Arthur Colombini Gusmão

Interpreting Embedding Models of Knowledge Bases

São Paulo

2018

Arthur Colombini Gusmão

Interpreting Embedding Models of Knowledge Bases

Master thesis presented to the Escola Politécnica da Universidade de São Paulo to obtain the Master of Science degree.

Concentration Area:
Computer Engineering

Supervisor:
Fabio Gagliardi Cozman

São Paulo

2018

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, _____ de _____ de _____

Assinatura do autor: _____

Assinatura do orientador: _____

Catalogação-na-publicação

Gusmão, Arthur Colombini
Interpreting Embedding Models of Knowledge Bases / A. C. Gusmão -- versão corr. -- São Paulo, 2018.
80 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.Explainable AI 2.Interpretability 3.Relational Machine Learning 4.Embedding Models 5.Knowledge Bases I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

Acknowledgements

This work could not have been accomplished without the great support that I have received from so many people over the years. I wish to offer my most heartfelt thanks to the following people:

To my advisor, Fabio Gagliardi Cozman. Thank you for being present every single moment that I needed it, without measuring efforts to help me with my projects and to foster my learning.

To my parents, Janete Colombini Gusmão and João Gusmão Filho. Thank you for your support from day 1, not only in this but in all endeavors I have put myself in my life. You have shown me what unconditional love is.

To Glauber de Bona, Denis Deratani Mauá, and Bernardo Nunes Gonçalves. Thank you for all the useful discussions and rich contributions during the whole development of this work.

To Anna Reali Costa. Thank you for keeping your office door open and for having put me in contact with my advisor.

To Alvaro Henrique Chaim Correia and Andrey Ruschel. Thank you for having collaborated so much with this research, for the times when we had to stay working until late at night on weekends and you stayed motivated and by my side. You have been two of my closest colleagues and closest friends in the entire course.

To Francisco Henrique Otte Vieira de Faria. Thank you for giving me the privilege to collaborate with your research and to participate in several prestigious publications and awards. Also, for so many moments that we shared together during my first year of the course, for the great (pragmatic and philosophical) discussions about our field of work, about life, and about the world, which made us develop a good friendship.

To Luiz Henrique Barbosa Mormille. Thank you for being such a good friend and colleague, and for helping me throughout several subjects during our Master's program.

To my colleagues Rodrigo Monteiro de Aquino and João Marcos Correia Marques that have directly contributed with your knowledge and discussions in stages of this research, and to all the other colleagues that I had the privilege to know throughout the course.

To IBM and the São Paulo Research Foundation (FAPESP), for supporting this research (grant 2017/19007-6), and to CNPq and the Programa de Pós Graduação em Engenharia Elétrica da Escola Politécnica da Universidade de São Paulo, for providing me financial support during the first seven months of my Master's course.

The text has disappeared under the interpretation.

Friedrich Nietzsche

Abstract

Knowledge bases are employed in a variety of applications, from natural language processing to semantic web search; alas, in practice, their usefulness is hurt by their incompleteness. To address this issue, several techniques aim at performing knowledge base completion, of which embedding models are efficient, attain state-of-the-art accuracy, and eliminate the need for feature engineering. However, embedding models predictions are notoriously hard to interpret. In this work, we propose model-agnostic methods that allow one to interpret embedding models by extracting weighted Horn rules from them. More specifically, we show how the so-called "pedagogical techniques", from the literature on neural networks, can be adapted to take into account the large-scale relational aspects of knowledge bases, and show experimentally their strengths and weaknesses.

Resumo

Bases de conhecimento apresentam diversas aplicações, desde processamento de linguagem natural a pesquisa semântica da web; contudo, na prática, sua utilidade é prejudicada por não serem totalmente completas. Para solucionar esse problema, diversas técnicas focam em completar bases de conhecimento, das quais modelos de *embedding* são eficientes, atingem estado da arte em acurácia, e eliminam a necessidade de fazer-se engenharia de características dos dados de entrada. Entretanto, as predições dos modelos de *embedding* são notoriamente difíceis de serem interpretadas. Neste trabalho, propomos métodos agnósticos a modelo que permitem interpretar modelos de *embedding* através da extração de regras *Horn* ponderadas por pesos dos mesmos. Mais especificamente, mostramos como os chamados "métodos pedagógicos", da literatura de redes neurais, podem ser adaptados para lidar com os aspectos relacionais e de larga escala de bases de conhecimento, e mostramos experimentalmente seus pontos fortes e fracos.

List of Figures

List of Tables

Contents

1 Introduction

Recently, a large number of graph structured *knowledge bases* (KBs) have been created, such as NELL (MITCHELL, 2010), Freebase (BOLLACKER et al., 2008), and WordNet (MILLER, 1995). Graph structured KBs, also called *knowledge graphs* (KGs), contain structured data in the form of entities and relations that can be encoded as a graph, with nodes representing entities and edges representing relationships between the respective entities. KB applications are wide, ranging from natural language processing to semantic web search (SCHUHMACHER; PONZETTO, 2014; CUCERZAN, 2007). However, a common problem when employing them to such tasks is that even the largest KBs are incomplete, in the sense that they do not contain all facts that are true. In this context, one can apply *Statistical Relational Learning* (SRL) techniques to learn models that are able to predict unexisting relationships, thus *completing* the KBs, a task that is often called *KB completion* (NICKEL et al., 2015).

In contrast to traditional machine learning methods, SRL techniques "represent, reason, and learn in domains with complex relational and rich probabilistic structure" (GETOOR; TASKAR, 2007). Specifically, they target learning problems encompassing relationships between multiple entities (RAEDT, 2008), which is the case of KBs. However, since KBs can contain millions of entities and billions of relationships, there is a special need to focus on *scalable* SRL techniques. Nickel et al. (2015) considers as scalable in this context methods whose computational complexity grows at most linearly with the size of the graph.

Embedding models, also known as *latent feature models*, are a class of KB completion techniques that models entities as vectors and relations as operations in a continuous, low-dimensional vector space (WANG et al., 2017). Interest in these techniques has grown significantly in the last five years (WANG et al., 2017); a remarkable characteristic of embedding models is that they turn semantic concepts into a smooth vector space where scalable learning is possible by calculating and following gradients. On the other hand, this characteristic also accounts for a

relevant weakness: because embedding models transform a semantically rich input into numeric representations, where each dimension bears little meaning to us, humans, these techniques are very poorly interpretable.

Interpretability is a topic that has received special attention recently due to the rise in popularity of very complex models, such as deep neural networks, that are commonly regarded as not interpretable. As pointed by Ribeiro, Singh e Guestrin (2016), real-world data can be significantly different from test data, making the case where accuracy metrics are not good indicatives of a model's performance and/or robustness. In this scenario, interpretability becomes crucial for assessing *trust* in the models, thus contributing to the adoption of machine learning techniques in real world tasks.

Even though the term "interpretability" is often seen in machine learning conferences, its formalization is an ongoing research topic where there is no clear agreement (DORAN; SCHULZ; BESOLD, 2018). In short, in this work we follow Doran, Schulz e Besold (2018) take at interpretability, in which we consider a model interpretable if it allows the user to study and understand how inputs are mathematically mapped to outputs. For instance, the relative importance of features in a regression model can be realized by comparing covariate weights, whereas features that may be automatically learned and transformed through non-linearities in a deep neural network are unlikely be to interpretable by most users. Moreover, we regard the term "to explain (a prediction)", as used by Ribeiro, Singh e Guestrin (2016), as to present textual or visual artifacts that provide qualitative understanding about the relationship between the instance's components (e.g., relationships between entities, in the case of KBs) and the model's prediction.

Therefore, in this work, we tackle the interpretability problem of embedding models of KBs. To do that, we propose *model-agnostic* methods (i.e., methods that work for *any* embedding model, regardless of its internal structure), by adapting the so-called "pedagogical techniques" from the literature on neural networks, to explain embedding models in the relational setting. Intuitively, a pedagogical technique learns an interpretable model over a non-interpretable but accurate one, without having access to its internal structure. The big advantage of pedagogical techniques, hence, is that they are *model-agnostic* by definition. Model-agnosticity is especially

important in interpretability because we commonly have the scenario where the machine learning practitioner has to assess and compare a model's relative reliability to other alternatives (RIBEIRO; SINGH; GUESTRIN, 2016).

More specifically, the methods proposed here extract weighted Horn rules from embedding models; rules that are both relational and interpretable, hence satisfying our needs. We propose non-trivial adaptations that must be made to apply pedagogical techniques in the relational setting, with respect to interpretable feature generation and handling very large input spaces such as the ones that occur in large KBs. In fact, one can think of several ways to apply model-agnostic techniques to explain the predictions of embedding models of KBs; this work, to the best of our knowledge, is the first to broadly discuss, propose, implement and analyse pedagogical techniques able to explain knowledge embedding models. These contributions have been described in the paper:

- Arthur Colombini Gusmão, Alvaro Henrique Chaim Correia, Glauber De Bona, and Fabio Gagliardi Cozman. Interpreting Embedding Models of Knowledge Bases: A Pedagogical Approach. Proceedings of the ICML Workshop on Human Interpretability in Machine Learning, pp. 79-86, 2018.

During the course of our research, we also published two papers related to *parameter learning* in *probabilistic logic programs*:

- Francisco Henrique Otte Vieira de Faria, Arthur Colombini Gusmão, Fabio Gagliardi Cozman, and Denis Deratani Mauá. Speeding-up ProbLog's parameter learning (Extended Abstract). International Workshop on Statistical Relational AI (StarAI), 2017;
- Francisco Henrique Otte Vieira de Faria, Arthur Colombini Gusmão, Glauber De Bona, Denis Deratani Mauá, and Fabio Gagliardi Cozman. Parameter learning in ProbLog with probabilistic rules. Symposium on Knowledge Discovery, Mining and Learning (KDMILE), pp. 27-34, 2017,

the latter of which received an honored mention (3rd Best Paper Award) at the Symposium on Knowledge Discovery, Mining and Learning (KDMiLe 2017).

The decision to change the research's direction from probabilistic logic programs to embedding models was made due to the fact that, even with the proposed enhancements, learning probabilistic logic programs at a speed capable of processing large KBs in tractable amounts of time does not seem to be feasible.

The rest of this work is divided as follows: in Chapter 2 we provide the reader with a basis to understand the proposed techniques, including interpretability in machine learning, KBs, KB completion, and SRL models of KBs. Reported efforts to bring interpretability to embedding models of KBs or in various similar or related contexts are discussed in Chapter 3. In Chapter 4, we discuss interpretability issues in embedding models of KBs and propose novel, model-agnostic methods for overcoming them. Experiments of the proposed methods are reported in Chapter 5, together with analysis of the results, and in Chapter 6 we draw our final conclusions.

2 Background

The goal of this chapter is to provide the reader with a basis to understand the rest of our work. We start by presenting important definitions that have been proposed in the rather new literature on interpretability in machine learning. Next, we provide an overview of knowledge bases, the task of completing a knowledge base, and statistical relational learning models capable of performing such tasks. In particular, we focus on the models that are especially important for the methods we propose later.

2.1 Interpretability in Machine Learning

Understanding the behavior of predictive systems is essential for promoting scrutability and increasing *trust* in the system (RIBEIRO; SINGH; GUESTRIN, 2016). A means of achieving that is designing for *interpretability*. However, both the motivations underlying interest in interpretability and the notions of interpretability itself are diverse and even discordant in certain occasions; a strict definition of interpretability is still an open question in explainable artificial intelligence (AI) research (LIPTON, 2016). In what follows, we present recent work that tries to refine our understanding and definitions of interpretability and discuss motivations for focusing on interpretable classifiers.

Lipton (2016) suggests that interpretability in fact reflects several distinct ideas rather than being a monolithic concept. The author explores both desiderata and properties of interpretable models. Regarding the first topic, five desiderata are discussed through the lens of the literature on interpretability: trust, causality, transferability, informativeness, and fair and ethical decision-making. In particular, one of the author's definition of trust fits especially well with the usage of the term in this work: "*when the training and deployment objectives diverge, trust might denote confidence that the model will perform well with respect to the real objectives and scenarios.*" When it comes to properties that an interpretable model should

exhibit, the author proposes two broad categories. *Transparency* denotes some sense of understanding how the model works; it is the opposite of *blackbox-ness* or *opacity*. He also distinguishes the level of transparency a model can exhibit. *Simulatability* is the level of transparency of which a person can contemplate the entire model at once; *decomposability* denotes the notion of transparency at which each part of the model admits an intuitive explanation (for instance, the parameters of a linear model could be described as representing strengths of association between each feature and the label); and the notion of *algorithmic transparency* applies to the level of the algorithm itself (e.g., we understand the shape of the error surface for linear models, but we do not understand the heuristic optimization procedures for deep neural networks). The second category for properties of an interpretable model is *post-hoc interpretability*. This category comprises methods that use a distinct approach to extracting informations from learned models without necessarily elucidating how they work. Some of these methods are *text explanations*, *visualization*, *local explanations*, and *explanation by example*. It is interesting to note that when we consider humans to be interpretable (e.g., when they give an explanation of their actions), the latter category is the form of interpretability that applies.

Additionally, Lipton (2016) also discusses some takeaways entailed by these notions of interpretability. Very importantly, he discusses the idea that claims about interpretability must be qualified. That is, to be meaningful, any assertion about interpretability should fit a specific definition. Also, the author warns that transparency may be at odds with broader objectives of AI (the short term goal of focusing on transparent models might prevent further advancements) and that post-hoc interpretations can be potentially misleading (e.g., one might optimize an algorithm to present misleading but pausible explanations. Again, that would be consistent with the behavior of human beings). Finally, a minor but interesting takeaway is that linear models are not strictly more interpretable than neural networks (consider the case where linear models lose simulatability or decomposability by using high dimensional or heavily engineered features in order to achieve similar performance to neural networks), an idea contrary to what is popularly believed.

Motivated by a corpus analysis of NIPS, ACL, COGSCI, and ICCV/ECCV

paper titles on explainable AI, Doran, Schulz e Besold (2018) distinguish between three types of systems: *opaque systems* are the ones that have its input-output mapping invisible to the user, *interpretable systems* are systems that allow the user to see, study and understand its underlying mathematics, and systems that emits symbols along with outputs are called *comprehensible systems*. Finally, the authors introduce a fourth notion: *truly explainable systems* should take upon the deficiencies of interpretable and comprehensible systems in not explaining the decision making process of a model for the end user via a line of *reasoning using human-understandable features* of the input data.

In their general method for explaining the predictions of any classifier, Ribeiro, Singh e Guestrin (2016) define "to explain a prediction" as the task of presenting textual or visual artifacts that provide qualitative understanding of the relationship between the model's prediction and the instance's components (e.g., patches, in the case of an image, or relationships, in the case of KBs). They argue that explaining predictions is especially important to get humans to trust and use machine learning methods, mainly because oftentimes performance in validation data may not correspond to performance "in the wild"; there is a gap between developing (and testing) a system and deploying it (in the real world). Hence trust cannot rely solely on those methods of model evaluation. In addition, the authors advocate for model-agnosticity, noting that explanations are particularly useful when a method is able to produce them for *any* model, thus enabling the user to compare a multitude of methods.

2.2 Knowledge Graphs

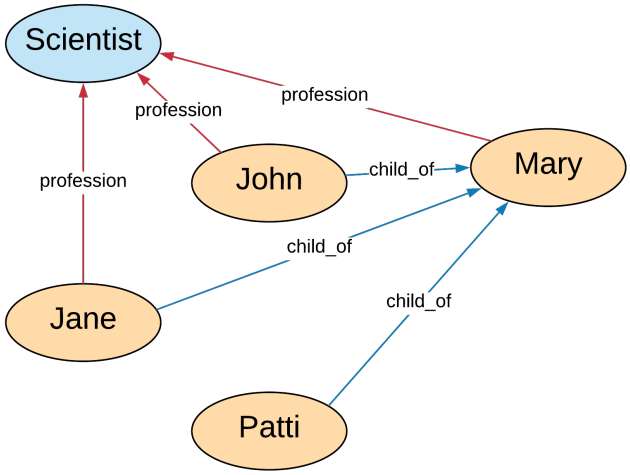
In this section we introduce *graph structured knowledge bases*, also referred to as *knowledge graphs* (KGs), and discuss how they are represented and used. Since we are only interested in these kinds of KBs, in this work we use the terms KB and KG interchangeably.

2.2.1 Knowledge Representation

KGs store information in the form of entities and relations. Following Nickel et al. (2015), in this work we loosely follow the W3C Resource Description Framework (RDF) standard, representing facts in the form of binary relationships, or triples, such as ⟨ head, relation, tail ⟩, where head and tail are entities and relation is the relation between them. The existence of a triple indicates the existence of the corresponding fact. For instance, the set of triples in the example below

Example 2.2.1 [1]

⟨ Jane, child_of, Mary ⟩
⟨ John, child_of, Mary ⟩
⟨ Patti, child_of, Mary ⟩
⟨ Mary, profession, Scientist ⟩
⟨ Jane, profession, Scientist ⟩
⟨ John, profession, Scientist ⟩



indicates that the information

Jane, John and Patti are child of Mary;
Jane, John and Mary have the profession of Scientist.

is correct.

The combination of a set of triples can be encoded as a multigraph, where nodes represent entities and directed edges represent relationships. The direction of an edge indicates whether an entity occur as head or tail, and its type (or label) indicates the corresponding relation (NICKEL et al., 2015). The representation of the KG constituted of the set of triples is also shown in Example 2.2.1.

[1] Adapted from (BORDES; WESTON, 2014).

Besides the facts themselves, KGs often provide type constraints (e.g., only a person can be child of another person) and type hierarchies (e.g., Jane is a child, which is a person, which is a living thing) (NICKEL et al., 2015).

2.2.2 Open and Closed World Assumptions

Even though, in this work, a set of triples will be considered to always encode true relationships, there are different ways in which we can interpret the non-existing triples. The *closed world assumption* (CWA) assumes that non-existing triples are necessarily false relationships, whereas the *open world assumption* (OWA) assumes that a non-existing triple may or may not be false, interpreting it as unknown (NICKEL et al., 2015). For instance, in Example 2.2.1, the fact that the triple

$$\langle \text{ Patti, profession, Scientist } \rangle$$

is not present would be interpreted as *"Patti **does not** have the profession of scientist"* by the CWA. In constrast, the OWA would not interpret the missing triple as not existent. Nickel et al. (2015) notes that this second, more cautious approach is justified since KGs are often very incomplete.

2.2.3 Knowledge Graph Applications

KGs feed multiple big data applications in a multitude of domains. A prime example that demonstrates the value of KGs is IBM's Watson, the question answering system able to beat human experts in *Jeopardy!* (NICKEL et al., 2015). Additional and very prominent examples are the use of KGs to power search engines; both Google (SINGHAL, 2012) and Microsoft (QIAN, 2013) have this implemented. Enhancing search results with semantic information is a step towards transforming text-based search engines into semantically-aware question answering services (NICKEL et al., 2015).

More examples include the use of KBs to build structured representations of natural language texts (SCHUHMACHER; PONZETTO, 2014) and to perform large-scale entity disambiguation in texts extracted from the web (CUCERZAN, 2007). We note that those tasks have been established as crucial in several areas, such

as semantic web search, machine translation, and information retrieval. In addition, there are several specialized domains in which KGs are used, e.g., to provide decision support in the life sciences (BELLEAU et al., 2008; RUTTENBERG et al., 2009).

2.3 Knowledge Base Completion

As already mentioned, a common obstacle in effectively using KGs is that they are incomplete, in the sense that they do not contain all existing facts (i.e., there are missing triples), and, often, their incompleteness is critical. For instance, in English DBpedia 2014, 60% of persons do not have a place of birth and 58% of scientists do not have a fact telling what they are known for (KROMPASS; BAIER; TRESP, 2015).

For a more concrete example, consider the case where a question answering system used the KB described in Example 2.2.1. If we were to ask *"Which is Patti profession?"*, even if the system was able to correcly match existing entities (*"Patti"* in text with entity Patti) and relations (*"profession"* in text with relation profession), it would be impossible for the system to answer the question because no triple in the KB would contain the respectives head entity and relation.

Therefore, much work in the SRL literature has been devoted towards completing KBs by using relational models of them to infer missing facts, a process called *KB completion*. There are two main tasks associated with KB completion: *link prediction* and *triple classification*; we detail both of them in the upcoming subsections. In addition, we discuss *entity resolution*, a task that does not involve adding missing facts to KBs but is closely related to KB completion since it performs de-duplication of entities or relations, which also improves the usefulness of KBs.

2.3.1 Link Prediction

Link prediction is usually defined as the task of predicting an entity that has a specific relation with another entity, that is, predicting the tail entity given the head entity and the relation or predicting the head entity given the relation and the tail entity (NGUYEN, 2017). A simpler form to put it would be to say that

link prediction can be seen as a task similar to answering a query. For instance, in Example 2.2.1, answering the query *"What is Patti profession?"* corresponds to the link prediction task of predicting which entity should occupy the tail position in the triple

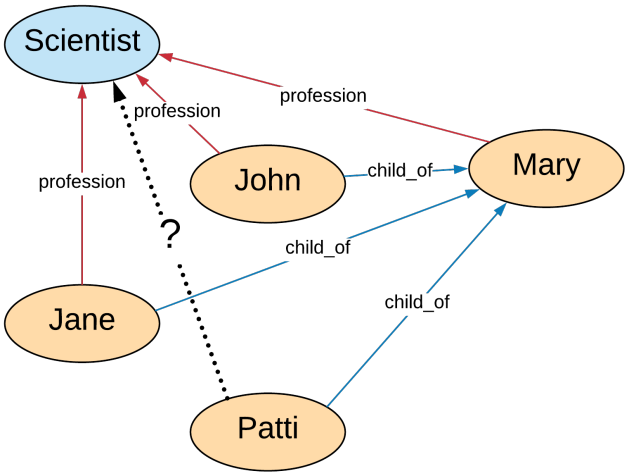
$$\langle \text{ Patti, profession, ? } \rangle.$$

Link prediction is also known as *entity prediction* (LIN et al., 2015) or *entity ranking* (BORDES et al., 2014). Similarly, we can predict relations between two entities, a task referred to as *relation prediction* (LIN et al., 2015).

Regarding task execution, link prediction is usually done by using the KB model to calculate the *plausibility score* of all possible answers for a given query and storing the results in a list ranked in descending order, in order to have a ranked list of candidate answers (WANG et al., 2017). A plausibility score is a real number that the model assigns to a triple in order to quantify the plausibility of its existence. Further details about plausibility scores are addressed in Section 2.4.

For evaluation, recording ranks of correct answers in such ranked lists is a common practice. Several evaluation metrics can be applied to the ranks, so as to see whether the model is assigning higher plausibility to correct answers than to incorrect ones. Examples are the *mean rank* (MR), *mean reciprocal rank* (MRR), *Hits@n*, and the *area under the precision recall curve* (AUC-PR) (WANG et al., 2017). Specifically, consider a set of true facts $\mathcal{D}^{+}_{\text{test}}$ used to evaluate a KB model's performance. The MR corresponds to the average of the predicted ranks for the correct answers in $\mathcal{D}^{+}_{\text{test}}$. The reciprocal rank of a query response is the multiplicative inverse of the rank of the first correct answer (e.g., 1 for first place, 1/2 for second place, etc.); the MRR is the average of the reciprocal ranks for the correct answers in $\mathcal{D}^{+}_{\text{test}}$. Hits@n corresponds to the proportion of ranks for triples in $\mathcal{D}^{+}_{\text{test}}$ no higher than n. The AUC-PR is evaluated over the plausibility score. Sometimes, one may want to consider the area under a constrained portion of the PR curve, as proposed by Garcia-Gasulla et al. (2015). In order to focus on producing high certainty and high utility predictions, the *contrained AUC* (CAUC) "considers only the part of the PR curve where the score is incorrectly accepting

Figure 1 – Example of relationship evaluated in triple classification



Example of missing link to be predicted when performing triple classification for ⟨ Patti, profession, Scientist ⟩ in the KG from example 2.2.1, represented by the dotted edge marked as ? between the two nodes.

less edges than the number of edges in the graph, dismissing the rest of the area under the curve" (Garcia-Gasulla et al., 2015).

2.3.2 Triple Classification

The task of classifying a triple as true or not is known as *triple classification*. Since the models tend to assign higher plausibility scores to triples that are true facts, as mentioned in the previous subsection, the task can be carried out from the rank used for link prediction as follows: given a validation set of triples, calculate the plausibility score for each of them. Then, for each relation r, find an optimal *threshold* δ_r such that triples whose plausibility score falls above the threshold are considered as true facts and triples whose plausibility score falls below the threshold are considered as false ones (WANG et al., 2017).

For instance, consider again Example 2.2.1. The task of evaluating if the triple ⟨ Patti, profession, Scientist ⟩ holds is a case of triple classification. Figure ?? highlights the corresponding edge that would be added the KG if such a triple was considered as a true fact.

Because in the triple classification scenario we can see our model as a traditional classifier, traditional evaluation metrics can be applied, such as micro-averaged accuracy and macro-averaged accuracy (GUO et al., 2015). The former is averaged over all examples; the latter is averaged over the accuracy averages of sets of examples composed of the same relation, for all relations in the KG.

2.3.3 Entity Resolution

In some KGs, different nodes might actually refer to the same object. *Entity resolution* is the task of de-duplicating such nodes (BORDES et al., 2014). In some cases, the execution of the task can be done by reducing it to a special case of triple classification: when there is a relation that indicates the equivalence between two entities, e.g., equals_to, entity resolution degenerates into predicting whether the triple that contains both entities and the equivalence relation holds, e.g., if the triple $\langle$ head, equals_to, tail $\rangle$ holds.

However, a different approach is needed when there is no equivalence relation in the KG. To this end, Nickel, Tresp e Kriegel (2011) proposed to perform entity resolution by comparing entity representations. Specifically, let $\boldsymbol{h}$ and $\boldsymbol{t}$ denote the vectors in the embedding space that represent entities e_h and e_t, respectively. They proposed a score $s(e_h, e_t) = e^{-\|\boldsymbol{h}-\boldsymbol{t}\|_2^2/\sigma}$ that represents the similarity between the two entities, used to measure the likelihood that e_h and e_t refer to the same entity.

The most widely adopted evaluation metric for entity resolution is the area under the precision-recall curve (AUC-PR) (WANG et al., 2017), where precision is calculated over entity pairs whose entities were classified as referring to the same object and recall is calculated over entity pairs known to have its entities refer to the same object.

2.4 Statistical Relational Learning for Knowledge Graphs

Statistical Relational Learning targets the creation of statistical models for relational data. In this section, we address statistical relational learning models of knowledge graphs. Following Nickel et al. (2015), we assume that all entities and

relations in a KG are known, whereas triples are assumed to be incomplete and noisy.

2.4.1 Probabilistic Knowledge Graphs

Let $\mathcal{E} = \{e_1, ..., e_{N_e}\}$, of size N_e, be the set of all entities and $\mathcal{R} = \{r_1, ..., r_{N_r}\}$, of size N_r, be the set of all relations in a KG $\mathcal{G}$, where N_e and N_r are the number of entities and relations, respectively. Also, let $x_{h,r,t} = \langle e_h, r_r, e_t \rangle$ correspond to a possible triple. We say that entity e_h is the *head* of $x_{h,r,t}$ if it appears as subject of the relationship and entity e_t is the *tail* of $x_{h,r,t}$ if it appears as object. Each possible triple can be modeled as a binary random variable $y_{h,r,t} \in \{0, 1\}$ that indicates its existence (NICKEL et al., 2015):

$$y_{h,r,t} = \begin{cases} 1, & \text{if the triple } \langle e_h, r_r, e_t \rangle \text{ exists} \\ 0, & \text{otherwise.} \end{cases}$$

Note that, under the OWA, $y_{h,r,t} = 0$ does not imply that $x_{h,r,t}$ is a false fact, but simply that it is not observed in $\mathcal{G}$, as discussed in Subsection 2.2.2. Under the CWA, on the other hand, having $y_{h,r,t} = 0$ would imply that the corresponding triple is a false fact.

Let $\mathcal{T} = \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, of size N_t, denote the set of all possible triples in $\mathcal{G}$, and $\mathcal{Y} \in \{0, 1\}^{N_e \times N_r \times N_e}$ the set of their corresponding binary random variables. We say that each possible realization of $\mathcal{Y}$ can be interpreted as a possible world. Hence, when deriving a model for the entire KG, we are interested in estimating the joint distribution $\mathbb{P}(\mathcal{Y})$, from a subset of observed triples $\mathcal{D} \subset \mathcal{T}$ (NICKEL et al., 2015), of size $N_d < N_t$. In the rest of this work, we always make the open world assumption (OWA), which means that $\mathcal{D}$ stores only true facts.

Notice that, although $\mathcal{Y}$ can be enormous for large KGs, only a tiny fraction of triples are likely to be true facts. This remains true even when we have a KB with type constraints and we consider only syntactically valid facts (triples whose combination of head, relation and tail do not violate the type constraints). Consider the example mentioned by Nickel et al. (2015): there are over 250,000 movies and 450,000 thousand actors in Freebase, but each actor stars only in a small number of movies. Thus, an important issue when modeling KGs is dealing with the large

number of possible relationships while exploiting its sparsity (NICKEL et al., 2015). To this end, Nickel et al. (2015) argues that, ideally, a relational model should scale at most linearly in N_e, linearly in N_r, and linearly in N_d, to efficiently handle large-scale KGs.

In the rest of this section, we present different models of knowledge bases. The models covered here fall within two classes: *embedding models* are the models we are interested in interpreting, and *graph feature models* are models that comprise techniques that we later use to explain the predictions of the first category.

2.4.2 Embedding Models

Embedding models embed components of KGs, such as entities and relations, into vectors or operations in a continuous vector space. Usually, vectors represent entities and operations represent relations. Each dimension of such representations is called a *latent feature*, in the sense that they model intrinsic characteristics of the respective components and are not observed in the data. For instance, a possible explanation for the fact "*Patti has the profession scientist*" may be that Patti is curious about nature. This explanation uses latent features of entities (curiosity about nature) to explain observable facts (Patti having the profession of scientist), as described in Example 2.4.1. The task of embedding models is to infer these features automatically from the data; hence, they are also known as *latent feature models* (NICKEL et al., 2015). The main goal of using them is that the embedded representations allow to simplify computations on the KG while maintaining its inherent structure (WANG et al., 2017). In addition, the models perform representation learning when automatically learning features, eliminating the need for feature engineering.

Example 2.4.1 [2]

Let an embedding model learned over the KB introduced in Example 2.2.1 represent each entity i *as a vector* $\boldsymbol{e}_i \in \mathbb{R}^k$*, where* k *denotes the number of latent features in the model. Particularly, let the entity* Patti *be represented by the vector* $\boldsymbol{e}_{Patti}$ *and the entity* Scientist *be represented by the vector* $\boldsymbol{e}_{Scientist}$*. We could model that*

[2] Adapted from (NICKEL et al., 2015).

Patti is curious about nature and that a scientist is, for instance, an inquisitive professional via the vectors

$$\boldsymbol{e}_{Patti} = \begin{bmatrix} 0.9 \\ 0.2 \\ \vdots \end{bmatrix} \quad \textit{and} \quad \boldsymbol{e}_{Scientist} = \begin{bmatrix} 0.2 \\ 0.8 \\ \vdots \end{bmatrix},$$

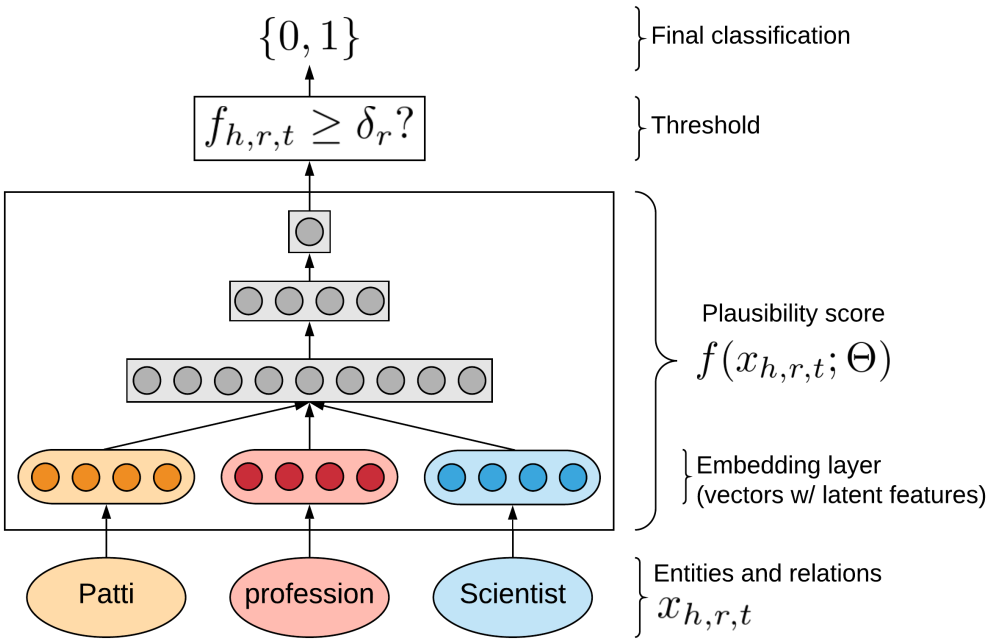
where the first latent feature $\boldsymbol{e}_{i,1}$ of each vector represents the characteristic 'curious about nature' and the second latent feature $\boldsymbol{e}_{i,2}$ represents the characteristic 'inquisitive professional'. Notice, however, that unlike in this example, latent features learned by embedding models are usually hard to interpret (NICKEL et al., 2015).

A typical embedding model defines a *score function* $f(x_{h,r,t}; \Theta)$ that represents the *plausibility* (the model's confidence) that a triple $x_{h,r,t}$ is a true fact given the set of all parameters Θ, as shown in Figure 2. The part of the model that comprises the entities' and relations' embedded representations is known as the *embedding layer*, which can be seen as a lookup table that maps entities and relations to its corresponding representations (e.g., vectors or matrices). The set of all parameters includes both the embedding layer and additional parameters. For instance, f could be a neural network (NN) which has several layers upon the embedding layer, and each additional layer's parameters would also be part of Θ. By its definition, the underlying assumption that any embedding model makes is that the variables $y_{h,r,t}$ are conditionally independent given Θ.

2.4.2.1 Model Training

Usually, one learns Θ by solving an optimization problem that maximizes the total plausibility of observed facts $\mathcal{D}$ (WANG et al., 2017). The embedding layer is randomly initialized, as the other layer's parameters, using a specific probability distribution that has been empirically observed to improve training, such as a truncated normal distribution. Next, model training happen in the same fashion as traditionally done in NNs, using *stochastic gradient descent* (SGD) or one of its variants. Indeed, the score functions of embedding models used in knowledge base completion can generally be viewed as neural networks (NICKEL et al., 2015; NGUYEN, 2017; WANG et al., 2017).

Figure 2 – General embedding model scheme



A general scheme for an embedding model for KB completion. Entities and relations are inputs ($x_{h,r,t}$) to an arbitrary plausibility score function f. Notice that, to perform link prediction, assessing the resulting plausibility score $f_{h,r,t}$ is enough. However, for the case of triple classification, one must calculate and apply a threshold δ_r to the plausibility score in order to obtain a final classification, as shown at the top of the figure.

Training embedding models under the OWA requires that we find a way to generate negative training examples, in order to give the optimization problem a reference of less plausible triples. This is done by, given a set of observed true facts $\mathcal{D}$, constructing a negative set $\mathcal{D}^-$ whose triples are derived from $\mathcal{D}$ by replacing either the head or the tail entity with a random entity sampled uniformly from $\mathcal{E}$:

$$\begin{aligned}\mathcal{D}^- = \{\langle e_{h'}, r_r, e_t\rangle \mid e_{h'} \in \mathcal{E} \wedge e_{h'} \neq e_h \wedge \langle e_h, r_r, e_t\rangle \in \mathcal{D}\} \\ \cup \{\langle e_h, r_r, e_{t'}\rangle \mid e_{t'} \in \mathcal{E} \wedge e_{t'} \neq e_t \wedge \langle e_h, r_r, e_t\rangle \in \mathcal{D}\}.\end{aligned}$$

Sometimes, $\mathcal{D}^-$ can also be generated by randomly corrupting the relation

as well (WANG et al., 2017), i.e.:

$$\begin{aligned}\mathcal{D}^- = &\{\langle e_{h'}, r_r, e_t\rangle \mid e_{h'} \in \mathcal{E} \wedge e_{h'} \neq e_h \wedge \langle e_h, r_r, e_t\rangle \in \mathcal{D}\} \\ &\cup \{\langle e_h, r_r, e_{t'}\rangle \mid e_{t'} \in \mathcal{E} \wedge e_{t'} \neq e_t \wedge \langle e_h, r_r, e_t\rangle \in \mathcal{D}\} \\ &\cup \{\langle e_h, r_{r'}, e_t\rangle \mid r_{r'} \in \mathcal{R} \wedge r_{r'} \neq r_r \wedge \langle e_h, r_r, e_t\rangle \in \mathcal{D}\}.\end{aligned}$$

However, one issue with generating negative examples sampled from a uniform distribution is that false-negative examples might be often generated, especially in the case of N-to-1 or 1-to-N relations (for instance, consider the relation *gender*, that has only two possible tail entities, namely, *male* or *female*, but many possible head entities). Thus, to improve negative training examples generation, Wang et al. (2014) proposed to corrupt either the head or the tail with different probabilities, using a Bernoulli distribution. The idea is to give more chance of replacing the head if the relation tends to be more 1-to-N, and to give more chance of replacing the tail if the relation tends to be more N-to-1. More formally, given a relation and its positive facts, let *tph* denote the mean number of tail entities per head entity and *hpt* denote the mean number of head entities per tail entity. The method proposes to corrupt the original fact by replacing the head with probability $tph/(tph + hpt)$, and the tail with probability $hpt/(tph + hpt)$.

In the following subsections, we describe instances of embedding models in more detail, where we present specific definitions of loss functions that use the generated negative training examples.

2.4.2.2 TransE

TransE (BORDES et al., 2013) is a simple yet efficient embedding model that is generally used as baseline for knowledge base completion (GUO et al., 2015; TROUILLON et al., 2016; LIU; WU; YANG, 2017). The model is inspired by methods such as Word2Vec (MIKOLOV et al., 2013) as it represents entities and relations as points in the same vector space. Relationships are represented as translations in the embedding space: if a triple holds, the vector that represents the tail entity should be close to the head entity vector plus the relation vector.

Formally, let $\boldsymbol{h} \in \mathbb{R}^k$, $\boldsymbol{r} \in \mathbb{R}^k$ and $\boldsymbol{t} \in \mathbb{R}^k$ denote the vectors that represent the head entity, the relation and the tail entity of an input triple $x_{h,r,t}$, respectively.

Figure 3 – Illustration of TransE embedding vectors.

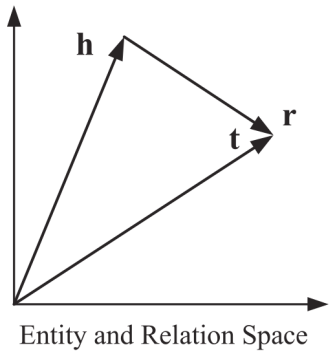


Illustration of TransE embedded vectors in a 2D plane, adapted from (WANG et al., 2014). $\boldsymbol{h}$, $\boldsymbol{r}$ and $\boldsymbol{t}$ are vectors that represent the head entity, the relation and the tail entity of an input triple $x_{h,r,t}$, respectively. The vectors are illustrated to represent the case where $x_{h,r,t}$ holds perfectly, that is, when the plausibility score is zero ($f^{\text{TransE}}(x_{h,r,t}) = 0$).

TransE defines the score function

$$f^{\text{TransE}}(x_{h,r,t}) = -\left\|\boldsymbol{h} + \boldsymbol{r} - \boldsymbol{t}\right\|_{\ell_{1/2}},$$

where $\ell_{1/2}$ indicates the application of either the ℓ_1 or the ℓ_2 norm. The model is learned in such a way that the score tends to be higher if $\langle e_h, r_r, e_t \rangle$ holds, and lower otherwise. Figure 3 shows a simple illustration of how the vectors should interact in the case where a triple holds perfectly. More specifically, the embedding vectors in TransE are trained to minimize the pairwise ranking loss function

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

over the training set, where $\mathcal{D}$ denotes the set of observed triples, $\mathcal{D}^-$ denotes the set of corrupted triples, as described in 2.4.2.1, and γ denotes the margin, an additional hyperparameter that should be tuned.

During training, we generate a new set of negative examples $\mathcal{D}^-$ for each true fact in $\mathcal{D}$ at each epoch, with the goal of improving the plausibility score of true facts without necessarily forcing a smaller set of generated (and possibly incorrect) negative examples to be classified as false facts.

Despite its efficiency, TransE has flaws in handling 1-to-N, N-to-1, and N-to-N relationships (WANG et al., 2014). To overcome this issue, several generalizations of TransE have been proposed (WANG et al., 2017), which we do not cover in this work. Rather, we introduce another class of embedding model, namely, ANALOGY, discussed in the next section.

2.4.2.3 ANALOGY

ANALOGY (LIU; WU; YANG, 2017) is an embedding model able to capture the *analogical* properties of the embedded entities and relations. It makes use of a mathematical formulation of desirable analogical structures and leverages them in the objective function, in order to optimize the embeddings with respect to the analogical properties. The model represents entities as vectors and relations as linear transformations defined by matrices (also known as linear maps). If a triple holds, the linear transformation that represents the relation should transform the vector that represents the head entity from its original position in the vector space to somewhere near the vector that represents the tail entity. Additionally, ANALOGY defines constraints so as to make the linear transformations have the properties of *normality* and *commutativity*, making them a *commuting family* of normal matrices. In short, the authors call as *commuting family* the property of any pair of relations in $\mathcal{R}$ to satisfy the commutative constraint, i.e.,

$$r_n \circ r_{n'} = r_{n'} \circ r_n \ ,$$

where $\circ$ denotes the composition operation between two relations, which is naturally implemented via matrix multiplication in the embedding space. Figure 4 shows an illustration of linear maps that obey this constraint.

Formally, let $\boldsymbol{v}_n \in \mathbb{R}^k$ represent the embedding vector for entity e_n and $\boldsymbol{W}_n \in \mathbb{R}^{k \times k}$ represent the embedding matrix for relation r_n. ANALOGY defines the score function

$$f^{\text{ANALOGY}}(x_{h,r,t}) = \langle \boldsymbol{v}_h^\intercal \boldsymbol{W}_r, \boldsymbol{v}_t \rangle = \boldsymbol{v}_h^\intercal \boldsymbol{W}_r \boldsymbol{v}_t \ ,$$

that outputs higher scores for true facts and lower scores to false ones. Hence, the goal is to learn the embedding vectors and matrices from a set of observed facts

Figure 4 – Illustration of ANALOGY linear maps.

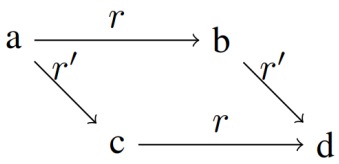


Illustration of ANALOGY linear maps forming a parallelogram, adapted from (LIU; WU; YANG, 2017). a, b, c and d represent entities, and r and r' relations. The relations satisfy the commutative constraint, i.e., both compositions $r \circ r'$ and $r' \circ r$ correspond to the same resulting linear transformation in the embedding space.

$\mathcal{D}$ in a way that for any valid triple $x_{h,r,t} = \langle e_h, r_r, e_t \rangle$ the equation $\boldsymbol{v}_h^\intercal \boldsymbol{W}_r \approx \boldsymbol{v}_t^\intercal$ is satisfied. That can be done by minimizing a loss function that penalizes higher scores for false facts and lower scores for true facts. One example is the log loss, used in the original paper's experiments,

$$\mathcal{L} = \sum_{x_{h,r,t} \in \mathcal{D} \cup \mathcal{D}^-} -\log \sigma(z_{h,r,t} \cdot f(x_{h,r,t})),$$

where σ denotes the sigmoid activation function, $\mathcal{D}$ denotes a set of true facts, $\mathcal{D}^-$ denotes a set of false ones, and

$$z_{h,r,t} = \begin{cases} +1, & \text{if } x_{h,r,t} \in \mathcal{D} \\ -1, & \text{if } x_{h,r,t} \in \mathcal{D}^- \end{cases}$$

denotes a label used by the loss function. Again, when under the OWA, $\mathcal{D}^-$ can be generated by corrupting observed triples in $\mathcal{D}$, as described in 2.4.2.1. When minimizing the loss, ANALOGY imposes the constraints

$$\begin{aligned} \boldsymbol{W}_n \boldsymbol{W}_n^\top &= \boldsymbol{W}_n^\top \boldsymbol{W}_n \quad \forall r_n \in \mathcal{R} \quad \text{and} \\ \boldsymbol{W}_n \boldsymbol{W}_{n'} &= \boldsymbol{W}_{n'} \boldsymbol{W}_n \quad \forall r_n, r_{n'} \in \mathcal{R} \ , \end{aligned}$$

of which the former corresponds to *normality* and the latter to *commutativity*, as mentioned earlier in this subsection.

As one may notice, ANALOGY's constrained optimization appear to be computationally expensive at first. However, the authors exploited special properties

of commuting normal matrices to formulate an alternative objective function, of substantially lower complexity than the original one, making optimization in the model computationally tractable. We do not cover the derivation of the more efficient algorithm here, asking the interested reader to refer to the original paper for more details.

Lastly, we mention ANALOGY's authors' comparison of the model with additive transformation models (such as TransE), in which they argue in favor of their model's capacity: *"the linear transformation defined by a matrix (...) is a richer operator than the additive transformation defined by a vector"*.

2.4.3 Subgraph Feature Extraction

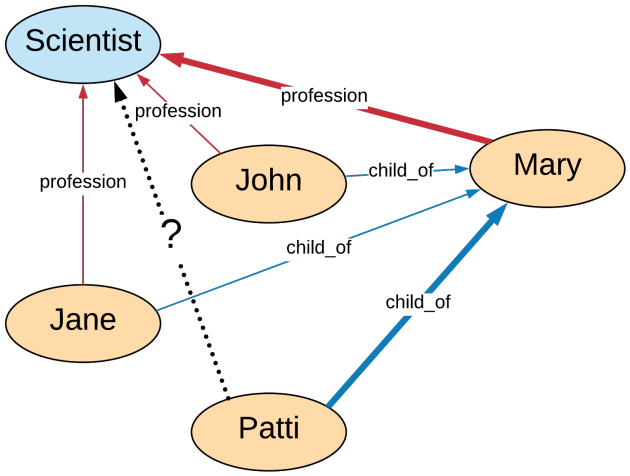
In contrast to embedding models, *graph feature models* do not assume that the variables $y_{h,r,t}$ are conditionally independent given a set of latent features and parameters. Rather, they predict the existence of a relationship based on features extracted from observed edges in the knowledge graph (NICKEL et al., 2015).

There are several methods that perform KB completion using graph features. However, in this work we focus on one that is especially important to the methods we propose in the upcoming chapters, namely, *subgraph feature extraction* (SFE).

SFE (GARDNER; MITCHELL, 2015) is a variant of the *Path Ranking Algorithm* (PRA) (LAO; COHEN, 2010; LAO; MITCHELL; COHEN, 2011) shown to achieve better performance, with respect to both running time and prediction metrics. For a given triple in a KG, SFE builds a *feature vector* composed of binary features from the graph. Each feature indicates the existence of a path, i.e., a sequence of relations, between the triple's entities. The method then concatenates all feature vectors for triples of the same relation to build a *feature matrix*, that later can be used as part of the training data in any desired classification model (traditionally logistic regression).

Formally, let π denote a path type defined by some sequence of edges (i.e., relations) -r_1-r_2-...-r_l- in a knowledge graph $\mathcal{G}$. SFE constructs a subgraph $\mathcal{G}_n$ centered around each entity $e_n \in \mathcal{G}$ using k random walks. Each random walk that leaves e_n follows some path type $\pi_{n,i}$ and ends at an intermediade node e_i. To

Figure 5 – Example of path found by SFE.



Applying SFE to the KG introduced in Example 2.2.1, only two different path types would be found between nodes Patti and Scientist. One such path, of length 2, is composed of the relations child_of and profession, whose edges are highlighted in bold on the figure. This path corresponds to the feature value highlighted in bold in Table 1. The other path would be composed of the relations child_of, child_of^{-1} and profession (there are two different such paths on the graph, but notice that the path type—the sequence of relations—is the same), in the given order, where the superscript "$^{-1}$" denotes an edge followed in its inverse direction. Information about the existence or absense of such paths can be useful when trying to predict a missing link (represented by the dotted edge marked as ?) between the two nodes.

construct a PRA-like feature vector for a source-target pair (e_n, e_m), SFE merges the subgraphs $\mathcal{G}_n$ and $\mathcal{G}_m$ on the intermediate nodes e_i, taking the combinations of all path types $\pi_{n,i}$ and $\pi_{m,i}$ for all e_i as binary features. Each feature vector is saved as a row in a feature matrix, that can be used as input to any supervised classifier. Figure 5 highlights one instance of path that SFE would find by being applied to the KB from Example 2.2.1, and Table 1 shows part of the resulting feature matrix that SFE would yield by being applied to the same KB.

Because features extracted by PRA can be understood as bodies of weighted rules, the model is usually regarded as "easily interpretable" (NICKEL et al., 2015). As features are restricted Horn clauses extracted from the graph, the method is

Table 1 – Example of Feature Matrix built by SFE.

	child_of	**child_of**$^{-1}$	**child_of, profession**	**child_of, child_of**$^{-1}$	...	**child_of, child_of**$^{-1}$**, profession**	...
(Patti, Scientist)	0	0	**1**	0	...	1	...
(Patti, Mary)	1	0	0	0	...	0	...
(Patti, Jane)	0	0	0	1	...	0	...
⋮	⋮	⋮	⋮	⋮	⋱	⋮	⋮
(Jane, Scientist)	0	0	1	0	...	1	...
⋮	⋮	⋮	⋮	⋮	...	⋮	⋱

Partial representation of a feature matrix for the relation profession that SFE would build when applied to the KG introduced in Example 2.2.1. Each row of the feature matrix corresponds to an entity pair, and each column to a path type. Thus, each feature value (i.e., each cell of the feature matrix) indicates the existence of the respective path type (column) between the respective entity pair (row) if it is 1 or its absence if it is 0. The feature value highlighted in bold for the node pair (Patti, Scientist) corresponds to the path highlighted in Figure 5.

closely connected to logical inference (GARDNER; TALUKDAR; MITCHELL, 2015). Moreover, one could argue that, for the specific case of SFE, the fact that features are binary makes the interpretation of its explanations rather direct. SFE also allows for extending its feature space and using more expressive features, but in this work we restrict ourselves to paths between two nodes in the graph, as described above, which are referred to as "PRA-like" features in the literature (GARDNER; MITCHELL, 2015).

3 Related Work

In this chapter, we start by reviewing concepts from the literature of non-relational rule extraction from NNs, which helped to shape our work, and then move to topics more directly related to our interest, namely, interpretability in the context of relational learning and SRL models of KBs.

As discussed in Subsection 2.4.2, the score functions of embedding models used in KB completion can generally be regarded as neural networks (NNs). So, at least in part, what we are interested in when trying to interpret embedding models is to understand how NNs operate. Thus, we start by reviewing concepts from the literature of non-relational rule extraction from NNs, of which a considerable part comes from the 1990s. Andrews, Diederich e Tickle (1995) classifies rules extraction algorithms in five dimensions, one of them being the *translucency* of the neural network to the eyes of the rule extraction method. In this dimension, *decompositional* techniques are the ones that focus on extracting rules from the level of hidden layers in the neural network (the internal structure of the network is seen as transparent, in the sense that the method has access to its internal values), and *pedagogical* techniques are the ones that treat the network as a black box. Methods that combine elements of both approaches are called *eclectic*. Recent work can be found for both pedagogical (AUGASTA; KATHIRVALAVAKUMAR, 2012) and decompositional (ZILKE; MENCÍA; JANSSEN, 2016) approaches. In particular, Zilke, Mencía e Janssen (2016) extended to deep networks a decompositional approach, which was defined only for the case of NNs with one hidden layer.

In the relational learning scenario, we find methods that are able to extract knowledge from neural networks in the form of rules; techniques that are often called *neuro-symbolic integration* (GARCEZ et al., 2015). More recently, França, Garcez e Zaverucha (2015) adapted TREPAN (CRAVEN; SHAVLIK, 1996), an algorithm that uses queries to induce a decision tree that tries to model a NN, to extract rules from a network trained in data propositionalized from a first-order example set. Another example is the framework proposed by Srinivasan e Vig

(2017), that constructs explanations from a logical model that uses the structure and prediction of a network whose inputs are patterns in first-order logic.

Switching now to the context of KBs, there are also a variety of techniques that aim at obtaining interpretability. Murphy, Talukdar e Mitchell (2012) take the interpretability of each dimension of a word embedding as the capacity one has of distinguishing an intrusive word, i.e., one which has a low value in that dimension in comparison to other words in a group. From that perspective, they proposed a variant of matrix factorization that is highly interpretable. Chandrahas et al. (2017) proposed a technique for inducing interpretability in KB embeddings by incorporating additional entity co-occurence statistics from text, while still maintaining comparable performance in predictive tasks. Barbieri, Bonchi e Manco (2014) proposed a stochastic topic model for link prediction that produces explanations based on the type of each predicted link (links can be "topical" or "social"). KSR (XIAO, 2016) learns semantic features for knowledge graphs. Xie et al. (2017) proposed an embedding model and designed a learning algorithm to induce interpretable sparse representations in it. Engelen, Boekhout e Takes (2016) developed an efficient and explainable technique for link prediction using topological features.

Finally, we mention the work developed by Carmona e Riedel (2015), which perhaps is the reported effort that is closest to ours. They have employed pedagogical techniques for extracting interpretable models from matrix factorization models for KB completion. In constrast to our work, their application is restricted to the model structure of matrix factorization techniques. In addition, it does not address difficulties with feature building and instance generation that appear with large KBs, issues we have tackled and whose solution alternatives are detailed in the next chapter.

4 Explaining Embedding Models of Knowledge Bases

In this chapter, we discuss issues in the interpretability of embedding models, and propose *model-agnostic* methods for overcoming them. The methods proposed here do not restrict the model's structure or its input feature space. Thus, they differ from previous work by being the first that, to the best of our knowledge, explain the predictions of embedding models for KB completion via weighted Horn rules in a model-agnostic fashion. As discussed in the end of Section 2.1, pedagogical approaches allow a model to have its relative trust assessed and compared to other methods, a property that is especially important in the scenario where the machine learning practitioner has to select a model among a number of alternatives (RIBEIRO; SINGH; GUESTRIN, 2016).

4.1 Challenges in Interpreting Embeddings

In the previous chapter, we mentioned techniques able to build interpretable classifiers that mimick neural networks and, thus, to offer insights into how they work. However, embedding models offer an intrinsic difficulty in effectively using these techniques for interpretability: because embedding models turn a semantically rich input (i.e., the space of concepts composed of entities and relations) into numeric representations (i.e., vectors in a vector space), one cannot operate in the vector space in which the classification takes place. Rather, one must return to the space of entities and relations, where interpretations can be meaningful to the end user. How sensible would an explanation generated by a logistic regression operating on $\Re^{40}$ be, were it to state that dimension #28 is the one determining a triple to exist?

As a matter of fact, there is an fundamental reason why one must not accept explanations based on the embedding's dimensions: the embedding layer is still part of the model itself, and must also be explained. So, even if the embedding

vector's dimensions were humanly interpretable, we still could not rely solely on using the values of the most relevant dimensions as explanations. This is because we would still have to explain why the model assigned such values to the entities and relation in question. This point is better discussed in the next example.

Example 4.1.1

Suppose we learned an embedding model as the one discussed in Example 2.4.1, in such a way that we are sure of each latent feature meaning, and we are interested in understanding why the model predicts that Patti has the profession of scientist. Also, suppose that, in such a model, Patti having a high value in its first latent feature $\mathbf{e}_{Patti,1}$*, which corresponds to the characteristic 'curious about nature', is the most relevant factor for the embedding model's prediction. If we were to explain the predicted triple based on the embedding's dimensions, we would give an explanation of the form* "Patti has the profession of scientist because Patti is curious about nature"*. However, this explanation is not enough to build trust in the model. Notice that, in doing that, we are taking for granted the mechanism that the model used to assign a high value to dimension* $\mathbf{e}_{Patti,1}$*, and that we would still have to explain why that happened (i.e., why Patti was modeled as an entity that is curious about nature). This last step is crucial because, even though the first explanation makes sense for one who believes in this line of reasoning, it could be for the wrong reason. For instance, the model could have learned that Patti is curious about nature because, say, it captured a dataset bias (e.g., the majority of people in the data were scientists) and is assigning that every person is curious about nature, regardless of other factors. This is a reason that would actually decrease the user's trust in the model, making it to be seen as less robust. Conversely, the model could have assigned a high value to* $\mathbf{e}_{Patti,1}$ *because Patti is similar to many people who are scientists, a reason that would probably increase one's trust in the model.*

In sum, the inherent intricacies of embedding models require that we find a way to provide interpretations based on the semantic space of triples, in such a way that the entire behavior of the embedding model is captured and the interpretations make sense to the end user. To this end, we propose two methods, detailed in the following sections.

4.2 Explaining Knowledge Embedding Models with Predicted Features (XKE-PRED)

Here we propose the first method for explaining embedding models of KBs. It is perhaps the simpler pedagogical technique one can think of. We treat the embedding model as a black box, and assume no other source of information is available. The basic idea is to construct a training set for the interpretable classifier by feeding the original classifier with varied inputs and observing its outputs. Notice, however, that because each input to the original classifier consists of simply a triple, which has no inherent (interpretable) features, this is not feasible in principle. To overcome this difficulty, we resort to applying SFE (presented in Subsection 2.4.3) to a KG composed of triples predicted as true facts by the black box, which we refer to as a *predicted graph*. With it, we can proceed with the default pedagogical framework: by using features (in the form of paths in the graph—or Horn clauses) extracted by SFE and labels (which indicate the existence or absence of a fact) predicted by the embedding model, we train a logistic regression, from which we draw explanations in the form of weighted Horn clauses.

Specifically, let $\mathcal{T} = \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ denote the set of all possible triples in a relational setting; let $\Pi_{\mathcal{G}}$ represent the set of all possible path types between two entities in a knowledge graph $\mathcal{G}$ and $\mathcal{P}(\Pi_{\mathcal{G}})$ its power set; let $\text{SFE}\colon \mathcal{T} \to \mathcal{P}(\Pi_{\mathcal{G}})$ denote the feature extraction function performed by SFE for a triple $x_{h,r,t} \in \mathcal{T}$, given $\mathcal{G}$; and let $\Pi_{h,r,t|\mathcal{G}} \in \mathcal{P}(\Pi_{\mathcal{G}})$ denote the result of applying SFE in $\mathcal{G}$ for the triple $x_{h,r,t}$, i.e., a set of path types that exist between entities h and t in $\mathcal{G}$. We call *explaining knowledge embedding models with predicted features* (XKE-PRED) the following scheme. First, construct a graph

$$\hat{\mathcal{G}} = \{x_{h,r,t} \in \mathcal{T} \mid g(x_{h,r,t}) = 1\}$$

composed of all triples predicted as true facts by the embedding model $g\colon \mathcal{T} \mapsto \{0, 1\}$. Then, select an arbitrary set of triples $\mathcal{D} \subseteq \mathcal{T}$ to serve as training instances for an interpretable classifier. Next, build a training set

$$\mathbb{D}' = \{(\text{SFE}(x_{h,r,t} \mid \hat{\mathcal{G}}), g(x_{h,r,t})) \mid x_{h,r,t} \in \mathcal{D}\}$$

that contains, for each instance in $\mathcal{D}$, features extracted by SFE from the predicted graph $\hat{\mathcal{G}}$ and a label that corresponds to the embedding model's prediction. Then, train an interpretable classifier $g' \colon \mathcal{P}(\Pi_{\hat{\mathcal{G}}}) \mapsto \{0, 1\}$ using $\mathbb{D}'$ as training data. Finally, draw explanations from the interpretable classifier g'. A more detailed version of the method is presented as a procedure in Algorithm 1.

Algorithm 1 XKE-PRED

function BUILD-PREDICTED-TRUE-SET(g, $\mathcal{T}$)
    $\hat{\mathcal{G}} \leftarrow \{\}$ ▷ Initialize $\hat{\mathcal{G}}$ as an empty set
    for all $x_{h,r,t} \in \mathcal{T}$ **do** ▷ For every triple in $\mathcal{T}$
        if $g(x_{h,r,t}) = 1$ **then** ▷ If triple is predicted as true fact
            $\hat{\mathcal{G}} \leftarrow \hat{\mathcal{G}} \cup \{x_{h,r,t}\}$ ▷ Add triple to graph
    return $\hat{\mathcal{G}}$
function BUILD-TRAINING-SET(g, $\mathcal{D}$, $\mathcal{G}$)
    $\mathbb{D} \leftarrow \{\}$ ▷ Initialize $\mathbb{D}$ as an empty set
    for all $x_{h,r,t} \in \mathcal{D}$ **do** ▷ For every triple in $\mathcal{D}$
        $\Pi_{h,r,t|\mathcal{G}} \leftarrow \text{SFE}(x_{h,r,t} \mid \mathcal{G})$ ▷ Extract features using SFE
        $\hat{y} \leftarrow g(x_{h,r,t})$ ▷ Get the embedding model's prediction
        $\mathbb{D} \leftarrow \mathbb{D} \cup \{(\Pi_{h,r,t|\mathcal{G}}, \hat{y})\}$ ▷ Add instance to dataset
    return $\mathbb{D}$
procedure XKE-PRED(g, $\mathcal{T}$)
    $\hat{\mathcal{G}} \leftarrow$ BUILD-PREDICTED-TRUE-SET(g, $\mathcal{T}$)
    Arbitrarily selected a set of triples $\mathcal{D} \subseteq \mathcal{T}$
    $\mathbb{D}' \leftarrow$ BUILD-TRAINING-SET(g, $\mathcal{D}$, $\hat{\mathcal{G}}$)
    Using $\mathbb{D}'$, train an interpretable classifier $g' \colon \mathcal{P}(\Pi_{\hat{\mathcal{G}}}) \mapsto \{0, 1\}$
    Draw explanations from g'

4.2.1 Predicted Graph via Entity Similarity

Generating the fully complete predicted graph $\hat{\mathcal{G}}$ by trying all possible triples is not feasible in practice, as the number of possible triples $|\mathcal{T}| = |\mathcal{E}|^2 \cdot |\mathcal{R}|$ grows quadratically with the number of entities. Even for relatively small graphs, such as the ones in Table 2, we already have billions of possible triples. Moreover, as discussed in Subsection 2.4.1, only a small subset of $\mathcal{T}$ is likely to be true, which means that running inference for each possible triple would be very inefficient anyway.

To arrive at a tractable number of triples to be inferred, a naïve approach would be to sample a subset $\mathcal{T}_0 \subset \mathcal{T}$ of all possible triples and get their embedding model's predictions to build a subset $\hat{\mathcal{G}}_0 \subset \hat{\mathcal{G}}$ of all true predictions. Rather, we propose a more sophisticated method, that uses the entities' vector representations comprised in the parameters Θ_g of an embedding model classifier $g(\,\cdot\,;\Theta_g)$. Let $N_k(e_n \mid \Theta_g)$ be the set that contains entity e_n and its $k-1$ nearest neighbors in the entities' vector space. The method we propose for building an approximation of the predicted graph works as follows. First, select an arbitrary subset of triples $\mathcal{T}_0 \subset \mathcal{T}$ such that we are able to find an arbitrarily sufficient number of triples classified as true facts by the embedding model g,

$$\mathcal{T}_0^+ = \{x_{h,r,t} \in \mathcal{T}_0 \mid g(x_{h,r,t}) = 1\}.$$

Then, build an extended set of triples from the initial set $\mathcal{T}_0^+$ of positive triples by corrupting both head and tail entities with its k nearest neighbors in the entities' vector space, including the original entity itself,

$$\mathcal{T}_0^k = \{x_{h',r,t'} \in \mathcal{T}_0^+ \mid e_{h'} \in N_k(e_h \mid \Theta_g) \wedge e_{t'} \in N_k(e_t \mid \Theta_g)\}.$$

Finally, make inference of the triples in the extended set $\mathcal{T}_0^k$ using the embedding model to build an approximation of the predicted graph $\hat{\mathcal{G}}$,

$$\hat{\mathcal{G}}_0^k = \{x_{h,r,t} \in \mathcal{T}_0^k \mid g(x_{h,r,t}) = 1\},$$

composed of all triples in the extended set predicted as true facts. We call this method BUILD-PREDICTED-TRUE-SET_ENTITY-SIM, and present it in the form of a function in Algorithm 2.

The intuition behind the method is that, by corrupting triples predicted as true facts with its nearest neighbors in the embedding vector space, we generate triples that are more likely to be classified as true facts as well, since the corrupted entities will tend to be replaced by similar ones in the embedding space. In other words, by using the BUILD-PREDICTED-TRUE-SET_ENTITY-SIM method, we are building a more complete approximation for the predicted graph $\hat{\mathcal{G}}$ than one would randomly sampling triples and checking if they were predicted as true facts, in the sense that we will tend to have a larger number of true triples for a fixed number of inferred triples.

Algorithm 2 Predicted Graph via Entity Similarity

1: **function** BUILD-PREDICTED-TRUE-SET_ENTITY-SIM(g, $\mathcal{T}$, k, $|\mathcal{T}_0|$)
2: Sample a set of triples $\mathcal{T}_0 \subset \mathcal{T}$ of arbitrary size $|\mathcal{T}_0|$
3: $\mathcal{T}_0^+ \leftarrow$ BUILD-PREDICTED-TRUE-SET(g, $\mathcal{T}_0$)
4: $\mathcal{T}_0^k \leftarrow \{\}$
5: **for all** $x_{h,r,t} \in \mathcal{T}_0^+$ **do**
6: $N_k(e_h \mid \Theta_g) \leftarrow \{e_h\} \cup \{(k-1)$ nearest neighbors of e_h given $\Theta_g\}$
7: $N_k(e_t \mid \Theta_g) \leftarrow \{e_t\} \cup \{(k-1)$ nearest neighbors of e_t given $\Theta_g\}$
8: $\mathcal{T}_0^k \leftarrow \mathcal{T}_0^k \cup \left[N_k(e_h \mid \Theta_g) \times \{r_r\} \times N_k(e_t \mid \Theta_g)\right]$
9: $\hat{\mathcal{G}}_0^k \leftarrow$ BUILD-PREDICTED-TRUE-SET(g, $\mathcal{T}_0^k$)
10: **return** $\hat{\mathcal{G}}_0^k$

Refer to Algorithm 1 for the definition of BUILD-PREDICTED-TRUE-SET.

XKE-PRED can be adapted to generate the predicted graph using BUILD-PREDICTED-TRUE-SET_ENTITY-SIM by modifying line 15 in Algorithm 1, passing an arbitrary number of nearest neighbors k and an arbitrary size of sampled triples $|\mathcal{T}_0|$ as additional arguments. Since $|\mathcal{T}_0|$ and k influence directly on the size of the resulting predicted graph $\hat{\mathcal{G}}_0^k$, from which SFE extracts features, these parameters influence directly on the number of features (path types) found. The greater $|\mathcal{T}_0|$ and k, the more features we tend to find, and the more information the interpretable classifier has to try to model the black box. On the other hand, the greater the number of features, the larger the explanations tend to be. Thus, the optimal tuning of these parameters depends on the position that the user is willing to take regarding the trade-off between predictive accuracy and interpretability.

In practice, there may be cases where the machine learning practitioner has access to the embedding model's original training data. In this scenario, the original training data can be used as $\mathcal{T}_0$, instead of a randomly sampled set of triples. In fact, one could argue that, since positive training triples tend to be classified as correct by the model (by the learning process definition), this turns out to be a more effective approach for building a larger predicted graph.

Finally, we make three remarks about the entity similarity method proposed here. First, the idea of using nearest neighbors to navigate the training data is not new in the pedagogical setting (ETCHELLS; LISBOA, 2006). Second, the usage

of the embedding layer to search for nearest neighbors deviates from the original pedagogical framework of not having access to the black box's internal structure. However, notice that XKE-PRED does not depend on this technique to be applied and, more importantly, that the method remains model-agnostic even when the predicted graph is generated via entity similarity, because the embedding layer is common to all embedding models. Third, when the KG provides type constraints (mentioned in Subsection 2.2.1), we can use them to decrease the number of possible triples to be inferred, but it does not dismiss the usefulness of the method proposed here. As noted by Nickel et al. (2015), "even amongst the syntactically valid triples [i.e., triples that respect type constraints], only a tiny fraction are likely to be true." Our method is originally formulated in the more general case where we do not have type constraints, but it can be easily adapted to use them by only allowing entities that respect the constraints in the initial sampling and nearest neighbors stages (lines 2 and 6-7 of Algorithm 2, respectively).

4.2.2 Explanations Based on Predicted Features

Note that applying SFE to the data generated by the original classifier involves generating features for a given triple based on other triples predicted as true facts by the embedding model. Thus, an interesting characteristic of XKE-PRED is that explanations depend on other predictions (hence, its name). By looking at an explanation, one is analyzing not how features from the real world influence the model's predictions, but how the model organizes its internal representation. In other words, the method captures correlations between the model's predictions and outputs explanations that allow the user to assess their internal coherence.

One must question whether this is the best way to interpret a prediction. One may be more interested in understanding how patterns from the real world influence the embedding's decisions, and not solely if the embedding's internal representations make sense. To handle that, we propose another method in the next section, that allows for the usage of an external source of interpretable features in explanations.

4.3 Explaining Knowledge Embedding Models with Observed Features (XKE-TRUE)

In this section, we propose a variation of XKE-PRED that assumes an *external source of knowledge* besides the embedding model, regarded as *ground truth* of our relational domain, from which we extract interpretable features. We call this approach *explaining knowledge embedding models with observed features* (XKE-TRUE). The motivation behind it is that we want to explain the black box's predictions based on *real* features, instead of predicted ones (thus the word "TRUE" in its acronym).

Following the notation used in Section 4.2, let $\mathcal{G}$ be a set of triples that represent the ground truth for a relational setting, acquired from an external source aside from the embedding model. XKE-TRUE constructs a set of examples

$$\mathbb{D} = \{(\text{SFE}(x_{h,r,t} \mid \mathcal{G}), g(x_{h,r,t})) \mid x_{h,r,t} \in \mathcal{D}\}$$

from an arbitrary set of triples $\mathcal{D} \in \mathcal{T}$. Then, the procedure trains an interpretable classifier $g'' \colon \mathcal{P}(\Pi_{\mathcal{G}}) \mapsto \{0, 1\}$ using $\mathbb{D}$, from which it draws explanations. Algorithm 3 presents a more detailed version of the method in the form of a procedure.

Algorithm 3 XKE-TRUE

1: **procedure** XKE-TRUE(g, $\mathcal{T}$, $\mathcal{G}$)
2: Arbitrarily selected a set of triples $\mathcal{D} \subseteq \mathcal{T}$
3: $\mathbb{D} \leftarrow$ BUILD-TRAINING-SET(g, $\mathcal{D}$, $\mathcal{G}$)
4: Using $\mathbb{D}$, train an interpretable classifier $g'' \colon \mathcal{P}(\Pi_{\mathcal{G}}) \mapsto \{0, 1\}$
5: Draw explanations from g''

Refer to Algorithm 1 for the definition of BUILD-TRAINING-SET.

When compared to XKE-PRED, the main difference of this method is that $\mathcal{G}$ contains information about a set of instances, and, therefore, the explanation to each prediction from the embedding model is based on a set of observations from the real world. Put in another way, when one analyzes explanations from XKE-PRED, one is assessing correlations between the embedding model's predictions and features from real data.

In practice, the most meaningful and effective way of applying XKE-TRUE may be to use all available data for both training the embedding model and extracting features (although the method does not impose this restriction, and $\mathcal{G}$ can be whatever source of information we regard as ground truth at the moment of inference). It is effective because we are using the maximum number of instances in the training data for both the original and the interpretable classifiers; and it is meaningful because, specifically, we are generating explanations whose features were extracted from the same data of which the embedding model was learned (i.e., the method will give explanations that are a function of paths found in the same set of triples that the model was learned).

4.4 Discussion

We refer to the general neuro-symbolic integration framework presented here, of extracting interpretable features and using them to train an interpretable classifier to explain embedding models, as XKE. Despite of the similarities between each XKE variant, their applications differ in practice, depending on the the scenario in which the machine learning practitioner finds herself. For instance, a classic use case for XKE-PRED is in auditing. When someone or some enterprise is auditing a machine learning model, it is often the case where they do not have access to the original training data. In this scenario, one may be more inclined to explain the embedding model using XKE-PRED rather than XKE-TRUE, since one may not have access to any external source of information to feed the latter method. Conversely, one may prefer to use XKE-TRUE in the case where the machine learning practitioner is the one who will train the embedding model and evaluate its reliability. Since in this scenario explanations will be based on the same data used to train the embedding model, the user may prefer to base explanations on those kinds of correlations.

Further, one may have noticed that taking into account dataset noise is of extreme importance when analyzing XKE-TRUE results: if the data distribution of the external source of information that one is using to extract interpretable features is little representative of the reality one is trying to model, the correlations

presented by the method may influence the user into thinking that the embedding model is flawed. Hence, data exploration of the ground truth data is essential. Additionally, dataset noise in training data dictates the quality of the embedding model. Because of this, when using XKE to compare different embedding models, one must be careful not to generalize the conclusions drawn in one particular dataset to an entire embedding model class (i.e., before stating that TransE is not capturing a particular behavior, one must first ensure that the data allows such behavior to be learned).

5 Experiments

Here we present and compare experimental results for both XKE variants. The datasets used in our experiments are described in Table 2. FB13 is a subset of Freebase (BOLLACKER et al., 2008), introduced by Bordes et al. (2013), and NELL186 is a subset of NELL (MITCHELL et al., 2015), introduced by Guo et al. (2015). For each dataset, we trained the TransE and ANALOGY embedding models, presented in Subsections 2.4.2.2 and 2.4.2.3, respectively. Then, we applied XKE-TRUE and XKE-PRED to each resulting model. XKE-PRED was applied with different generated predicted graphs; specifically, we varied the number of nearest neighbors used to corrupt true facts in the entity similarity stage.

5.1 Implementation

We implemented both XKE-TRUE and XKE-PRED in a software package which we call XKE, available at <https://github.com/arthurcgusmao/XKE>. The software allows one to execute each main stage of both XKE variants, i.e., learn an embedding model, extract interpretable features to build a training set, and learn an interpretable model able to explain the embedding one, in a fully automated manner. With modifications, our implementation uses the core of OpenKE (HAN et al., 2018), an open source package for learning embedding models of KBs, to learn the black box models, and the original SFE implementation (GARDNER; MITCHELL, 2015) to extract interpretable features.

5.2 Evaluation Criteria

To measure the quality of the explanations, we use metrics that reveal performance and rules' format. The main performance metrics are *fidelity* and *accuracy*. Fidelity defines the ability of the pedagogical method to mimic the behavior of the embedding model; accuracy defines the weighted rules' ability of correctly predicting real data. Regarding rules format, we report the *mean number*

Table 2 – Datasets.

Dataset	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{R}\rvert$	$\lvert\mathcal{T}\rvert$	**Train**	**Valid**	**Test**
FB13	75,043	13	$73 \cdot 10^9$	316,232	5,908	23,733
NELL186	14,463	186	$39 \cdot 10^9$	31,134	5,000	5,000

Datasets used in the experiments. $\lvert\mathcal{E}\rvert$ denotes the number of entities, $\lvert\mathcal{R}\rvert$ the number of relations, and $\lvert\mathcal{T}\rvert$ the number of all possible triples in each dataset. **Train**, **Valid** and **Test** denote the number of triples in the training, the validation, and the test set, respectively. Both datasets contain one negative example per positive one, for validation and test, not included in the count.

of rules (path types with weight different than zero) and the *mean body rule length* (number of edges in each path). All metrics are described in more detail in the rest of this section.

5.2.1 Performance Metrics

5.2.1.1 Fidelity

Fidelity allows one to assess the ability of XKE in modeling the embedding model. Formally, let $\hat{y}_{\text{g}}$ be the black box model's prediction for an input triple and $\hat{y}_{\text{XKE}}$ be XKE's prediction for the same triple. Fidelity is defined as the ratio of prediction matches, i.e., the probability of XKE outputting the same prediction as the embedding model:

$$\text{Fidelity} = \mathbb{P}(\hat{y}_{\text{XKE}} = \hat{y}_{\text{g}}).$$

In practice, usually one estimates fidelity from the classifier's results in a test set of triples $\mathcal{D}_{\text{test}}$, used only to assess the performance of the classifiers:

$$\text{Fidelity} \mathrel{\widehat{=}} \sum_{x \in \mathcal{D}_{\text{test}}} \frac{1}{|\mathcal{D}_{\text{test}}|} \cdot \Big(1 - |\hat{y}_{\text{XKE}} - \hat{y}_{\text{g}}|\Big).$$

In our experiments, we used FB13 and NELL186 default test sets as $\mathcal{D}_{\text{test}}$ to estimate all metrics.

5.2.1.2 Accuracy

Accuracy defines a model's ability of correctly predicting real data. Formally, let $\hat{y}$ be the prediction of a classification model for an input instance and y be a

binary random variable that models the existence of the same instance, as in Section 2.4.1. Accuracy is defined as the ratio of correct predictions, i.e., the probability of the classification model correctly predicting the existence of a triple:

$$\text{Accuracy} = \mathbb{P}(\hat{y} = y).$$

As with fidelity, the metric is usually estimated from a test set $\mathcal{D}_{\text{test}}$ as follows:

$$\text{Accuracy} \mathrel{\widehat{=}} \sum_{x \in \mathcal{D}_{\text{test}}} \frac{1}{|\mathcal{D}_{\text{test}}|} \cdot \Big(1 - |\hat{y} - y|\Big).$$

5.2.1.3 F1 Score

In addition to fidelity and accuracy, we report their respective F1 score. The F1 score is a way of representing both *precision* (proportion of correct predictions considering only positive predictions) and *recall* (proportion of correct predictions considering only positive examples) and finding a balance between them. In order to ensure that the score will be high only when both metrics are high, it uses the *harmonic mean* between precision and recall.

Formally, let TP be the number of positive examples predicted correcly, FP be the number of negative examples predicted incorrectly, and FN be the number of positive examples predicted incorrectly. We calculate the F1 score as follows:

$$\text{F1} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}.$$

The F1 score is especially useful when there are large class imbalances, since in these cases accuracy cannot be considered a realiable metric. In our experiments, we use F1 specifically because, even though the validation and test sets of the datasets are balanced with a 1-to-1 ratio, this may not be the case for fidelity—where our label of reference becomes the embedding model's predictions—, where there is no guarantee that the original balance will be preserved.

5.2.1.4 Filtered and Weighted Performance Metrics

In order to better understand the results of XKE, we add two modified versions of the original performance metrics, namely, *accuracy (or fidelity) filtered*

for examples with # features > 0 and *accuracy (or fidelity) weighted by the # features*. The former is calculated using a subset of triples for which SFE was able to find a number of features (number of path types between the head and the tail entities) greater than zero; the latter is calculated in the form of a weighted average, in which the weight of each instance corresponds to its number of features found by SFE.

Formally, let $|\Pi_{h,r,t|\mathcal{G}}| = |\text{SFE}(x_{h,r,t} \mid \mathcal{G})|$ correspond to the number of features SFE found for a triple $x_{h,r,t}$, let $\mathcal{D}_{\text{test}}$ be a test set of triples, and let $\mathcal{D}_{\text{filter}} = \left\{ x_{h,r,t} \in \mathcal{D}_{\text{test}} \;\middle|\; |\Pi_{h,r,t|\mathcal{G}}| > 0 \right\}$ denote a subset of triples filtered from $\mathcal{D}_{\text{test}}$. Following the notation introduced in Subsections 5.2.1.1 and 5.2.1.2, we define and estimate the two modified versions of accuracy and fidelity from the test set as follows:

$$\text{‘Fidelity filtered for examples with \# features} > 0\text{’} \mathrel{\widehat{=}} \sum_{x \in \mathcal{D}_{\text{filter}}} \frac{1}{|\mathcal{D}_{\text{filter}}|} \cdot \Big(1 - |\hat{y}_{\text{XKE}} - \hat{y}_{\text{g}}|\Big),$$

$$\text{‘Fidelity weighted by the \# features’} \mathrel{\widehat{=}} \sum_{x \in \mathcal{D}_{\text{test}}} \frac{|\Pi_{h,r,t|\mathcal{G}}|}{\sum\limits_{x \in \mathcal{D}_{\text{test}}} |\Pi_{h,r,t|\mathcal{G}}|} \cdot \Big(1 - |\hat{y}_{\text{XKE}} - \hat{y}_{\text{g}}|\Big),$$

$$\text{‘Accuracy filtered for examples with \# features} > 0\text{’} \mathrel{\widehat{=}} \sum_{x \in \mathcal{D}_{\text{filter}}} \frac{1}{|\mathcal{D}_{\text{filter}}|} \cdot \Big(1 - |\hat{y} - y|\Big),$$

$$\text{‘Accuracy weighted by the \# features’} \mathrel{\widehat{=}} \sum_{x \in \mathcal{D}_{\text{test}}} \frac{|\Pi_{h,r,t|\mathcal{G}}|}{\sum\limits_{x \in \mathcal{D}_{\text{test}}} |\Pi_{h,r,t|\mathcal{G}}|} \cdot \Big(1 - |\hat{y} - y|\Big),$$

5.2.2 Rules Format Metrics

In this subsection we cover metrics that give information about the format of our explanations. These are metrics that can be used to assess interpretability; however, one must not forget that these are objective measures which ignore the subjective nature of interpretability (FREITAS, 2014). In order to account for that, we also offer a qualitative analysis, that is of a more subjective character. Each rules format metric is formally defined as follows.

5.2.2.1 Mean Number of Rules

The *mean number of rules* represents each explanation's average number of *active rules* for which the logistic regression model (logit) assigns a weight different than zero. We call a rule (or feature) active if its feature value is 1, i.e., if SFE found the path type that corresponds to that feature between the entities of the respective triple. Formally, let w_{π_i} be the weight that the logit assigns for the path type π_i and $\Pi_{h,r,t|\mathcal{G}} = \text{SFE}(x_{h,r,t} \mid \mathcal{G})$ correspond to the set of path types SFE found for a triple $x_{h,r,t}$. Accordingly, let

$$\Pi^{(w\neq 0)}_{h,r,t|\mathcal{G}} = \{\pi_i \in \Pi_{h,r,t|\mathcal{G}} \mid w_{\pi_i} \neq 0\}$$

denote the set of active features for $x_{h,r,t}$ whose weight is not zero. We estimate the mean number of rules, from a test set $\mathcal{D}_{\text{test}}$, by calculating:

$$\text{'Mean number of rules'} \mathrel{\widehat{=}} \sum_{x_{h,r,t}\in\mathcal{D}_{\text{test}}} \frac{\left|\Pi^{(w\neq 0)}_{h,r,t|\mathcal{G}}\right|}{|\mathcal{D}_{\text{test}}|}.$$

5.2.2.2 Mean Body Rule Length

The *mean body rule length* corresponds to the average number of relations (edges) in each explanation's active rule. Put in another way, it represents the average number of relations in each instance feature whose value is 1 and whose weight is different than zero. Following the notation introduced in Subsection 5.2.2.1, let $|\pi_i|$ denote the number of edges in a path type π_i. The mean body rule length estimate can be defined as:

$$\text{'Mean body rule length'} \mathrel{\widehat{=}} \frac{\sum_{x_{h,r,t}\in\mathcal{D}_{\text{test}}} \sum_{\pi_i\in\Pi^{(w\neq 0)}_{h,r,t|\mathcal{G}}} |\pi_i|}{\sum_{x_{h,r,t}\in\mathcal{D}_{\text{test}}} \left|\Pi^{(w\neq 0)}_{h,r,t|\mathcal{G}}\right|}.$$

5.3 Model Training

The TransE models were trained via grid search following Nguyen et al. (2016). Negative examples were generated using a Bernoulli distribution, as described in Subsection 2.4.2.1, with a 1-to-1 ratio (1 negative example was generated

for each positive one); training was limited to 1000 epochs, in each epoch the entire training data was split uniformily and randomly into 100 mini-batches; SGD with RMSProp adaptative learning rate was used to minimize the pairwise ranking loss over the training set. In both datasets, the better accuracy on the validation set was obtained with the following hyperparameters: a learning rate $\eta = 1$, the ℓ_2 norm (used in the calculation of the score function f^{TransE}), and a margin $\gamma = 1$ (used in the calculation of the pairwise ranking loss function). The optimal number of dimensions was $k = 100$ for FB13 and $k = 50$ for NELL186.

We trained ANALOGY via grid search. Negative examples were generated with a 6-to-1 ratio (we also experimented with 3-to-1, but it had worse performance), where, for each positive example, 4 negative examples were generated by corrupting either the head or the tail (entity corruption) and 2 negative examples were generated by corrupting the relation. For the negative examples that were generated via entity corruption, using the Bernoulli distribution procedure was only applied for FB13, since it did not help for NELL186. Training was initially limited to 500 epochs, which was enough for NELL186. For FB13, however, the best results were achieved with 2000 epochs. SGD with AdaGrad adaptative learning rate was used to minimize the log loss. The better accuracy on the validation set was obtained with the following hyperparameters: a number of dimensions $k = 200$, a learning rate $\eta = 0.1$, and a mini-batch size $B = |\mathcal{D}_{\text{train}}|/100$, where $\mathcal{D}_{\text{train}}$ represents the training set, in FB13; and $k = 200$, $\eta = 0.01$, and $B = |\mathcal{D}_{\text{train}}|/10$ in NELL186.

XKE-PRED and XKE-TRUE were applied following the procedures proposed in Chapter 4. The same set of true facts used for training the embedding models was used as the external source of information for XKE-TRUE and for initializing $\mathcal{T}_0$ in XKE-PRED. As with the embeddings, negative examples for the interpretable model were generated via the Bernoulli distribution, with a 2-to-1 ratio. We extracted features for each training instance using SFE, of which details are discussed in the next paragraph. Next, we trained a logistic regression model (logit) for each relation. The logit was trained using SGD to minimize the log loss with elastic net regularization. Our implementation was set to perform a grid search for the logit and automatically select the model that achieved the best fidelity on the validation set. The following parameter ranges were used: elastic net ℓ_1

regularization ratio $\lambda = \{0.1, 0.5, 0.7, 0.9, 0.95, 0.99, 1\}$ ($\lambda = 0$ corresponds to ℓ_2 only; $\lambda = 1$ corresponds to ℓ_1 only), regularization weight $\alpha = \{10^{-2}, 10^{-3}, 10^{-4}\}$ (the term that multiplies the regularization in the loss function), and stopping criteria $\epsilon = 10^{-3}$ (minimal loss improvement necessary to continue learning). In addition, learning was set to balance classes by adjusting class weights inversely proportional to their frequencies in the input data.

The feature extraction performed by SFE in our experiments had two limitations. First, the maximum body rule length was limited to four, due to computational performance reasons. Although this limitation reduces the number of paths found by SFE, it helps with interpretability with the argument that smaller rules are easier to understand. The second limitation is that a node must have a maximum number of 100 outgoing edges in order for that edge type to be expanded during breadth-first search (BFS). The idea is to control the exponential time complexity of BFS (GARDNER; MITCHELL, 2015). This second limitation happens to be relevant to our experiments, as we discuss in the next section.

5.4 Results

We present the results of both XKE-PRED and XKE-TRUE in Tables 4 and 5, together with the respective embedding models' results. TransE achieved greater accuracy in FB13, whereas ANALOGY was best in NELL186. In addition to the embedding models, we trained an interpretable model directly on the original datasets, of which results are shown in Table 3. As in XKE, the interpretable model consists of a logistic regression model learned over a dataset of which features were extracted using SFE.

Looking at XKE-PRED's results, we see that the entity similarity method worked well to increase the size of $\hat{\mathcal{G}}_0^k$ in all cases. It also increased the number of extracted features per example for the TransE models. It did not, however, increase the percentage of examples with at least one extracted feature, which was relatively low in all cases, including when an external source was used (XKE-TRUE). This turns out to be a relevant problem because, without features, the explanation for a given example is defined only by the bias term of the logistic regression,

compromising both fidelity and interpretability.

Considering interpretability, the mean body rule length is less than four for all cases (by constraint), indicating that the Horn clauses whose weight is greater than zero are short, which we consider easily interpretable. The mean number of rules in FB13 is small, favoring interpretability, but it increases in NELL186, going up to almost 160 in XKE-PRED for TransE. In the latter case, if a user finds explanations with such a number of rules poorly interpretable, a possible alternative may be to force higher logit λ values (making the elastic net regularization closer to LASSO), with the price of sacrificing fidelity.

Regarding fidelity, both XKE-TRUE and XKE-PRED achieved over 80% for examples with at least one feature (filtered results), with the exception of ANALOGY in FB13. However, this number drops significantly when we consider all examples, due to the large proportion of examples without features. Additionally, in general, there is little or no improvement comparing the metrics weighted by the number of features with the filtered ones. These results indicate that the probability of correctly classifying an example increases considerably from zero to one feature extracted, and, after that, the impact gets less relevant as we add more features to the same example. Thus, by looking at the number of examples with number of features greater than zero, we can understand why XKE-PRED had, on average, lower fidelity than XKE-TRUE. Initially, we would expect the former to achieve superior fidelity because it predicts the embedding model decisions using a graph defined by the embedding itself. The fact that we were not able to increase the percentage of examples with at least one feature, even when extracting them from graphs with almost 50 times the number of true facts than the original graph (e.g., XKE-PRED with $k = 7$ nearest neighbors compared to XKE-TRUE, both for ANALOGY in NELL186), possibly indicates an internal inconsistency in the embedding model with regard to relational modeling in the way that SFE does. In other words, information encoded by these embedding models may not be representable in the form of weighted Horn clauses. Another possibility is that the constraints imposed over SFE do not work well with graphs generated using XKE-PRED's entity similarity method.

For accuracy, the values achieved are very close to the corresponding fidelity

value (with the exception of ANALOGY in FB13), even when the interpretable model was learned using the predictions of the embedding model as labels. In fact, the best practice before considering to apply XKE is to first compare the embedding model's results with the results of an interpretable model learned directly on the original data (Table 3), which we refer to as 'SFE + logit' in the following paragraphs. We train 'SFE + logit' using labels from the training data, aiming for accuracy instead of fidelity. The idea is to check whether it is possible to have an interpretable classifier that is at least as accurate as an embedding one.

When the embedding model's accuracy is higher than that of 'SFE + logit', one may choose to use the embedding model for prediction and XKE for explanation. However, the option of using an interpretable model alone is not yet discarded, since it depends on the accuracy-interpretability trade-off position one is willing to take. For instance, consider the case of XKE-TRUE for ANALOGY learned in NELL186. The embedding model achieved 92.78% accuracy, whereas 'SFE + logit' achieved 90.41%. However, by using the embedding model and XKE-TRUE (the best XKE variant for this case) to interpret it, one is able to interpret 91.01% of the predictions. On the other hand, were one to use 'SFE + logit', one would be able to interpret all predictions. In this case, it is up to the machine learning practitioner to decide which alternative best suits the application and the users in question.

When the accuracy of 'SFE + logit' is higher than that of the black box, the best option is to choose the interpretable model for both predicting and explaining. An example of this scenario is TransE in NELL186, in which the embedding model achieved 86.40% accuracy, less that the over 90% achieved by 'SFE + logit'.

Finally, from the results we see that the F1 score in the majority of cases was relatively close to their corresponding metric (accuracy or fidelity), which means that the interpretable models were able to achieve balanced performances in precision and recall for the positive class, even with possible class imbalances cause by the embedding model's behavior.

Table 3 – Results of SFE + logit.

Dataset	**FB13**	**NELL186**
Accuracy	56.62	90.41
Accuracy (filt. examples with # features > 0)	71.88	93.88
Accuracy (weighted by # features)	78.97	97.86
F1	51.30	88.56

Results for learning a logistic regression over features extracted by SFE and the original datasets' labels (labels correspond to the existence of the triple in the respective datasets). In other words, these results correspond to applying an interpretable classifier directly to original data, instead of trying to use it to explain the labels of a black box classifier.

5.5 Qualitative Analysis

Now we discuss some examples of predictions made by both XKE variants, for TransE and ANALOGY, in FB13. Results presented in this section were selected so as to represent a variety of cases that the user might face, in order to demonstrate how one can analyze the output of each method. Input triples are shown in Tables 6 and 7, alongside with their explanations (weighted rules and the bias term), with the interpretable classifier's score, and with the labels predicted by the embedding model. Each example receives an ID between #1 and #10 that is used as reference in the rest of this section.

There are two scenarios that can arise when we compare XKE's prediction with the embedding model's one: they may match or differ. In the former case, one can use XKE to explain the embedding model's result. In the latter, one cannot take XKE's active rules into account as explanations for that particular prediction. Since the logit's output is in the form of a probability, we say that the predictions match if either the embedding model predicts a triple as a true fact and XKE's prediction is greater than 0.5, or the embedding model's prediction indicates a false fact and XKE's prediction is smaller than 0.5. In addition, one may argue that the closer the logit's probability is to the embedding model's prediction, the more one can take into account XKE's explanation, i.e., the more one is inclined to believe that XKE's set of active rules are characteristics of the input instance that help us to explain its predicted label. In turn, the more one takes into account the

explanations, the more they influence the user's trust in the model.

Examples #1, #2, #3, #4, #6, #7, #8 and #9 are instances of prediction matches. Of them, #4, #7 and #8 represent weak matches: XKE's prediction probability is close to the 0.5 threshold.

In example #1, the reasons why the religion of henry_the_lion is catholicism make sense in practice. XKE's explanation consists of the two rules that point to the same fact, i.e., Henry is Catholic because his children have the same religion as him. These are the sort of correlations that we want our embedding model to capture.

Examples #6 and #9 are instances that also help us build trust in the model: in #6, XKE outputs the explanation that Nicolae Ceaușescu is believed to have died by firearm because his spouse died for the same reason; in #9, the entity alexander_hurley is related to entity singer through relation profession because his spouse has the same profession.

Example #2 is yet another instance that may help to build trust in the model, depending on the behavior that the user expects. XKE outputs two active rules; their interpretation is that one is likely to belong to a certain institution if other persons who have the same religion or the same profession as the person in question also belong to that institution.

Contrary to the previously discussed examples, #3 is a case that may decrease one's trust in the model. XKE outputs several active rules that are correlated with the negative prediction; the two more relevant active rules point to an undesirable correlation: if D is the cause of death of another person who has the same gender of a person P or P's parents, then P is less likely to have died for D. Ideally, we would want these rules to have a coefficient that is closer to zero, so that they would not be correlated to neither positive nor negative examples.

Finally, examples #5 and #10 are instances where the interpretable classifier is not capable of elucidating the given prediction, since the predictions diverge.

Table 4 – Results of both XKE variants for TransE.

Dataset	**FB13**				**NELL186**			
XKE variant[1]	**TRUE**	**PRED$_3$**	**PRED$_5$**	**PRED$_7$**	**TRUE**	**PRED$_3$**	**PRED$_5$**	**PRED$_7$**
Embedding Accuracy (TransE)	82.55				86.40			
# Positive triples in $\mathcal{G}$ (XKE-TRUE) or $\hat{\mathcal{G}}_0^k$ (XKE-PRED)	322k	830k	1,668k	2,658k	36k	196k	524k	987k
$\hat{\mathcal{G}}_0^k$ positive over predicted ratio	-	0.286	0.207	0.168	-	0.604	0.581	0.558
# Features per example	2.91	0.91	1.34	1.79	70.66	159.54	249.86	337.41
% Examples with # features > 0	54.73	33.83	37.88	41.81	50.01	39.39	45.57	51.87
Explanation Mean # Rules (for explanations with size > 0)	2.29	2.19	2.70	2.57	105.30	51.33	159.02	158.87
Explanation Mean Body Rule Length	3.09	3.00	2.87	2.82	3.86	3.78	3.89	3.89
Fidelity	73.26	66.65	74.36	69.99	86.55	77.00	74.94	75.64
Fidelity (filtered for examples with # features > 0)	80.52	84.30	85.74	83.28	87.02	85.00	83.07	84.47
Fidelity (weighted by the # features)	75.21	82.67	84.58	84.80	85.66	88.09	86.24	88.22
Accuracy	73.43	64.58	71.78	68.11	89.10	75.79	76.18	76.44
Accuracy (filtered for examples with # features > 0)	80.78	81.00	82.02	80.34	91.19	84.08	84.30	85.11
Accuracy (weighted by the # features)	71.68	78.42	81.28	82.19	82.12	86.56	89.11	89.41
F1 (Fidelity)	76.66	50.11	71.14	61.13	83.19	61.41	68.07	68.03
F1 (Accuracy)	77.35	49.07	69.16	59.69	86.89	62.66	71.14	70.68

Results of both XKE-TRUE and XKE-PRED for two TransE models; one trained in FB13 and another in NELL186.

[1] XKE-PRED is indexed by the number of nearest neighbors used when generating the predicted knowledge graph, e.g., PRED$_3$ corresponds to results from applying XKE-PRED with a predicted graph generated with the entity similarity method, described in Section 4.2.1, with $k = 3$ nearest neighbors.

Table 5 – Results of both XKE variants for ANALOGY.

Dataset	**FB13**				**NELL186**			
XKE variant[(1)]	**TRUE**	**PRED$_3$**	**PRED$_5$**	**PRED$_7$**	**TRUE**	**PRED$_3$**	**PRED$_5$**	**PRED$_7$**
Embedding Accuracy (ANALOGY)	73.83				92.78			
# Positive triples in $\mathcal{G}$ (XKE-TRUE) or $\hat{\mathcal{G}}_0^k$ (XKE-PRED)	322k	1,288k	2,438k	3,761k	36k	325k	903k	1,771k
$\hat{\mathcal{G}}_0^k$ positive over predicted ratio	-	0.445	0.303	0.238	-	0.807	0.736	0.689
# Features per example	2.91	2.58	2.63	2.57	70.36	178.25	227.66	217.55
% Examples with # features > 0	54.73	50.50	47.74	45.68	50.12	53.91	56.36	56.85
Explanation Mean # Rules (for explanations with size > 0)	3.94	3.40	3.65	3.34	59.45	90.85	77.38	96.17
Explanation Mean Body Rule Length	3.16	3.14	3.02	3.02	3.81	3.84	3.83	3.84
Fidelity	78.67	78.69	77.30	76.44	91.01	87.79	86.58	85.59
Fidelity (filtered for examples with # features > 0)	76.97	79.01	80.20	79.94	92.62	90.82	91.59	91.68
Fidelity (weighted by the # features)	79.93	79.81	79.74	80.61	97.59	95.06	94.38	92.61
Accuracy	72.41	70.67	68.29	67.25	90.57	86.25	85.46	84.93
Accuracy (filtered for examples with # features > 0)	80.54	80.65	81.41	81.03	94.51	89.91	90.92	91.63
Accuracy (weighted by the # features)	83.09	82.62	83.06	82.54	97.90	95.35	94.96	94.44
F1 (Fidelity)	80.65	79.92	75.31	74.85	88.68	82.89	80.51	79.00
F1 (Accuracy)	76.66	74.06	68.71	68.19	89.06	82.03	80.22	79.40

Results of both XKE-TRUE and XKE-PRED for two ANALOGY models; one trained in FB13 and another in NELL186.

(1) XKE-PRED is indexed by the number of nearest neighbors used when generating the predicted knowledge graph, e.g., PRED$_3$ corresponds to results from applying XKE-PRED with a predicted graph generated with the entity similarity method, described in Section 4.2.1, with $k = 3$ nearest neighbors.

Table 6 – Explanations by XKE-TRUE and XKE-PRED (5-NN) for TransE in FB13.

ID	**#1 (XKE-TRUE)**	**#2 (XKE-TRUE)**	**#3 (XKE-TRUE)**	**#4 (XKE-PRED)**	**#5 (XKE-PRED)**
Head	henry_the_lion	chaim_potok	philip_iv_of_france	arthur_murphy	louis_bloomfield
Relation	religion	institution	cause_of_death	profession	ethnicity
Tail	catholicism	mount_holyoke_college	bladder_cancer	writer	judaism
Active rule #1	(0.649) parents^{-1}, religion	(0.718) religion, religion^{-1}, institution	(-2.495) gender, gender^{-1}, cod	(0.160) nationality, profession	(3.458) religion
Active rule #2	(0.500) children, religion	(0.277) profession, profession^{-1}, institution	(-1.290) parents, gender, gender^{-1}, cod	–	
Active rule #3	–	–	(-1.075) spouse^{-1}, gender, gender^{-1}, cod	–	–
Active rule #4	–	–	(-0.845) children^{-1}, gender, gender^{-1}, cod	–	–
Active rule #5	–	–	(-0.612) parents^{-1}, religion, religion^{-1}, cod	–	–
Active rule #6	–	–	(-0.219) children, religion, religion^{-1}, cod	–	–
Bias (logit)	(0.681)	(0.194)	(0.928)	(0.185)	(0.647)
XKE prediction (logit)	0.861767	0.766624	0.003652	0.585188	0.983776
Embedding prediction	1	1	0	1	0

Instances of FB13 predicted by TransE and its corresponding explanations by XKE-TRUE and XKE-PRED (trained using BUILD-PREDICTED-TRUE-SET_ENTITY-SIM with $k = 5$). Explanations are composed of weighted active rules (paths that are present between the given head and tail entities) and the logit's bias. Paths that are not observed for the given triple are dropped, since they do not influence the logit results. The bias term and each path's weight are shown in parenthesis.

Note: cod and nat are abbreviations for the relations cause_of_death and nationality, respectively.

Table 7 – Explanations by XKE-TRUE and XKE-PRED (5-NN) for ANALOGY in FB13.

ID	#6 (XKE-TRUE)	#7 (XKE-TRUE)	#8 (XKE-TRUE)	#9 (XKE-PRED)	#10 (XKE-PRED)
Head	nicolae_ceausescu	alan_turing	philip_iv_of_france	alexander_hurley	bon_scott
Relation	cause_of_death	ethnicity	cause_of_death	profession	profession
Tail	firearm	israelis	animal_attack	singer	composer
Active rule #1	(1.470) spouse, cod	(1.003) nat, nat^{-1}, ethnicity	(-0.803) gender, gender^{-1}, cod	(1.016) spouse, profession	(1.941) profession, profession^{-1}, children
Active rule #2	(1.411) spouse^{-1}, cod	(-0.894) gender, gender^{-1}, ethnicity	(0.445) children^{-1}, gender, gender^{-1}, cod	(0.753) spouse^{-1}, profession	(-0.455) gender, gender^{-1}, children
Active rule #3	–	(0.310) location, nat^{-1}, ethnicity	(0.219) children^{-1}, gender, gender^{-1}, cod	–	–
Bias (logit)	(0.058)	(-0.474)	(0.058)	(0.088)	(0.088)
XKE prediction (logit)	0.949773	0.486347	0.523594	0.865049	0.828401
Embedding prediction	1	0	1	1	0

Instances of FB13 predicted by ANALOGY and its corresponding explanations by XKE-TRUE and XKE-PRED (trained using BUILD-PREDICTED-TRUE-SET_ENTITY-SIM with $k = 5$). Explanations are composed of weighted active rules (paths that are present between the given head and tail entities) and the logit's bias. Paths that are not observed for the given triple are dropped, since they do not influence the logit results. The bias term and each path's weight are shown in parenthesis.

Note: cod and nat are abbreviations for the relations cause_of_death and nationality, respectively.

6 Conclusions and Future Work

In this work, we discussed challenges that arise when trying to interpret embedding models of large knowledge bases and proposed two model-agnostic methods for overcoming them. The methods are original neuro-symbolic integrations that explain predictions from a black box model in terms of weighted paths in a graph; each path is composed of a series of relations, and corresponds to a Horn rule that can be interpreted by users.

The first method, XKE-PRED, uses other predictions from the embedding model as reference for its explanations. Thus, it allows the user to analyze the model's internal coherence through sequences of relations that are commonly related to the existence of a specific type of relationship in the knowledge graph. The second method, XKE-TRUE, uses an external source of information, regarded as ground truth, to provide explainable features. Differently from the first method, XKE-TRUE reveals how paths between two entities present in real data are correlated with the embedding model's predictions.

Both techniques were able to generate a relatively small number of short weighted Horn clauses that are much easier for a human to interpret than the original embedding space. The percentage of examples that have no features extracted accounts for a significant drop in both models' fidelity and interpretability; it is the major point for improvement to be tackled in future work. We expect this initial work to serve as a basis of comparison and inspiration for the development of novel methods for explaining embedding models in KB completion.

Bibliography

ANDREWS, Robert; DIEDERICH, Joachim; TICKLE, Alan B. Survey and Critique of Techniques for Extracting Rules from Trained Artificial Neural Networks. *Knowledge-Based Systems*, v. 8, n. 6, p. 373–389, dez. 1995. ISSN 09507051. Cited in page 43.

AUGASTA, M. Gethsiyal; KATHIRVALAVAKUMAR, T. Reverse Engineering the Neural Networks for Rule Extraction in Classification Problems. *Neural Processing Letters*, v. 35, n. 2, p. 131–150, abr. 2012. ISSN 1370-4621, 1573-773X. Cited in page 43.

BARBIERI, Nicola; BONCHI, Francesco; MANCO, Giuseppe. Who to Follow and Why: Link Prediction with Explanations. In: . [S.l.]: ACM Press, 2014. p. 1266–1275. ISBN 978-1-4503-2956-9. Cited in page 44.

BELLEAU, François; NOLIN, Marc-Alexandre; TOURIGNY, Nicole; RIGAULT, Philippe; MORISSETTE, Jean. Bio2RDF: Towards a Mashup to Build Bioinformatics Knowledge Systems. *Journal of Biomedical Informatics*, Elsevier, v. 41, n. 5, p. 706–716, 2008. Cited in page 28.

BOLLACKER, Kurt; EVANS, Colin; PARITOSH, Praveen; STURGE, Tim; TAYLOR, Jamie. Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. In: . [S.l.]: ACM Press, 2008. p. 1247. ISBN 978-1-60558-102-6. Cited 2 times in pages 19 and 55.

BORDES, Antoine; GLOROT, Xavier; WESTON, Jason; BENGIO, Yoshua. A Semantic Matching Energy Function for Learning with Multi-Relational Data. *Machine Learning*, Springer, v. 94, n. 2, p. 233–259, 2014. Cited 2 times in pages 29 and 31.

BORDES, Antoine; USUNIER, Nicolas; Garcia-Duran, Alberto; WESTON, Jason; YAKHNENKO, Oksana. Translating Embeddings for Modeling Multi-Relational Data. In: *Advances in Neural Information Processing Systems*. [S.l.: s.n.], 2013. p. 2787–2795. Cited 2 times in pages 36 and 55.

BORDES, Antoine; WESTON, Jason. Tutorial, *Embedding Methods for Natural Language Processing*. 2014. Tutorial. Cited in page 26.

CARMONA, Ivan Sanchez; RIEDEL, Sebastian. Extracting Interpretable Models from Matrix Factorization Models. In: *Proceedings of the 2015th*

International Conference on Cognitive Computation: Integrating Neural and Symbolic Approaches-Volume 1583. [S.l.]: CEUR-WS. org, 2015. p. 78–84. Cited in page 44.

CHANDRAHAS; SENGUPTA, Tathagata; PRAGADEESH, Cibi; TALUKDAR, Partha Pratim. Inducing Interpretability in Knowledge Graph Embeddings. *arXiv:1712.03547 [cs]*, dez. 2017. Cited in page 44.

CRAVEN, Mark; SHAVLIK, Jude W. Extracting Tree-Structured Representations of Trained Networks. In: *Advances in Neural Information Processing Systems*. [S.l.: s.n.], 1996. p. 24–30. Cited in page 43.

CUCERZAN, Silviu. Large-Scale Named Entity Disambiguation Based on Wikipedia Data. In: *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. Prague, Czech Republic: Association for Computational Linguistics, 2007. p. 708–716. Cited 2 times in pages 19 and 27.

DORAN, D; SCHULZ, SC; BESOLD, TR. What Does Explainable AI Really Mean? A New Conceptualization of Perspectives. In: *CEUR Workshop Proceedings*. [S.l.: s.n.], 2018. v. 2071. Cited 2 times in pages 20 and 25.

ENGELEN, Jesper; BOEKHOUT, Hanjo; TAKES, Frank. Explainable and Efficient Link Prediction in Real-World Network Data. In: *International Symposium on Intelligent Data Analysis*. Cham: Springer International Publishing, 2016. (Lecture Notes in Computer Science, v. 9897). ISBN 978-3-319-46348-3 978-3-319-46349-0. Cited in page 44.

ETCHELLS, T.A.; LISBOA, P.J.G. Orthogonal Search-Based Rule Extraction (OSRE) for Trained Neural Networks: A Practical and Efficient Approach. *IEEE Transactions on Neural Networks*, v. 17, n. 2, p. 374–384, mar. 2006. ISSN 1045-9227. Cited in page 50.

FRANÇA, Manoel Vitor Macedo; GARCEZ, Artur S. D'Avila; ZAVERUCHA, Gerson. Relational Knowledge Extraction from Neural Networks. In: *Proceedings of the 2015th International Conference on Cognitive Computation: Integrating Neural and Symbolic Approaches-Volume 1583*. [S.l.]: CEUR-WS. org, 2015. p. 146–154. Cited in page 43.

FREITAS, Alex A. Comprehensible Classification Models: A Position Paper. *ACM SIGKDD Explorations*, v. 15, n. 1, p. 10, mar. 2014. ISSN 1931-0145. Cited in page 58.

GARCEZ, A. d'Avila; BESOLD, Tarek R.; RAEDT, Luc De; FÖLDIAK, Peter; HITZLER, Pascal; ICARD, Thomas; KÜHNBERGER, Kai-Uwe; LAMB, Luis C.; MIIKKULAINEN, Risto; SILVER, Daniel L. Neural-Symbolic Learning and Reasoning: Contributions and Challenges. In: *Proceedings of the AAAI Spring Symposium on Knowledge Representation and Reasoning: Integrating Symbolic and Neural Approaches, Stanford*. [S.l.: s.n.], 2015. Cited in page 43.

Garcia-Gasulla, Dario; CORTÉS, Ulises; AYGUADÉ, Eduard; LABARTA, Jesús. Evaluating link prediction on large graphs. In: *Artificial Intelligence Research and Development: Proceedings of the 18th International Conference of the Catalan Association for Artificial Intelligence*. [S.l.: s.n.], 2015. v. 277. Cited 2 times in pages 29 and 30.

GARDNER, Matt; MITCHELL, Tom M. Efficient and Expressive Knowledge Base Completion Using Subgraph Feature Extraction. In: *EMNLP*. [S.l.: s.n.], 2015. p. 1488–1498. Cited 4 times in pages 40, 42, 55, and 61.

GARDNER, Matt; TALUKDAR, Partha; MITCHELL, Tom. Combining Vector Space Embeddings with Symbolic Logical Inference over Open-Domain Text. p. 5, 2015. Cited in page 42.

GETOOR, Lise; TASKAR, Ben. *Introduction to Statistical Relational Learning*. Cambridge, Mass: MIT Press, 2007. (Adaptive computation and machine learning). OCLC: ocm80019804. ISBN 978-0-262-07288-5. Cited in page 19.

GUO, Shu; WANG, Quan; WANG, Bin; WANG, Lihong; GUO, Li. Semantically Smooth Knowledge Graph Embedding. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China: Association for Computational Linguistics, 2015. p. 84–94. Cited 3 times in pages 31, 36, and 55.

HAN, Xu; CAO, Shulin; XIN, Lv; LIN, Yankai; LIU, Zhiyuan; SUN, Maosong; LI, Juanzi. OpenKE: An Open Toolkit for Knowledge Embedding. In: *Proceedings of EMNLP*. [S.l.: s.n.], 2018. Cited in page 55.

KROMPASS, Denis; BAIER, Stephan; TRESP, Volker. Type-Constrained Representation Learning in Knowledge Graphs. In: Springer. *International Semantic Web Conference*. [S.l.], 2015. p. 640–655. Cited in page 28.

LAO, Ni; COHEN, William W. Relational Retrieval Using a Combination of Path-Constrained Random Walks. *Machine Learning*, v. 81, n. 1, p. 53–67, out. 2010. ISSN 0885-6125, 1573-0565. Cited in page 40.

LAO, Ni; MITCHELL, Tom; COHEN, William W. Random Walk Inference and Learning in a Large Scale Knowledge Base. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. [S.l.]: Association for Computational Linguistics, 2011. p. 529–539. Cited in page 40.

LIN, Yankai; LIU, Zhiyuan; LUAN, Huanbo; SUN, Maosong; RAO, Siwei; LIU, Song. Modeling Relation Paths for Representation Learning of Knowledge Bases. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, 2015. p. 705–714. Cited in page 29.

LIPTON, Zachary C. The Mythos of Model Interpretability. In: *Proceedings of the ICML Workshop on Human Interpretability in Machine Learning*. [S.l.: s.n.], 2016. p. 96–100. Cited 2 times in pages 23 and 24.

LIU, Hanxiao; WU, Yuexin; YANG, Yiming. Analogical Inference for Multi-relational Embeddings. In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*. [S.l.: s.n.], 2017. p. 2168–2178. Cited 3 times in pages 36, 38, and 39.

MIKOLOV, Tomas; CHEN, Kai; CORRADO, Greg; DEAN, Jeffrey. Efficient Estimation of Word Representations in Vector Space. *CoRR*, abs/1301.3781, 2013. Cited in page 36.

MILLER, George A. WordNet: A Lexical Database for English. *Commun. ACM*, p. 39–41, nov. 1995. Cited in page 19.

MITCHELL, Tom. *Never-Ending Learning: (660332010-001)*. [S.l.]: American Psychological Association, 2010. Cited in page 19.

MITCHELL, Tom M.; COHEN, William W.; JR, Estevam R. Hruschka; TALUKDAR, Partha Pratim; BETTERIDGE, Justin; CARLSON, Andrew; MISHRA, Bhavana Dalvi; GARDNER, Matthew; KISIEL, Bryan; KRISHNAMURTHY, Jayant; others. Never Ending Learning. In: *AAAI*. [S.l.: s.n.], 2015. p. 2302–2310. Cited in page 55.

MURPHY, Brian; TALUKDAR, Partha; MITCHELL, Tom. Learning Effective and Interpretable Semantic Models using Non-Negative Sparse Embedding. *Proceedings of COLING 2012*, p. 1933–1950, 2012. Cited in page 44.

NGUYEN, Dat Quoc. An Overview of Embedding Models of Entities and Relationships for Knowledge Base Completion. *CoRR*, abs/1703.08098, 2017. Cited 2 times in pages 28 and 34.

NGUYEN, Dat Quoc; SIRTS, Kairit; QU, Lizhen; JOHNSON, Mark. Neighborhood Mixture Model for Knowledge Base Completion. In: GOLDBERG, Yoav; RIEZLER, Stefan (Ed.). *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*. [S.l.]: ACL, 2016. p. 40–50. Cited in page 59.

NICKEL, Maximilian; MURPHY, Kevin; TRESP, Volker; GABRILOVICH, Evgeniy. A Review of Relational Machine Learning for Knowledge Graphs. *Proceedings of the IEEE*, v. 104, n. 1, p. 11–33, 2015. ISSN 0018-9219, 1558-2256. Cited 10 times in pages 19, 26, 27, 31, 32, 33, 34, 40, 41, and 51.

NICKEL, Maximilian; TRESP, Volker; KRIEGEL, Hans-Peter. A Three-Way Model for Collective Learning on Multi-Relational Data. In: *ICML*. [S.l.: s.n.], 2011. v. 11, p. 809–816. Cited in page 31.

QIAN, Richard. *Understand Your World with Bing*. Microsoft, 2013. Disponível em: <https://blogs.bing.com/search/2013/03/21/understand-your-world-with-bing/>. Cited in page 27.

RAEDT, Luc De. *Logical and Relational Learning*. Berlin: Springer, 2008. (Cognitive technologies). OCLC: 249565193. ISBN 978-3-540-20040-6 978-3-540-68856-3. Cited in page 19.

RIBEIRO, Marco Tulio; SINGH, Sameer; GUESTRIN, Carlos. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In: *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. San Francisco, California, USA: ACM, 2016. (KDD '16), p. 1135–1144. ISBN 978-1-4503-4232-2. Cited 5 times in pages 20, 21, 23, 25, and 45.

RUTTENBERG, Alan; REES, Jonathan A; SAMWALD, Matthias; MARSHALL, M Scott. Life Sciences on the Semantic Web: the Neurocommons and beyond. *Briefings in Bioinformatics*, Oxford University Press, v. 10, n. 2, p. 193–204, 2009. Cited in page 28.

SCHUHMACHER, Michael; PONZETTO, Simone Paolo. Knowledge-Based Graph Document Modeling. In: *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*. [S.l.]: ACM Press, 2014. p. 543–552. ISBN 978-1-4503-2351-2. Cited 2 times in pages 19 and 27.

SINGHAL, Amit. *Introducing the Knowledge Graph: Things, not Strings*. Google, 2012. Disponível em: <https://googleblog.blogspot.com/2012/05/introducing-knowledge-graph-things-not.html>. Cited in page 27.

SRINIVASAN, Ashwin; VIG, Lovekesh. Mode-Directed Neural-Symbolic Modelling. In: *Submitted to: The 27th International Conference on Inductive Logic Programming (ILP2017)*. [S.l.: s.n.], 2017. Cited in page 44.

TROUILLON, Théo; WELBL, Johannes; RIEDEL, Sebastian; GAUSSIER, Eric; BOUCHARD, Guillaume. Complex Embeddings for Simple Link Prediction. In: BALCAN, Maria Florina; WEINBERGER, Kilian Q. (Ed.). *Proceedings of The 33rd International Conference on Machine Learning*. New York, New York, USA: PMLR, 2016. (Proceedings of Machine Learning Research, v. 48), p. 2071–2080. Cited in page 36.

WANG, Q.; MAO, Z.; WANG, B.; GUO, L. Knowledge Graph Embedding: A Survey of Approaches and Applications. *IEEE Transactions on Knowledge and Data Engineering*, v. 29, n. 12, p. 2724–2743, dez. 2017. ISSN 1041-4347. Cited 8 times in pages 19, 29, 30, 31, 33, 34, 36, and 38.

WANG, Zhen; ZHANG, Jianwen; FENG, Jianlin; CHEN, Zheng. Knowledge Graph Embedding by Translating on Hyperplanes. In: *Twenty-Eighth AAAI Conference on Artificial Intelligence*. [S.l.: s.n.], 2014. Cited 3 times in pages 36, 37, and 38.

XIAO, Han. KSR: A Semantic Representation of Knowledge Graph within a Novel Unsupervised Paradigm. *arXiv:1608.07685 [cs]*, ago. 2016. Cited in page 44.

XIE, Qizhe; MA, Xuezhe; DAI, Zihang; HOVY, Eduard. An Interpretable Knowledge Transfer Model for Knowledge Base Completion. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. [S.l.: s.n.], 2017. v. 1, p. 950–962. Cited in page 44.

ZILKE, Jan Ruben; MENCÍA, Eneldo Loza; JANSSEN, Frederik. DeepRED – Rule Extraction from Deep Neural Networks. In: *International Conference on Discovery Science*. Cham: Springer International Publishing, 2016. (Lecture Notes in Computer Science, v. 9956). ISBN 978-3-319-46306-3 978-3-319-46307-0. Cited in page 43.