.

## 6 Conclusion

Knowledge Bases (KBs) find many uses in AI applications, such as personal assistants, question answering systems, or text analysis. And yet, KBs are usually incomplete and miss facts. Two avenues of research have taken to predict missing facts: a symbolic one, based on rule mining, and a neural one, based on embeddings. Each of them has their respective strengths, and in this article we have presented an overview of both. We have also discussed recent studies on the criticism of the benchmark and protocols used during evaluation of embedding models. We have then presented approaches that successfully combine both symbolic and neural methods to perform fact prediction in KBs. While there are several approaches that use rules in order to improve embeddings, there are rather few approaches that use embeddings to improve rule mining. This may thus be an interesting direction for further research.

## References

- 1 Naser Ahmadi, Viet-Phi Huynh, Vamsi Meduri, Stefano Ortona, and Paolo Papotti. Mining expressive rules in knowledge graphs. *Journal of Data and Information Quality (JDIQ)*, 12(2):1–27, 2020.
- 2 Farahnaz Akrami, Mohammed Samiul Saeef, Qingheng Zhang, Wei Hu, and Chengkai Li. Realistic re-evaluation of knowledge graph completion methods: An experimental study. In *ACM SIGMOD*, 2020.
- 3 Alessandro Artale, Diego Calvanese, Roman Kontchakov, and Michael Zakharyashev. The dl-lite family and relations. *Journal of artificial intelligence research*, 36:1–69, 2009.
- 4 Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. Dbpedia: A nucleus for a web of open data. In *ISWC*, 2007.
- 5 Ivana Balažević, Carl Allen, and Timothy M Hospedales. Hypernetwork knowledge graph embeddings. In *ICANN*, 2019.
- 6 Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94(2):233–259, 2014.
- 7 Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *NeurIPS*, 2013.
- 8 Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka, and Tom M Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, 2010.

- 1108 9 Yuanfei Dai, Shiping Wang, Neal N Xiong, and Wenzhong Guo. A survey on knowledge graph  
1109 embedding: Approaches, applications and benchmarks. *Electronics*, 9(5):750, 2020.
- 1110 10 Tu Dinh Nguyen Dai Quoc Nguyen, Dat Quoc Nguyen, and Dinh Phung. A novel embedding  
1111 model for knowledge base completion based on convolutional neural network. In *NAACL*,  
1112 2018.
- 1113 11 Claudia d’Amato, Nicola Flavio Quatraro, and Nicola Fanizzi. Injecting background knowledge  
1114 into embedding models for predictive tasks on knowledge graphs. In *ESWC*, 2021.
- 1115 12 Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional  
1116 2d knowledge graph embeddings. In *AAAI*, 2018.
- 1117 13 Boyang Ding, Quan Wang, Bin Wang, and Li Guo. Improving knowledge graph embedding  
1118 using simple constraints. *arXiv preprint arXiv:1805.02408*, 2018.
- 1119 14 Xin Dong, Evgeniy Gabrilovich, Jeremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas  
1120 Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: A web-scale approach to  
1121 probabilistic knowledge fusion. In *ACM SIGKDD*, 2014.
- 1122 15 Bahare Fatemi, Siamak Ravanbakhsh, and David Poole. Improved knowledge graph embedding  
1123 using background taxonomic information. In *AAAI*, volume 33, 2019.
- 1124 16 Jun Feng, Minlie Huang, Mingdong Wang, Mantong Zhou, Yu Hao, and Xiaoyan Zhu.  
1125 Knowledge graph embedding by flexible translation. In *KR*, 2016.
- 1126 17 Luis Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. Amie: Association  
1127 rule mining under incomplete evidence in ontological knowledge bases. In *WWW*, 2013.
- 1128 18 Luis Galárraga, Christina Teflioudi, Katja Hose, and Fabian M. Suchanek. Fast Rule Mining  
1129 in Ontological Knowledge Bases with AMIE+. In *VLDBJ*, 2015.
- 1130 19 Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. Jointly embedding knowledge  
1131 graphs and logical rules. In *EMNLP*, 2016.
- 1132 20 Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. Knowledge graph embedding  
1133 with iterative guidance from soft rules. In *AAAI*, 2018.
- 1134 21 Shizhu He, Kang Liu, Guoliang Ji, and Jun Zhao. Learning to represent knowledge graphs  
1135 with gaussian embedding. In *CIKM*, 2015.
- 1136 22 Frank Lauren Hitchcock. The expression of a tensor or a polyadic as a sum of products.  
1137 *Journal of Mathematics and Physics*, 6:164–189, 1927.
- 1138 23 Nitisha Jain, Jan-Christoph Kalo, Wolf-Tilo Balke, and Ralf Krestel. Do embeddings actually  
1139 capture knowledge graph semantics? In *ESWC*, 2021.
- 1140 24 Nitisha Jain, Trung-Kien Tran, Mohamed H Gad-Elrab, and Daria Stepanova. Improving  
1141 knowledge graph embeddings with ontological reasoning. In *ISWC*, 2021.
- 1142 25 Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. Knowledge graph embedding  
1143 via dynamic mapping matrix. In *ACL*, 2015.
- 1144 26 Rudolf Kadlec, Ondřej Bajgar, and Jan Kleindienst. Knowledge base completion: Baselines  
1145 strike back. In *Repl4NLP*, 2017.
- 1146 27 Seyed Mehran Kazemi and David Poole. Simple embedding for link prediction in knowledge  
1147 graphs. In *NeurIPS*, 2018.
- 1148 28 Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional  
1149 networks. *arXiv preprint arXiv:1609.02907*, 2016.
- 1150 29 Bhushan Kotnis and Vivi Nastase. Analysis of the impact of negative sampling on link  
1151 prediction in knowledge graphs. *arXiv preprint arXiv:1708.06816*, 2017.
- 1152 30 Jonathan Lajus, Luis Galárraga, and Fabian M. Suchanek. Fast and Exact Rule Mining with  
1153 AMIE 3. In *ESWC*, 2020.
- 1154 31 Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation  
1155 embeddings for knowledge graph completion. In *AAAI*, 2015.
- 1156 32 Michael Loster, Davide Mottin, Paolo Papotti, Jan Ehmüller, Benjamin Feldmann, and Felix  
1157 Naumann. Few-shot knowledge validation using rules. In *TheWebConf*, 2021.
- 1158 33 Farzaneh Mahdisoltani, Joanna Asia Biega, and Fabian M. Suchanek. YAGO3: A Knowledge  
1159 Base from Multilingual Wikipedias. In *CIDR*, 2015.

- 1160 34 Christian Meilicke, Melisachew Wudage Chekol, Daniel Ruffinelli, and Heiner Stuckenschmidt.  
1161 An introduction to anyburl. In *KI*, 2019.
- 1162 35 Pasquale Minervini, Luca Costabello, Emir Muñoz, Vít Nováček, and Pierre-Yves Vanden-  
1163 bussche. Regularizing knowledge graph embeddings via equivalence and inversion axioms. In  
1164 *ECML PKDD*, 2017.
- 1165 36 Sameh K Mohamed, Vít Nováček, Pierre-Yves Vandenbussche, and Emir Muñoz. Loss functions  
1166 in knowledge graph embedding models. *DL4KG@ ESWC*, 2377:1–10, 2019.
- 1167 37 Brian Murphy, Partha Talukdar, and Tom Mitchell. Learning effective and interpretable  
1168 semantic models using non-negative sparse embedding. In *COLING*, 2012.
- 1169 38 Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective  
1170 learning on multi-relational data. In *ICML*, 2011.
- 1171 39 Pouya Ghiasnezhad Omran, Kewen Wang, and Zhe Wang. An embedding-based approach to  
1172 rule learning in knowledge graphs. *IEEE Transactions on Knowledge and Data Engineering*,  
1173 33(4):1348–1359, 2021.
- 1174 40 Stefano Ortona, Venkata Vamsikrishna Meduri, and Paolo Papotti. Robust discovery of  
1175 positive and negative rules in knowledge bases. In *ICDE*, 2018.
- 1176 41 Pouya Pezeshkpour, Yifan Tian, and Sameer Singh. Revisiting evaluation of knowledge base  
1177 completion models. In *AKBC*, 2020.
- 1178 42 Jay Pujara, Eriq Augustine, and Lise Getoor. Sparsity and noise: Where knowledge graph  
1179 embeddings fall short. In *EMNLP*, 2017.
- 1180 43 Simon Razniewski, Hiba Arnaout, Shrestha Ghosh, and Fabian M. Suchanek. Completeness,  
1181 Recall, and Negation in Open-World Knowledge Bases. In *VLDB*, 2021.
- 1182 44 Simon Razniewski, Fabian M. Suchanek, and Werner Nutt. But What Do We Actually Know?  
1183 In *AKBC workshop*, 2016.
- 1184 45 Tim Rocktäschel, Sameer Singh, and Sebastian Riedel. Injecting logical background knowledge  
1185 into embeddings for relation extraction. In *NAACL*, 2015.
- 1186 46 Andrea Rossi, Denilson Barbosa, Donatella Firmani, Antonio Matinata, and Paolo Merialdo.  
1187 Knowledge graph embedding for link prediction: A comparative analysis. *ACM Transactions*  
1188 *on Knowledge Discovery from Data (TKDD)*, 15(2):1–49, 2021.
- 1189 47 Andrea Rossi and Antonio Matinata. Knowledge graph embeddings: Are relation-learning  
1190 models learning relations? In *EDBT/ICDT*, 2020.
- 1191 48 Daniel Ruffinelli, Samuel Broscheit, and Rainer Gemulla. You can teach an old dog new tricks!  
1192 on training knowledge graph embeddings. In *ICLR*, 2019.
- 1193 49 Ali Sadeghian, Mohammadreza Armandpour, Patrick Ding, and Daisy Zhe Wang. Drum:  
1194 End-to-end differentiable rule mining on knowledge graphs. In *NeurIPS*, 2019.
- 1195 50 Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and  
1196 Max Welling. Modeling relational data with graph convolutional networks. In *ESWC*, 2018.
- 1197 51 Chao Shang, Yun Tang, Jing Huang, Jinbo Bi, Xiaodong He, and Bowen Zhou. End-to-end  
1198 structure-aware convolutional networks for knowledge base completion. In *AAAI*, 2019.
- 1199 52 Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural  
1200 tensor networks for knowledge base completion. In *NeurIPS*, 2013.
- 1201 53 Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago - A Core of Semantic  
1202 Knowledge . In *WWW*, 2007.
- 1203 54 Fabian M. Suchanek, Jonathan Lajus, Armand Boschin, and Gerhard Weikum. Knowledge  
1204 Representation and Rule Mining in Entity-Centric Knowledge Bases. In *RW*, 2019.
- 1205 55 Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph  
1206 embedding by relational rotation in complex space. In *ICLR*, 2018.
- 1207 56 Zhiqing Sun, Shikhar Vashishth, Soumya Sanyal, Partha Talukdar, and Yiming Yang. A  
1208 re-evaluation of knowledge graph completion methods. In *ACL*, 2020.
- 1209 57 Kristina Toutanova and Danqi Chen. Observed versus latent features for knowledge base and  
1210 text inference. In *CVSC workshop*, 2015.

- 1211 **58** Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard.  
1212 Complex embeddings for simple link prediction. In *ICML*, 2016.
- 1213 **59** Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. Composition-based  
1214 multi-relational graph convolutional networks. In *ICLR*, 2019.
- 1215 **60** Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase.  
1216 *Commun. ACM*, 57(10):78–85, 2014.
- 1217 **61** Thanh Vu, Tu Dinh Nguyen, Dat Quoc Nguyen, Dinh Phung, et al. A capsule network-based  
1218 embedding model for knowledge graph completion and search personalization. In *NAACL*,  
1219 2019.
- 1220 **62** Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey  
1221 of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*,  
1222 29(12):2724–2743, 2017.
- 1223 **63** Quan Wang, Bin Wang, and Li Guo. Knowledge base completion using embeddings and rules.  
1224 In *ICAOI*, 2015.
- 1225 **64** Yanjie Wang, Daniel Ruffinelli, Rainer Gemulla, Samuel Broscheit, and Christian Meilicke.  
1226 On evaluating embedding models for knowledge base completion. In *Repl4NLP*, 2019.
- 1227 **65** Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by  
1228 translating on hyperplanes. In *AAAI*, 2014.
- 1229 **66** Gerhard Weikum, Luna Dong, Simon Razniewski, and Fabian M. Suchanek. Machine Know-  
1230 ledge: Creation and Curation of Comprehensive Knowledge Bases. In *Foundations and Trends*  
1231 *in Databases*, 2021.
- 1232 **67** Alfred North Whitehead and Bertrand Russell. *Principia mathematica*. Cambridge University  
1233 Press, 1913.
- 1234 **68** Han Xiao, Minlie Huang, Yu Hao, and Xiaoyan Zhu. Transg: A generative mixture model for  
1235 knowledge graph embedding. *arXiv preprint arXiv:1509.05488*, 2015.
- 1236 **69** Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and  
1237 relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*, 2014.
- 1238 **70** Rui Ye, Xin Li, Yujie Fang, Hongyu Zang, and Mingzhong Wang. A vectorized relational  
1239 graph convolutional network for multi-relational network alignment. In *IJCAI*, 2019.
- 1240 **71** Jindou Zhang and Jing Li. Enhanced knowledge graph embedding by jointly learning soft  
1241 rules and facts. *Algorithms*, 12(12):265, 2019.
- 1242 **72** Zhanqiu Zhang, Jianyu Cai, Yongdong Zhang, and Jie Wang. Learning hierarchy-aware  
1243 knowledge graph embeddings for link prediction. In *AAAI*, 2020.
- 1244 **73** Xiang Zhao, Weixin Zeng, Jiuyang Tang, Wei Wang, and Fabian M. Suchanek. An Experimental  
1245 Study of State-of-the-Art Entity Alignment Approaches . In *TKDE*, 2020.