Andrey Ruschel

Explaining Automatic Answers Generated from Knowledge Base Embedding Models

Corrected Version

São Paulo

2022

Andrey Ruschel

Explaining Automatic Answers Generated from Knowledge Base Embedding Models

Corrected Version

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

2022

Autorizo a reprodução e divulgação total ou parcial deste trabalho, por qualquer meio convencional ou eletrônico, para fins de estudo e pesquisa, desde que citada a fonte.

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

Ruschel, Andrey
Explaining Automatic Answers Generated from Knowledge Base Embedding Models / A. Ruschel -- versão corr. -- São Paulo, 2022.
82 p.

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

1.Machine Learning 2.Knowledge Base Embeddings 3.Knowledge Graphs 4.Deep Learning 5.Interpretability I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

Acknowledgements

Returning to the academy after almost two decades past graduation certainly is not a simple endeavor. It represented not only an opportunity to learn an amazing new field but also a bridge that helped me abandon my comfort zone and implement a desired mid-life career change. The efforts of countless hours of study to learn new concepts while remembering forgotten old ones, burning the midnight oil to meet deadlines, were filled with uncertainty and pain, but I feel proud of what I have accomplished in hindsight. Of course, none of that would have been possible without the support of great people I had the opportunity to meet over the years. I would like to offer the most heartfelt thanks to all of them, but especially for:

My advisor, Prof. Dr. Fabio Gagliardi Cozman. Thanks for letting me in, and thanks for all the knowledge, experience, respect, professionalism, and partnership.

To my parents, Nestor Ruschel and Jussara Ruschel (in memoriam), for all the support. My mother watched me beginning this journey but unfortunately left us. I know she is watching me now from wherever she might be.

To Arthur Colombini Gusmão, for setting the seed with his great work and for the endless conversations, either face-to-face or via *whatsapp* message that helped me develop my work.

To Anna Reali Costa. Thank you for keeping your office door open and introducing me to my advisor.

To Gustavo Poletti for the support during the development and writing of the paper we have published.

To Etienne Cartolano Jr. for inviting me to join a fantastic Itaú Data Science team, and Jardel Trento Negrão for further developing the team: The professional experience I gathered helped shape me as a Data Scientist, for sure.

To Rodrigo Monteiro Aquino, for the tons of coffee we had the opportunity to share at the laboratory, and of course, for all the excellent tips and coffee-filled conversations.

To all my good friends from C^2D.

To Itaú-Unibanco for supporting this work through the scholarship program PBI (Programa de Bolsas Itaú).

Abstract

While many chatbot systems rely on templates and shallow semantic analysis, advanced question-answering devices are typically produced with the help of large-scale knowledge bases such as DBpedia or Freebase. Information extraction is often based on embedding models that map semantically rich information into low-dimensional vectors, allowing computationally efficient calculations. When producing new facts about the world, embeddings often provide correct answers that are very hard to explain from a human perspective as they are based on operations performed in the low-dimensional vector space, thus bearing no meaning to human users. Although interpretability has become a central concern in machine learning, the literature so far has focused on non-relational classifiers (such as deep neural networks); embeddings, however, require a whole range of different approaches. In this work we improve an existing method designed to provide explanations for predictions made by embedding models.

Resumo

Atualmente, muitos sistemas de chat automatizados são baseados em templates e análises semânticas superficiais. Sistemas de pergunta-resposta avançados geralmente são construídos sobre bases de conhecimento de larga escala como DBPedia e Freebase. A extração de informações muitas vezes é baseada em modelos de variável latente, onde informações semanticamente ricas são convertidas em representações númericas dentro de espaços vetoriais. Quando utilizados para predizer novos fatos sobre o mundo, estes modelos de variável latente na maioria das vezes são capazes de produzir respostas corretas mas de difícil interpretação sob o ponto de vista humano, dado que são fruto de operações matemáticas dentro deste espaço vetorial. Apesar de interpretabilidade ter se tornado um tema central em aprendizado de máquina, a maioria dos modelos que visam produzir explicações são direcionados para modelos tradicionais e não relacionais (como por exemplo redes neurais profundas); modelos de variável latente para modelagem de bases de conhecimento, no entanto, requerem abordagens totalmente diferentes. Neste trabalho apresentamos um conjunto de melhorias para um modelo existente que visa explicar as predições realizadas por modelos de variável latente para bases de conhecimento.

List of Figures

List of Tables

Contents

1 Introduction

The representation of real-world facts in knowledge bases (KBs) has attracted significant attention in recent years (NICKEL et al., 2015). Such knowledge bases are often structured as knowledge graphs (KGs) that enable several applications like question-answering systems, chatbots, and semantic parsing. As examples of real-world KGs, we can mention NELL (MITCHELL et al., 2015), YAGO (SUCHANEK; KASNECI; WEIKUM, 2007), Freebase (BOLLACKER et al., 2008). Even though such KGs contain billions of facts, they are far from complete, and a broad range of scientific approaches for *knowledge base completion* have been proposed, in particular, approaches connected with *Statistical Relational Learning (SRL)* (GETOOR; TASKAR, 2007).

The state-of-the-art methods that perform knowledge base completion tasks are the so-called *latent feature models*, also known as *embeddings* (WANG et al., 2017), that work by learning low-dimensional vector representations of the semantically rich representation of facts. Entities are then represented by learned vectors of an arbitrary dimension, and relations among those entities are represented by transformations within the same low-dimensional vector space. The representation of entities and their relations are learned during the same training process. One significant advantage of the conversion of entities and relations into *embedded* vectors is that reasoning becomes a numerical task, allowing large-scale KBs to be used in real-world applications. Embeddings are generally very precise in finding relationships between entities, at the expense of interpretability, in the sense that it is challenging for a human to grasp the underlying mathematics that supports a given prediction.

Interpretability of machine learning techniques has always been a concern among scientists, especially more recently with the development of complex (and regarded as *non-interpretable*) machine learning methods such as *deep neural networks* that are applied to solve real-world problems, including several business applications. In 2016, *DARPA* launched the program *"Explainable AI"* (GUNNING,

2016), which only helped galvanize interest in the topic. Although there is no consensus about the definition of the term "interpretability" in the context of machine learning (DORAN; SCHULZ; BESOLD, 2017), several papers are trying to tackle the issue. The majority of them (RIBEIRO; SINGH; GUESTRIN, 2016; LUNDBERG; ALLEN; LEE, 2017) focus on more traditional machine learning tasks such as regression and classification. However, to be able to deal with relational models, these classical approaches are not applicable, and a whole range of different methods is required.

As we will discuss in more detail through this work, the interpretability of machine learning models predictions is very important to help humans to reconcile with those predictions, increasing trust in the models. Let us imagine, for instance, an advanced chatbot answering the question "Is Paris the capital of France?". It may use an embedded KG containing countries, cities, and the relation "*is capital of*" to provide the correct answer. However, the explanation for the reason as to why this answer was positive would probably be hard for a human to grasp because the prediction relies on complex numerical calculations. Our goal here is to enhance methods that produce explanations about the answers made by embedding models, thus allowing chatbots to provide human-understandable explanations on top of the answers.

More precisely, this research provides some improvements to the **XKE (Explaining Knowledge Embeddings)** method, proposed by Gusmão et al. (2018). As we can see in Figure 1, XKE is a model-agnostic explainer system to explain predictions from knowledge base embeddings using **SFE (Subgraph Feature Extraction)**, which itself is an approach to knowledge base completion from the family of graph-based methods. The expression model-agnostic means that the explainer system can be used with any embedding model.

Our novel contributions tackle some of the shortcomings of the XKE method, resulting in:

- Increased fidelity: the ability of the explainer model to mimic the predictions of the classifier being explained correctly;
- More interpretable explanations.

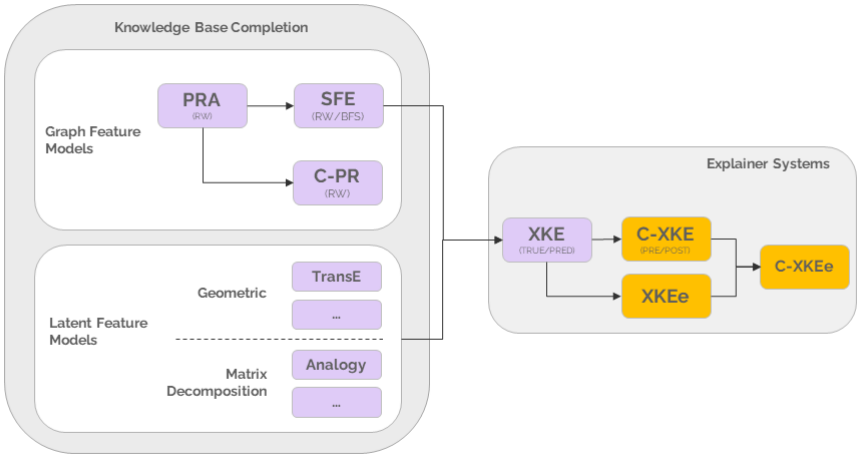

Figure 1 – Overview of the Knowledge Base Completion approaches and Explainer Systems that are discussed in this work.

In Figure 1 we see the two contributions depicted in orange: **C-XKE (Conext-aware XKE)** was designed to produce better and more interpretable explanations, while **XKEe (Enhanced XKE)** aims increasing fidelity. We described part of the contributions presented in this thesis in the paper:

- Andrey Ruschel, Arthur C. Gusmão, Gustavo Poletti, Fabio G. Cozman. **Explaining Answers Produced by Embeddings of Knowledge Bases.** Proceedings of the Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU), pp 325-335, 2019.

In Chapter 2, we provide the required background knowledge. Chapter 3 first addresses in section 3.1 the interpretable method XKE in a more detailed fashion, followed by a broader discussion about interpretability and other related work in section 3.2. In Chapter 4 we detail our novel contributions to foster XKE. Experimental results are presented in Chapter 5, and conclusion and future work in Chapter 6.

2 Background Knowledge

This chapter summarizes the concepts necessary to understand the present work. We start with basic definitions related to *knowledge bases (KBs)* and *knowledge graphs (KGs)*. This discussion is essential to contextualize why KGs are often incomplete. Then, we define the types of tasks connected with *knowledge base completion (KBC)* and the approaches to perform completion. Those are commonly divided into two main families: *graph feature* and *latent feature* models. Even though some hybrid KBC approaches combine both graph and latent feature models, we will see that in this work we make use of a *graph feature* model to produce explanations about predictions made by a *latent feature* model. We close the chapter discussing interpretability, our primary goal in this effort.

2.1 Knowledge Bases and Knowledge Graphs

As pointed out by Nickel et al. (2015), knowledge representations composed of entities and the relations among entities have a long history in topics such as logic and artificial intelligence. Following the RDF Framework (W3C, 2014) loosely, we represent a fact as a triple composed of two entities and one relation: $\langle head, relation, tail \rangle$. In several publications, the *head* is called *subject*, the *tail* is called *object* and the *relation* can be called *predicate*, hence a triple can be written as $\langle subject, predicate, object \rangle$.

For instance, to represent the fact that the team **UIC Flames** plays hockey, one would store in the KB the following triple: $\langle UIC\_Flames, plays, hockey \rangle$. Similarly, the triple $\langle Larry\_Pedrie, is\_coach, UIC \rangle$ represents the fact that **Larry Pedrie** is the coach of UIC.

The combination of a set of triples can be viewed as a directed *multigraph*, where the nodes of the graph represent entities, and the edges of the graph represent the relations connecting the entities, pointing from subjects to objects. Below we see an example of a set of different facts about the UIC Flames ice hockey team,

represented as triples and as a knowledge graph:

⟨ UIC, plays, Hockey ⟩
⟨ UIC, located_in, Chicago ⟩
⟨ Larry Pedrie, is_coach, UIC ⟩
⟨ Ray Staszak, plays_for, UIC ⟩
⟨ Ray Staszak, plays, Hockey ⟩

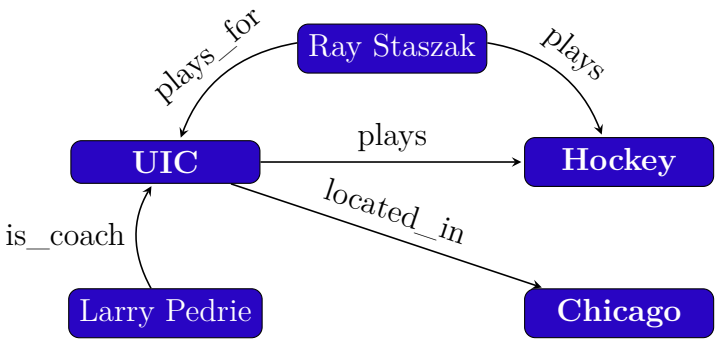

Figure 2 – Example extracted from NELL186.

In the remainder of this work, we will use the terms *knowledge base (KB)* and *knowledge graph (KG)* interchangeably.

KBs are constructed following certain assumptions and characteristics. Here we provide a brief explanation of different types of KB regarding assumptions and creation methods (NICKEL et al., 2015).

• **Open World vs. Closed World Assumption:** in a KB, an existing triple represents a known fact. Regarding missing triples, two approaches are considered: in KBs constructed under the *Open World Assumption (OWA)*, a missing triple represents an unknown fact, whereas, in KBs built under the *Closed World Assumption (CWA)*, a missing triple represents a non-existing (or false) fact.

• **Construction Method:** *curated* KBs are the ones constructed by a group of domain experts, while *collaborative* ones are built by a group of volunteers,

not necessarily experts in the field. KBs can also be constructed using an automated method that can be subdivided into two groups: the first one, in which KBs are created by extracting information from semi-structured field texts from the web, are called *automated semi-structured* approaches. The second one is comprised of methods where KBs are constructed from unstructured web text extraction using some natural language processing method, called *automated unstructured* approaches.

• **Schema:** KG's are *schema-based* when unique identifiers represent entities and relations; *schema-free* KG's allow different identifiers represent the same entity or relation.

2.2 Knowledge Base Completion

Due to the different strategies used to build large-scale knowledge bases, most of them relying on text extraction from the web using automated methods and following schema-free approaches, most KBs suffer from incompleteness and entity ambiguity. As pointed out by West et al. (2014), 68% of subjects of the individuals stored in Freebase lacked information regarding the profession, and 71% of them lacked place of birth. Incompleteness and ambiguity hinder the use of such KBs in real-world applications; hence the scientific community has been working to develop knowledge completion approaches to solve the problem.

Below we discuss some of the knowledge completion tasks. In some papers, the term *link-prediction* (NICKEL et al., 2015) is used to define any knowledge completion task. However, in this work, we are adopting a more specific definition (SOCHER et al., 2013; BORDES et al., 2013).

2.2.1 Triple Classification

When performing *triple classification*, one is concerned in finding out the answer TRUE or FALSE for a given previously unknown fact (triple) (SOCHER et al., 2013). This task is more similar to a conventional machine learning classification task, and therefore we commonly use the same metrics used for classifiers.

2.2.2 Link Prediction

This task is concerned with finding the answer to an open question. Given a known head entity plus a relation, what is (or what are) the matching tail entities that hold. We can post this question for any of the triple elements. So, we define *tail prediction* as $\langle head, relation, ? \rangle$, *head prediction* as $\langle ?, relation, tail \rangle$, and *relation prediction* as $\langle head, ?, tail \rangle$.

2.2.3 Entity Resolution

This is the task of finding out whether two entities are the same. For instance, in a given KB, one can find the former US president Barak Obama represented as *B. Obama* or simply *Obama*.

Later, in section 2.4.2.3, we will discuss in more detail how we can use Knowledge Base Embeddings to perform such tasks.

2.3 Knowledge Base/Graph Formal Notation

From *Statistical Relational Learning (SRL)* (GETOOR; TASKAR, 2007; NICKEL et al., 2015) we adopt some formal definitions and notations that will be necessary in the remainder of the work. We denote a knowledge graph by $\mathcal{G}$. We denote the set of all possible entities by $\mathcal{E} = \{e_1, ..., e_N\}$, and the set of all possible relations $\mathcal{R} = \{r_1, ..., r_M\}$. A possible triple is denoted by $x_{h,r,t} = \langle e_h, r, e_t \rangle$, where e represents entities and r represents a relation between them, while h and t represent head and tail, respectively. Each triple can be modeled as a binary random variable

$$y_{h,r,t} = \begin{cases} 1, & \text{if the triple is in the KG of interest;} \\ 0, & \text{otherwise.} \end{cases} \tag{2.1}$$

In this work, we adopt the *open world assumption*, where a missing triple may be either false, or simply unknown. The set of all possible triples within $\mathcal{G}$ is represented by $\mathcal{T} = \mathcal{E} \times \mathcal{R} \times \mathcal{E}$. We can calculate the amount of possible triples as $N^2 \times M$, where N and M represent the number of entities and relations, respectively. So,

the total amount of possible triples can be enormous depending on the number of entities and relations. Even though KG's are generally considered sparse graphs (NICKEL et al., 2015), they may contain millions or billions of stored triples, requiring clever machine learning methods to deal with them.

2.4 Knowledge Base Completion Approaches

To perform the downstream KG tasks mentioned in Section 2.2, a broad range of approaches have emerged from SRL. Here we first discuss some approaches from the so-called *graph feature* family of models, followed by the family of *latent feature* models.

2.4.1 Graph Feature Models

This family of knowledge base completion approaches relies on the assumption that the existence of an edge of the graph can be predicted based on extracted features from the observed graph (NICKEL et al., 2015). That is, it is possible to predict new triples using features extracted from other observed triples in the graph. Even though there are several methods in this family, we only discuss three of them. In section 2.4.1.1, we define the *Path Ranking Algorithm*, a seminal method. Then, we define the *Subgraph Feature Extraction* method, followed by the description of *Context-Aware Path Ranking* in sections 2.4.1.2 and 2.4.1.3 respectively, which are essential building blocks of the XKE algorithms as well as for the advancements we propose in our work.

2.4.1.1 Path Ranking Algorithm (PRA)

This approach was proposed by Lao e Cohen (2010) and it is based on the extraction of paths from the graph using random walks of bounded length, to build a feature matrix to predict new triples of a given relation of interest. More formally, let us consider a KG $\mathcal{G}$, formed by the set of entities $\mathcal{E} = \{e_1, ..., e_{Ne}\}$ and the set of relations $\mathcal{R} = \{r_1, ..., r_{Nr}\}$. For a given relation on interest, $r_k \in \mathcal{R}$, we build a subset $\mathcal{D}_k \in \mathcal{G}$ taking into account only triples from that relation, represented by $\langle e_h, r_k, e_t \rangle$ with $h, t \in \mathcal{E}$. Let us now consider a triple $\langle e_i, r_k, e_j \rangle \in \mathcal{D}_k$, we define

a path type $\pi_L(i,j,k)$ as the sequence of edges $s_1 \rightarrow s_2 \rightarrow ... \rightarrow s_L$ of a path $e_i \xrightarrow{s_1} e_2 \xrightarrow{s_2} ... \xrightarrow{s_L} e_j$, with maximum length L, connecting the head e_i and the tail e_j. The edges may be any relation $r \in \mathcal{R}$ and even its inverse direction, in this case represented by r_i^{-1}. For each triple, we define $\Pi_L(i,j,k)$ as the set of all different encountered paths types connecting e_i and e_j. It is important to mention that, in order to build the path types between two nodes, two random walks are performed, one departing from the head and the other departing from the tail, and the paths from each side are merged in any intersecting intermediate nodes. After finding the set of path types, the probability $P(\pi_L(i,j,k))$ is computed recursively and associated with each path type in the feature vector. The feature vector for a given triple is represented by

$$\Phi_{ijk}^{PRA} = [P(\pi) : \pi \in \Pi_L(i,j,k)]. \tag{2.2}$$

To train the logistic regression, PRA will construct a feature matrix where the training examples are a set of triples of the relation of interest, r_k, denoted by $\mathcal{D}_k^+$, combined with a set of negative (corrupted) examples $\mathcal{D}_k^-$. Finally, to find the probability that a certain triple holds, one can apply the logistic regressor function

$$f_{ijk}^{PRA} := w_k^T \Phi_{ijk}^{PRA}; \tag{2.3}$$

where w_k^T are the weights assigned by the logistic regressor for each feature during the training process.

2.4.1.2 Subgraph Feature Extraction (SFE)

This is a variant of the *Path Ranking Algorithm (PRA)* (LAO; MITCHELL; COHEN, 2011), proposed by Gardner e Mitchell (2015), with better performance metrics and running time. We will discuss all the relevant details because it is a very important sub-module of XKE. Similarly to PRA, for each relation, a feature matrix is extracted from observed edges in a given KG, and then fed to a classifier (generally a logistic regressor) so as to predict new triples. The fundamental difference from the previous approach is that, while PRA calculates the associated probability of existence for each path type to build the feature matrix, SFE skips this rather computationally expensive step and only fills each feature vector with binary values indicating the existence (or not) of a given path type.

Formally, let us consider a KG $\mathcal{G}$, and its set of entities $\mathcal{E}$ and relations $\mathcal{R}$, like we did in the previous section. For a target rekatuib, denoted by r_k, we build a subset of existing triples with that relation, $\mathcal{D}_k \in \mathcal{G}$. Similarly to PRA, let $\pi_L(i,j,k)$ represent a path type of maximum length L connecting entity e_i to e_j, and $\Pi_L(i,j,k)$ denote the set of all paths encountered between those entities. Now, let us define $p(\pi)$ as the binary variable that represents the existence (or not) of a given path connecting entities e_i and e_j. Similarly to Equation 2.2, the feature vector extracted for a given triple is represented by

$$\Phi_{ijk}^{SFE} = [p(\pi) : \pi \in \Pi_L(i,j,k)]. \tag{2.4}$$

To feed the logistic regressor, SFE will construct a feature matrix where the training examples are the triples of the relation of interest, r_k, denoted by $\mathcal{D}_k$ and a set of negative (corrupted) examples $\mathcal{D}_k^-$, and the features will be extracted using Equation 2.4. After training the logistic regressor, a parameter vector w_k is obtained for the relation r_k. If one wants to find out the existence of an arbitrary triple $\langle e_a, r_k, e_b \rangle \notin \mathcal{D}_k \cup \mathcal{D}_k^-$, the computation can be made, after extracting the SFE features Φ_{abk}^{SFE}, using

$$f_{abk}^{SFE} := w_k^T \Phi_{abk}^{SFE}. \tag{2.5}$$

Now, let us discuss in detail the process of extraction of paths $\pi_L(h,t,k)$ for a given triple $\langle e_h, r_k, e_t \rangle$. Take a path type π formed by an arbitrary sequence of edges in a KG, $\mathcal{G}$. As a first step, SFE (and PRA) constructs subgraphs originated from each entity e_h and e_t. As a result, two subgraphs $\mathcal{G}_h$ and $\mathcal{G}_t$ will contain paths $\pi_{h,i}$ and $\pi_{t,i}$ originated from each entity and arriving at some intermediate node i. Then, for each intermediate node i that is common to both subgraphs, a path type $\pi_{h,i} \cup \pi_{i,t}$ is stored in the feature matrix. To further clarify this process with an example, let us consider that we are running SFE to construct features with maximum length $L = 4$. The algorithm will construct subgraphs originating from e_h and e_t up to maximum length $L = 2$. Now, let us say that subgraph construction originated from the head e_h found the following path: $\pi_{h,i} = e_h \xrightarrow{r_1} x \xrightarrow{r_2^{-1}} i$, where we use the letter x to denote the node found at the first step, because this information will be disregarded in the feature construction. Additionally, r_1 and r_2^{-1} are the edge types found up to node i, and the -1 means that for the second

edge type, the directed graph was pointing from i to x, in other words, the edge was traversed in reverse direction. Now, in the opposite side, SFE expanded a subgraph departing from e_t with the form $\pi_{t,i} = e_t \xrightarrow{r_7} y \xrightarrow{r_5} i$. After building the sets of subgraphs originated from each entity, SFE checks whether there are intermediate nodes i in common for both subgraphs. In the case of this example, the combined path would be $\pi_{h,t} = e_h \xrightarrow{r_1} x \xrightarrow{r_2^{-1}} i \xrightarrow{r_5^{-1}} y \xrightarrow{r_7^{-1}} e_t$, and, disregarding the nodes, the path type that would be added to the feature vector of this triple would be $\xrightarrow{r_1} \cdot \xrightarrow{r_2^{-1}} \cdot \xrightarrow{r_5^{-1}} \cdot \xrightarrow{r_7^{-1}}$.

Lao, Mitchell e Cohen (2011) originally proposed subgraphs construction using random walks in order to make the problem computationally tractable. Gardner e Mitchell (2015) also adopted random walks for the SFE, but proposed, as an alternative method, to construct the subgraphs using a *Breadth First Search (BFS)* as a way to improve the number of extracted features. While this approach allows us to find all possible path types interconnecting two entities in the graph, it can quickly become computationally intractable, even for moderately sized graphs. In order to keep the BFS tractable while running SFE in large KGs, Gardner e Mitchell (2015) proposed to skip the expansion of graph nodes with a high degree. So, if a path departing from e_h reaches a node with a degree higher than a given number (a parameter of the model), that node will not be expanded further, even though it still will be considered as an intermediate node i that later can be merged to the subgraph originated from e_t. This approach significantly improved the number of extracted features compared to PRA or SFE performed with random walks.

2.4.1.3 Context-aware Path Ranking (C-PR)

Methods inspired on PRA and its predecessors, despite their good predictive performance (TOUTANOVA; CHEN, 2015), suffer from *scalability* and *feature explosion*. These two characteristics translate into high RAM requirements during the training process as well as elevated training time. To counteract this problem, Mazumder e Liu (2017) developed the *Context-aware Path Ranking (C-PR)* method, which relies on contextual similarities among entities (nodes of the graph) to serve as a heuristics during the feature extraction process. Indeed, this approach fosters

scalability of the method and enables the extraction of more interpretable features since it only allows the inclusion of nodes that are context-related to both head and tail during the random walk. Another advantage is the reduction of the feature vector size, helping to prevent feature explosion.

Formally, let us consider the same setting as provided in Sections 2.4.1.1 and 2.4.1.2, defining a KB, $\mathcal{G}$, formed by the set of entities $\mathcal{E} = \{e_1, ..., e_{Ne}\}$ and relations $\mathcal{R} = \{r_1, ..., r_{Nr}\}$. For a given triple $\langle e_i, r_k, e_j \rangle$, we define path type $\pi_L(i, j, k)$ as a sequence of edges with maximun lengh L connecting entities e_i, $e_j \in \mathcal{E}$. Let us again define $\Pi_L(i, j, k)$ as the set of all path types connecting e_i and e_j, and $p(\pi)$ the random variable representing the existence of a path between the given entities. Similarly to PRA and SFE, C-PR (MAZUMDER; LIU, 2017) constructs a set of features to model existence of triples defined by

$$\Phi_{ijk}^{C-PR} = [p(\pi) : \pi \in \Pi_L(i, j, k)]. \tag{2.6}$$

Inspired by PRA, C-PR builds the set of features $\Pi_L(i, j, k)$ using bi-directional random walks to expand subgraphs $\mathcal{G}_h$ and $\mathcal{G}_t$ originated from head and tail entities, respectively. The major difference is that it relies upon heuristics based on contextual similarities to decide which nodes will be included in the search.

The feature extraction is a two-step process: first, we calculate the *contextual similarity* between head and tail, then perform the *context-aware path search*:

Contextual Similarity: C-PR uses a neural word embedding model, such as **word2vec** (MIKOLOV et al., 2013a; MIKOLOV et al., 2013b), to capture the semantics of entities. Formally, let us consider a triple $\langle h, r, t \rangle$. Now, let us denote by W_h and W_t the vectors representing each entity. For each triple, C-PR calculates the context similarity using *cosine similarity* expressed by $sim(h, t) = \frac{W_h \cdot W_t}{||W_h|| \cdot ||W_t||}$.

Contex-aware Path Search: C-PR uses a bi-directional random walk to construct paths. Let $F_{seq} = \langle h, v_f^1, v_f^2, ..., v_f^k \rangle$ denote a random walk originated from head entity, h, and, similarly, $B_{seq} = \langle t, v_b^1, v_b^2, ..., v_b^k \rangle$ denotes the random walk originated from the tail entity, t, where intermediate nodes are represented by v_f and v_b for the forward and backward random walks, respectively. We detail the construction of the F_{seq} path:

- Step 1: Let $N_G(v)$ denote the set of neghboring nodes of a node v. C-PR constructs the set of contextual nodes, denoted by $N_{context}(v)$, based on $N_G(v)$ as follows:

$$N_{context}(v) = \{v' | v' \in N_G(v), Relv(v', h, t) \geq sim(h, t), v' \notin F_{seq}\}, \quad (2.7)$$

where

$$Relv(v', h, t) = \theta \cdot sim(v', h) + (1 - \theta) \cdot sim(v', t) \quad (2.8)$$

measures how much the label of the node v is contextually related to both head and tail labels, given a corpus $\mathcal{C}$. θ is a parameter of the model that defines the importance of head and tail for the contextual similarity of the current node.

After $N_{context}(v_f^K)$ is computed, the method selects a random node $v \in N_{context}(v_f^K)$ and performs step 2. If $N_{context}(v_f^K) = \emptyset$ the expansion of the path is halted.

- Step 2: The algorithm tries to find a complete path by checking whether $v = v_b^l \in B_{seq}, l \leq k$. If a common node between F_{seq} and B_{seq} is found, the algorithm infers that a complete path was found and returns the path sequence. More formally, the path that will be appended to F_{seq} is $B_{seq}^{-1} = \langle v_b^{l-1}, ...v_b^1, t \rangle$, and the final path is $P_{seq} = \langle h, v_f^1, ..., v, v_b^{l-1}, ...v_b^1, t \rangle$. Finally, the algorithm infers the relations (edges) of P_{seq} to append to the feature vector. If no complete path is found during this step, step 3 is performed.

- Step 3: If the algorithm is not able to find any path during step 2, it performs an additional step to prevent the search going out of context, by computing $\Delta(v, v_f^k) = Relv(v, h, t) - Relv(v_f^k, h, t)$. If $\Delta(v, v_f^k)$ is positive, the node is considered "in-context" by the algorithm and is appended to F_{seq} as v_f^{k+1}. Otherwise, the node is considered "out-of-context" with regards to head and tail and is disregarded from F_{seq}.

2.4.2 Latent Feature Models (Embeddings)

Latent feature models rely on features that are not observed (that is, they are latent) in the KG (NGUYEN, 2017; NICKEL et al., 2015; WANG et al., 2017). These models work by converting semantically rich information into low-dimensional

vector space representations, assuming we can derive relationships between entities from their latent features.

In this section, we present the taxonomy of different approaches of embedding models, followed by the discussion about the training process of generic embedding models, and then tackle how we can use embeddings to perform knowledge base downstream tasks. Finally, we discuss in a more detailed fashion **TransE** and **Analogy**, the embedding models we used in this research.

2.4.2.1 Overview of Embedding Families

Since the introduction of the considered seminal works in the field, namely **RESCAL** (NICKEL; TRESP; KRIEGEL, 2011) and **TransE** (BORDES et al., 2013), a vast array of approaches emerged. Since then, several works have grouped different embeddings into different families. In this work, we follow the taxonomy proposed by Rossi et al. (2021), dividing models into three main categories.

- **Tensor Decomposition Models:** the knowledge base embedding models of this family represent the KG as a three-dimensional tensor of $N \times M \times N$, where N is the number of entities and M represents the number of different relations in the KG. Another way of interpreting this representation is that the tensor comprises M adjacency stacked matrices, one for each relation. By performing tensor decomposition, one can obtain the latent representations of entities and relations as vectors or matrices. Generally, what differentiates each embedding model is the set of constraints applied to obtain the learned embeddings. In some works, such as Wang et al. (2017), this family is also named as *Semantic Matching Models*, because the scores produced by this type of embeddings can also be interpreted as similarity measurements of the underlying semantics of the entities and relations of the KG. As examples of embeddings from this family we list **RESCAL** (NICKEL; TRESP; KRIEGEL, 2011), **DistMult** (YANG et al., 2015), **HolE** (NICKEL; ROSASCO; POGGIO, 2015), **ComplexE** (TROUILLON et al., 2016), **SimplE** (KAZEMI; POOLE, 2018), and finally, **Analogy** (LIU; WU; YANG, 2017).

- **Geometric Models:** this family of models rely on the idea that relations are geometric transformations within the low-dimensional vector space. The plausibility of a triple is given by a scoring function obtained by some distance measurement between entities within the vector space, after some translation operation. Models from this family can be purely *translational*, such as **TransE** (BORDES et al., 2013), and its vast array of extensions: **TransH** (WANG et al., 2014), **TransR** (LIN et al., 2015), **TransD** (JI et al., 2015), **ManifoldE** (XIAO; HUANG; ZHU, 2016a), just to name a few. Within this family, there are some models that rely on translational embeddings augumented with additional embeddings, such as **STransE** (NGUYEN et al., 2016b) and **CrossE** (ZHANG et al., 2019), being the latter augumented with interaction embeddings. Finally, some models rely on operations that are not purely translational, like rotations. **TorusE** (EBISU; ICHISE, 2017) and **RotatE** (SUN et al., 2019) are examples of this approach.
- **Deep Learning Models:** this models are based on deep neural network architectures. Embeddings are learned in conjunction with the weights of the neural networks, resulting in good expressivenes of the models at the expense of harder training process and potential overfitting. This family of models can be further divided into three different subgroups based on the type of neural network architecture they employ. The first subgroup is the *convolutional neural networks*, that apply one or more *convulutional* layers. **ConvE** (DETTMERS et al., 2017), **ConvKB** (NGUYEN et al., 2017) and **ConvR** (JIANG; WANG; WANG, 2019), **ConEx** (DEMIR; NGOMO, 2020) are examples of this subgroup. The second subgroup is comprised of models based on *capsule networks* such as **CapsE** (NGUYEN et al., 2018), **HCapsE** (LI; HOU; ZHOU, 2021), and finally, the third subgroup is based on *recurrent neural networks*, being **RSN** (GUO; SUN; HU, 2019) one member of this subfamily.

The list of embedding models mentioned above is not exhaustive but helps us acknowledge how much traction this line of research has gained over the last decade. In the remainder of this section, we first discuss a generic notation and model training process, followed by a brief discussion of how knowledge base downstream

tasks are performed with embedding models. After that, we detail two of the embeddings mentioned above, **TransE** and **Analogy**, which are the models we chose to evaluate in the present work. The first one is because it is one of the most well-known embedding models and the latter for its analogical properties. As we discuss extensively in this work, XKE is a model-agnostic approach and, therefore, could be used to produce explanations for any of the embeddings mentioned above. As we will state in our conclusion, evaluating XKE performance with a more extensive list of embedding models is one of the potential research tracks to follow after the present work.

2.4.2.2 Formal Definition and Model Training

Usually, entities are represented by vectors of an arbitrary dimension, which can be regarded as a model hyper-parameter. Relations typically are represented by operations within that vector space. The next step consists of defining a plausibility score function $f(x_{h,r,t} \mid \Theta)$ that will model the existence of a triple given the set of learned parameters Θ, which includes representations of entities and relations itself, plus any other additional parameters depending on the embedding model. The third step is to define an optimization function that will be used to maximize the plausibility of all observed facts during the training phase. Two optimization functions commonly adopted are the *logistic loss function* and the *pairwise ranking loss* that we will formalize below.

Following the same notations adopted above, let $\mathcal{G}$ denote a knowledge graph composed of a set of positive triples $x_{h,r,t} = \langle e_h, r_r, e_t \rangle$, where $e_h, e_t \in \mathcal{E}$ and $r_r \in \mathcal{R}$. As before, $\mathcal{E}$ and $\mathcal{R}$ represent the set of entities and relations of $\mathcal{G}$. Each triple will be associated with $y_{h,r,t} = f(x_{h,r,t} \mid \Theta)$, where Θ is the set of parameters of the model that will be learned during training, and $y_{h,r,t}$ is the random variable representing the existence (or not) of a triple:

$$y_{h,r,t} = \begin{cases} 1, & \text{if the triple exists in } \mathcal{G}, \\ -1, & \text{otherwise.} \end{cases}$$

As we are following the *open world assumption*, the set of positive triples $\mathcal{D}^+$ that will be used during model training will be precisely the triples in $\mathcal{G}$, and

a set of negative triples $\mathcal{D}^-$ will be generated "on the fly" while carrying out the training — by randomly replacing either the head or the tail of a triple, following some heuristics that we will omit for the sake of simplicity.

If one is using the *logistic loss function*, then Θ will be learned to find

$$\min_{\Theta} \sum_{\tau \in \mathcal{D}^+ \cup \mathcal{D}^-} \log\left(1 + \exp\left(-y_{h,r,t} \cdot f(h,r,t)\right)\right), \tag{2.9}$$

where $\tau = (h, r, t)$ represents a training example or triple of the form $\langle e_h, r_r, e_t \rangle$.

On the other hand, the training process using *pairwise ranking loss function* will find

$$\min_{\Theta} \sum_{\tau^+ \in \mathcal{D}^+} \sum_{\tau^- \in \mathcal{D}^-} \max\left(0, \gamma - f(h,r,t) + f(h',r',t')\right), \tag{2.10}$$

where $\tau^+ = (h, r, t)$ is a positive training example, $\tau^- = (h', r', t')$ a negative one, and γ is separating margin between them. Optimization usually is performed using *stochastic gradient descent (SGD)* in mini-bach mode, employing either fixed or adaptative learning rates combined with additional constraints imposed by each different type of embedding model. Also, one can use different strategies for the generation of negative training examples through the corruption of positive triples. For the specific case of the *logistic loss function*, it is also possible to tune the proportion of negative training examples over positive ones, in order to foster model performance.

2.4.2.3 Knowledge Base Completion with Embeddings

Let us now detail how we perform completion tasks with embeddings, regardless of the type of embedding applied to the problem. Performance of embedding models is usually evaluated with the use of a train-test split of the dataset.

• **Triple Classification:** as we saw in the previous section, model training is performed with the set of all positive triples from $\mathcal{G}$, together with corrupted triples generated during the training process, trying to minimize the overall loss function. During this process, only the representation of entities and relations are learned globally. So, a further step is required in order to allow the embedding model to perform this task: for each relation $r \in \mathcal{R}$, a threshold δ_r must be obtained

via maximization of the model accuracy when tested against some test dataset. The δ_r will be selected as the value that minimizes the classification error. As we are dealing with a classification problem, all commonly used metrics for classification tasks can be applied.

More formally, let us denote by $\hat{y}_{h,r,t}$ a prediction performed by an embedding model. For a triple $x_{h,r,t}$, we represent the result of the *triple classification* task as

$$\hat{y}_{h,r,t} = \begin{cases} 1, & \text{if } f(x_{h,r,t} \mid \Theta) \leqslant \delta_r \ , \\ -1, & \text{otherwise.} \end{cases}$$

• **Link Prediction:** this is essentially a ranking task. For a given triple $\langle e_h, r_r, e_t \rangle$, first, the scoring function is calculated for $\langle e_h, r_r, x \rangle \ \forall x \in \mathcal{E}$, then results are sorted in descending order according to the plausibility of each possible answer. The ranking position of the correct tail is stored, and the process is repeated for $\langle x, r_r, e_t \rangle$. After processing link prediction for all triples of the test dataset, a mean rank is computed.

2.4.2.4 TransE

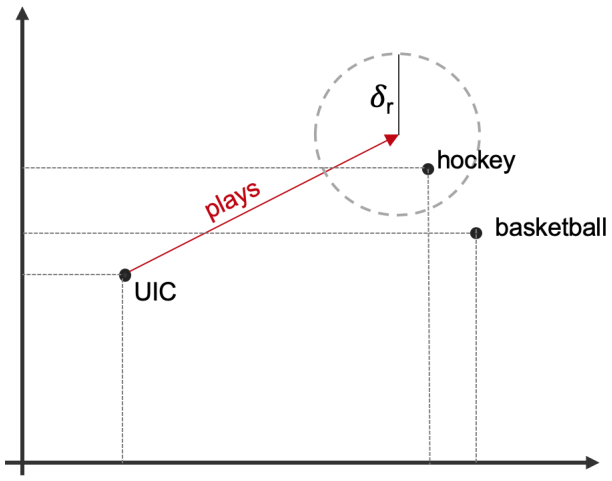

Figure 3 – Toy example for the representation of three entities and a relation using TransE within a 2-dimensional vector space.

Proposed by (BORDES et al., 2013), it was largely inspired in word2vec (MIKOLOV et al., 2013b), from natural language processing field. In this embedding model, entities and relations are represented by vectors within a d-dimensional vector space. Formally, let us denote by h, r and t the vector representations of entities e_h, r_r and e_t, respectively. The scoring function adopted is a simple vector operation represented by

$$f^{TransE}(x_{h,r,t}) = - \parallel h + r - t \parallel_{1;2}, \tag{2.11}$$

that can be computed using either norm ℓ_1 or ℓ_2. Figure 3 displays a toy example of two entities and a relation represented in a two-dimensional space. In this example, we can see that the result of $h + r$ falls at an arbitrary point within the vector space. All entities falling inside the circle represented in the picture, with radius δ_r, will be deemed positive by the embedding model. Hence, the model would classify the triple $\langle UIC, plays, hockey \rangle$ as true, whereas $\langle UIC, plays, basketball \rangle$ would be classified as false.

2.4.2.5 Analogy

Proposed by Liu, Wu e Yang (2017), ANALOGY is an embedding model that captures *analogical* properties among entities and relations. In this model, entities are represented as vectors and relations as matrices. In other words, relations are linear mappings that map the vector of the head entity to the tail entity.

More formally, let us represent by $v_n \in \mathbb{R}^k$ the latent feature vector of an entity e_n. Let $W_n \in \mathbb{R}^{k \times k}$ represent the embedding matrix for relation r_n. The ANALOGY scoring function is represented by

$$f^{ANALOGY}(x_{h,r,t}) = \langle v_h^T W_r, v_t \rangle = v_h^T W_r v_t. \tag{2.12}$$

To model analogical properties, the following constraints are imposed during model training

$$W_r W_r^T = W_r^T W_r, \forall r \in \mathcal{R}, \tag{2.13}$$

$$W_r W_{r'} = W_{r'} W_r, \forall r, r' \in \mathcal{R}; \tag{2.14}$$

ensuring the linear transformations performed by the relation matrices to have both properties of *normality* and *commutativity*. The training process can be carried out using the *log-loss function* described in Section 2.4.2.2.

2.5 Interpretability

Interpretability in *artificial intelligence (AI)* and *machine learning (ML)* has a long history (BIRAN; COTTON, 2017). However, in recent years, the interest in the topic is gaining traction, notably motivated by the evolution of machine learning methods that are becoming more and more complex to attain higher accuracy, such as *deep neural networks* and *random forests*.

According to Miller (2017), this resurgence is motivated by the fact that modern AI systems may end up with limited practical application due to ethical reasons, and notably due to the lack of trust from users. This lack of trust may be solved by the adoption of more interpretable and transparent systems, allowing their users to understand them better. He also reports that there is a gap between definitions of interpretability adopted in AI and other scientific fields, such as philosophy, cognitive, and social sciences (MILLER, 2017; MITTELSTADT; RUSSELL; WACHTER, 2019), even though philosophy, for instance, has been investigating this matter for centuries.

According to Lipton (2016), the definition of interpretability is not a clear consensus among AI researchers. One of the definitions is that *interpretability* means **the degree of human comprehensibility of a given black-box model or decision** (MITTELSTADT; RUSSELL; WACHTER, 2019). On the other hand, an *explanation* refers to **ways of exchanging information about a phenomenon**, be it the functionality of a model or a rationale for a decision, to different stakeholders (MITTELSTADT; RUSSELL; WACHTER, 2019). Here we begin to differentiate the meanings of interpretability and explanation. Another essential concept arises: that the agents that are receiving the explanation, sometimes called the *explainees*, may have different knowledge backgrounds, and therefore, require different types of explanations.

Miller (2017) simply defines an explanation as the answer for a why-question.

In his survey, he lists some findings from the literature about interpretability in fields other than AI that he considers important for the AI community:

• **Explanations are contrastive:** People do not ask why a certain event P happened. They ask why a certain event P happened, instead of another event Q. Counterfactual cases, like Q, are often denominated *foils*.

• **Explanations are selected:** Humans tend to select one or two causes from all possible causes to be used as an explanation to a phenomenon. This selection is influenced by cognitive bias.

• **Probabilities do not matter:** The most likely explanation is not necessarily the best explanation for a given phenomenon.

• **Explanations are social:** They happen within a social interaction, and a transfer of knowledge occurs among the subjects.

The major takeaway from these four points is that explanations are *contextual*. When trying to explain an event that has many causes, the explainer may select just a subset of the explanations, while the explainee may care about only a subset of this subset, and probably, they are going to discuss and interact about that sub-subset of explanations.

Finally, we present the definition of what is it to explain a machine learning prediction as proposed by Ribeiro, Singh e Guestrin (2016): **to present visual and textual artifacts that can provide a qualitative understanding of the relationship between the instance's components and the model's prediction**. In their work, while proposing a system that provides explanations for the predictions of any complex classifier, called LIME, they advocate that explaining a prediction is a crucial step to ensure trust from users.

Now that we have defined interpretability and explanation, let us discuss how machine learning systems and explainer systems can be classified.

2.5.1 Degrees of Interpretability

From the point of view of the model that is being evaluated, Doran, Schulz e Besold (2017) categorized AI systems regarding their degree of interpretability:

• **Interpretable system:** it is a system where the user can contemplate, study, and understand how inputs are mathematically mapped to outputs.

• **Opaque system (black-box):** it is a system that does not allow the user to access the underlying mathematics to try to understand how inputs are mapped to outputs. Systems that the user can access the underlying mathematics but, due to the complexity, draw no insight, also fall in this category.

• **Comprehensive system:** it is a system that emits symbols together with the outputs that allow the user to relate the input with the output.

2.5.2 Explainer Models

Machine learning models such as *linear regression*, *logistic regression*, for instance, are considered in most cases to be intrinsically interpretable. When we need to draw explanations from models that are considered opaque, one common approach is to plug an *explainer model*. Ribeiro, Singh e Guestrin (2016) define an *explainer model* as **any technique able to produce explanations about the predictions of any classifier in an interpretable and faithful manner**. They claim that such explainer model should be *model-agnostic*, meaning it should be able to produce explanations regardless of the black-box model. On the other hand, tailored methods to produce explanations only for a specific model are called *model-specific* (in Chapter 3 we will discuss several model-specific approaches).

Ribeiro, Singh e Guestrin (2016) also subdivide model-agnostic explainer models into two different groups:

• **Global-Surrogate:** This type of explainer model tries to mimic black-box predictions globally. The main shortcoming of this kind of approach is that, in most cases, they cannot faithfully mimic the original classifier while maintaining interpretability characteristics. A machine learning practitioner only would use a complex and opaque classifier aiming to get the best possible model performance; if a less complex (and more interpretable) classifier would be able to provide such performance, one whould avoid using the complex classifier in the first place.

• **Local-Surrogate:** This type of explainer model works by building local approximations of the black-box classifier. Instead of trying to map the inputs into

outputs for the entire dataset, it performs such mapping only on the neighborhood of the prediction being explained. Ribeiro, Singh e Guestrin (2016) claim that explainer models should be locally faithful.

Andrews, Diederich e Tickle (1995) wrote a survey of rule-extraction algorithms that were used to explain predictions of *artificial neural networks (ANN)*. In that survey, they classified different approaches within five different dimensions. For simplicity, we will only discuss one of those five dimensions, *translucency*, that divides rule-extraction models into three different categories:

• **Decompositional Approaches:** The ones that focus on extracting rules from the hidden layers of the ANN.

• **Pedagogical Approaches:** The ones that consider the ANN a black-box, and mine rules only by considering inputs and outputs (ANN's predictions).

• **Ecletic Approaches:** The one that combine the decompositional and pedagogical approaches.

Even though the latter classification was originally designed for rule-extraction algorithms, we can apply the same notation to classify explainer models. A pretty straightforward conclusion is that *pedagogical* explainer models tend to be also *model-agnostic* because they are trained disregarding model intrinsic characteristics, only taking into account inputs and predicted outputs.

3 Related Work

This chapter begins with a more detailed explanation of the XKE algorithm (GUSMÃO et al., 2018), the method we are proposing improvements in this thesis. We then provide a broader discussion about several approaches that tackle the problem of interpretability of latent feature models.

3.1 Explaining Answers from Knowledge Base Embeddings (XKE)

In Sections 3.1.1 and 3.1.2 will discuss in detail the algorithm proposed by Gusmão et al. (2018) to improve the interpretability of *knowledge base embeddings*. The idea of this *pedagogical* approach is to produce explanations about the predictions from a knowledge base embedding model by using a more intrinsically interpretable classifier (logistic regression) as an explainer model. Later, in Chapter 4, we will present two novel enhancements designed to improve XKE's fidelity (i.e., the ability of the explainable classifier to mimic the outputs of the black-box classifier correctly).

The idea of this *pedagogical* approach is to produce explanations about the predictions from an knowledge base embedding model by using a more intrinsically interpretable classifier (logistic regression) as an explainer model.

3.1.1 XKE-PRED

It is the most straightforward pedagogical method. Assuming the embedding model as a black-box classifier, we construct a training set composed of triples that are fed into the embedding model. The outputs produced by the embedding model are the predictions that we want to explain with the interpretable classifier. Because neither the input triples nor the operations performed by the embedding model within the low-dimensional vector space are interpretable in principle, we resort to

SFE to build a feature matrix based on the predicted graph, where labels are the embedding's predictions. The set of features extracted by the SFE algorithm can be regarded as Horn clauses.

More formally, let us consider a knowledge graph $\mathcal{G}$ and the set $\mathcal{T} = \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ representing the set of all possible triples of this KG. From now on, we represent each triple $\langle e_h, r_r, e_t \rangle$ as $x_{h,r,t}$. We denote by $g : \mathcal{T} \to \{0, 1\}$ the function of the embedding black-box classifier. We define $\Pi_{\mathcal{G}}$ as the set of all possible paths connecting two entities, and $P(\Pi_{\mathcal{G}})$ its power set. The feature extraction function performed by SFE for an arbitrary triple $x_{h,r,t} \in \mathcal{T}$, within the graph $\mathcal{G}$, is represented by $SFE : \mathcal{T} \to P(\Pi_{\mathcal{G}})$. The result of applying the SFE function to a triple $x_{h,r,t}$ is denoted by $\Pi_{h,r,t|\mathcal{G}} \in P(\Pi_{\mathcal{G}})$. We call *Explaining Knowledge Embeddings with Predicted Features (XKE-PRED)* the following scheme. First, construct a graph with all triples deemed positive by the black-box classifier g:

$$\hat{\mathcal{G}} = \{x_{h,r,t} \in \mathcal{T} \mid g(x_{h,r,t}) = 1\}. \tag{3.1}$$

Then, build an arbitrary subset of training examples $\mathcal{D} \subseteq \mathcal{T}$. The training set for the interpretable classifier will be constructed as follows:

$$\mathbb{D}' = \{(SFE(x_{h,r,t} \mid \hat{\mathcal{G}}), g(x_{h,r,t})) \mid x_{h,r,t} \in \mathcal{D}\}. \tag{3.2}$$

Finally, to draw explanations, train an interpretable classifier $g' : P(\Pi_{\hat{\mathcal{G}}}) \to \{0, 1\}$ using $\mathbb{D}'$.

It is important to note that building $\hat{\mathcal{G}}$ by checking every possible fact in $\mathcal{T}$ quickly becomes intractable as the size of $\mathcal{T}$ grows quadratically with the size of $\mathcal{E}$. Hence, Gusmão et al. (2018) resorted to an entity similarity method. Instead of checking every possible triple, they found, for each triple $x_{h,r,t} \in \mathcal{G}$, a set of k-nearest neighbors for both *head* and *tail* in the embedded vector space, and checked the plausibility of the existence of new triples for all possible combinations of those entities for the given relation.

One of the key characteristics of *XKE-PRED* is that, in order to explain a prediction from an embedding model, the feature matrix is obtained by SFE from the predicted graph $\hat{\mathcal{G}}$, in other words, explanations are based as well in predictions from the same model. This means that, when looking at the explanations produced

by the interpretable model, one is analyzing the internal coherence of the model rather than evaluating how features from the real world influence the predictions made by that embedding.

3.1.2 XKE-TRUE

To overcome the fact that explanations of XKE-PRED are also based on predictions obtained from the model that one is trying to interpret, Gusmão et al. (2018) proposed a variant called *explaining knowledge base embeddings with observed features (XKE-TRUE)*. In this approach, the feature matrix is obtained from the original graph $\mathcal{G}$, referred to as *ground truth*.

Following the same notation from XKE-PRED, the only difference is that the set of training instances constructed to train the interpretable classifier, $g'' : P(\Pi_{\mathcal{G}}) \rightarrow \{0, 1\}$, is built applying SFE to $\mathcal{G}$, not to $\hat{\mathcal{G}}$:

$$\mathbb{D} = \{(SFE(x_{h,r,t} \mid \mathcal{G}), g(x_{h,r,t})) \mid x_{h,r,t} \in \mathcal{D}\}. \tag{3.3}$$

3.2 Other Related Work

The interest in interpretability and explainability within the field of AI dates back to the seventies and comes from several different perspectives (BIRAN; COTTON, 2017). Attempts to produce explanations for outputs of *Expert Systems* were reported in 1975, more specifically for rule-based systems. In the two decades that followed, the interest in the field grew, and several approaches arose, aiming to provide explanations for Bayesian networks and recommender systems. Gusmão et al. (2018) already wrote a detailed review of the literature, on which we are also basing the present work, and most of the essential concepts have been discussed in Chapter 2. Here we will focus more on recent work related to knowledge base completion.

Bianchi et al. (2020) discussed some issues or explainability of knowledge graph embeddings in recent work. They state that explainability should be closely related to methods that support logical reasoning, as logical axioms support logical inferences. Another issue they address is the fact that, so far, there is no

consensus about how to measure how explainable a system is. Finally, they come up with another important reason to search for interpretability of knowledge graph embeddings: there is evidence of bias in knowledge graphs that affect the latent representations and being able to interpret such models would help prevent bias.

Xiao, Huang e Zhu (2016b) proposed a semantic representation of knowledge graphs composed of a two-level hierarchical process, able to extract global features and locally assign interpretable features for each triple. We also mention several works focusing on knowledge base completion using latent feature methods that claim to be interpretable. Xie et al. (2017) proposed an embedding model called **ITransF**, able to learn shared hidden concepts from the KG and to produce sparse representations claimed to be interpretable. In the same fashion, **CrossE**, proposed by Zhang et al. (2019), is another latent feature model aiming for knowledge base completion. As highlighted by the authors, this specific embedding can model interactions between entities and relations, the so-called *crossover interactions*. They deep dive into interpretability topics in their work, even proposing a taxonomy of explanation types. However, the interpretability of the model comes from its capacity to produce new links within the KG and an enhanced ability to find similarities among different relations. Takahashi, Tian e Inui (2018) proposed to train an embedding model jointly with an autoencoder, allowing the model to capture constraints in a better fashion, helping to generate sparse representations that are claimed to be interpretable. Chandrahas et al. (2017) proposed a method to induce interpretability of embeddings by using entity co-occurrence statistics. In contrast, Neil et al. (2018) introduced an attention parameter to train graph convolutional networks, allowing more accurate predictions and facilitating visualization and interpretation of the factors influencing a link prediction.

Another latent feature model claimed to have interpretable features is **SimplE**, proposed by Kazemi e Poole (2018), which is an enhanced tensor factorization approach where the training process is carried out by imposing some weight tying based on background knowledge, rendering the latent representations to be interpretable. Finally, as the last interpretability approach from the latent feature family of models, we cite the method proposed by Polleti e Cozman (2019) that works by introducing some random perturbations into vectors representing entities

and relations to extract explanations, in a similar fashion that LIME (RIBEIRO; SINGH; GUESTRIN, 2016) does. In another interesting work, Stadelmaier e Padó (2019) proposed the **CPM** (Context Path Model) method, which produces explanations for completions made by embedding models claimed to be model-agnostic. This method works as a wrapper to any embedding model and learns whether a given path provided as an explanation is informative.

In work focusing on the production of explanations for link prediction applied to e-commerce, Zhang et al. (2020) proposed **XTransE**, based on **TransE**, claimed to be a novel explainable embedding suitable for only one relation and equipped with an attention mechanism. The adoption of this mechanism allows the generation of explanations and rules that help support predictions. Unfortunately, the authors only performed tests with a specific dataset, providing no performance comparison to other embedding models.

From the *graph-feature models* family, Mohamed, Nováek e Vandenbussche (2018) proposed an evolution of the SFE (GARDNER; MITCHELL, 2015), called **DSP**, *distinct subgraph paths*, with better performance than its predecessor as well as being able to provide explanations of the completions via Horn clauses in a similar fashion as SFE.

Pezeshkpour, Tian e Singh (2019) introduced **CRIAGE**, a method based on adversarial attacks to evaluate the robustness of link predictions produced by knowledge embedding models. They claim that the method helps to increase the model interpretability since it provides the ability to evaluate the most influential facts for each relation. More recently, Yogendran et al. (2020) proposed **AIRL**, *Accurate and Interpretable Representation Learning*, an hybrid method that extends TransE and incorporates both semantic and hierarchical information from entities. In a procedure adapted from the Exploratory Factor Analysis, they apply a basis rotation to the pre-trained KG vectors to produce more interpretable latent representations. They claim that the latent representations are more interpretable through the subjective evaluation of a visual representation of the vectors before and after the transformation. Hence, the model does not provide explanations for a single prediction, but only entities representations deemed to be more interpretable.

4 Contributions: Improving XKE

Even though the XKE methods outlined in Section 3.1 work reasonably well, they suffer from some drawbacks. Before entering into details, let us first state some definitions. Taking into consideration that we are using a logistic regressor as the interpretable model, inspired by Robnik-Šikonja e Bohanec (2018), we propose two properties for an explanation:

- **Predictive Power** This is the degree to which a path's presence accounts for a triple's positive (or negative) prediction. In our case, we use the logistic regression coefficient for each feature path. The higher the absolute value of the feature coefficient, the more significant the influence in triple prediction.
- **Explainability Power:** The fact that a given path has a high *preditive power* does not necessarily imply that it constitutes a good explanation. So, we define *explainability power* as how close a given explanation is related to the context of the triple subject of explanation. For now, we will leave this definition somewhat subjective, but later we will propose a metric to make it less subjective.

To further understand the above properties, let us take our running example from Figure 2.1. Suppose we are trying to explain the triple $t = \langle UIC, plays, hockey \rangle$. The path $\pi_1 = \{UIC \xleftarrow{plays\_for} Ray\_Staszak \xrightarrow{plays} Hockey\}$, appears to be a good explanation since it is very likely that if a player is a professional hockey player, it is strong evidence that the team he plays for is indeed a hockey team. However, let us now consider the explanation $\pi_2 = \{UIC \xleftarrow{plays\_for} Ray\_Staszak \xrightarrow{gender} Male \xleftarrow{gender} Patrick\_Kane \xrightarrow{plays} Hockey\}$. This path type may be present as a feature for several triples and therefore be assigned a high coefficient by the logistic regressor, hence, having a high *preditive power*. However, it constitutes a very weak explanation because it is based on the generic fact that a person has a gender.

With these definitions in mind, we now discuss the drawbacks of the XKE methods that we will address in the present work:

- **Poor Fidelity:** The XKE algorithms are *global surrogate* models, hence their fidelity is far from 100%. Indeed, in many cases, they fail to correctly mimic the prediction of the black-box model, in which cases explanations generated by the explainer model make no sense. One of the reasons is that depending on the density of the KG, several training and test instances may end up with no features extracted by SFE, or the logistic regressor may assign a zero coefficient to all the features for a given instance.
- **Scalability:** At the other end of the spectrum, even for moderately dense KBs, using SFE with BFS may be computationally intractable. Resorting to SFE with random walks would be an option, but this would hinder fidelity (GARDNER; MITCHELL, 2015). Another effect of a KB high density is that SFE may lead to an explosion in the number of extracted features, rendering the use of a logistic regressor also intractable (MAZUMDER; LIU, 2017).
- **Presence of Poor Explanations:** XKE makes no preliminary selection of the features that are fed into the logistic regressor, and through the fitting process, the only aspect the classifier takes into account is predictive power, i.e., the features that will maximize the likelihood of the model, regardless of the (still subjective) explainability power of the features.
- **Interpretability Metrics:** XKE evaluates the interpretability of the explainer models with two metrics; the first one is *mean number of rules* and the second is *mean rule length*. These metrics capture only the number of explanations the explainer model provides, regardless of their explainability power.

In the remainder of this chapter, we propose two approaches to solve these issues (by enhancing some of the methods described in our publication Ruschel et al. (2019)).

4.1 Context-aware XKE (C-XKE)

To address the *presence of poor explanations* and the *scalability* issues of XKE, we propose **Context-aware Explanations of Knowledge Embeddings**

(C-XKE), a method inspired by the Context-aware Path Ranking Algorithm (C-PR) (MAZUMDER; LIU, 2017) detailed in Section 2.4.1.3. The method also allows us to develop a less subjective explainability metric.

Using word embeddings, C-PR leverages the contextual similarity among labels of entities (nodes of the graph) as a heuristics to perform the random walks to extract path features. We apply the same concept here, but since the primary goal of XKE is to provide sequences of path types (composed by edges of the graph) interconnecting entities as explanations, we use word embeddings to compute the similarity between relations, i.e., the labels of the edges of the graph. With this method, we can select only relations that are contextually related to the triple being explained, specifically to the relation of the triple subject to explanation. This concept also allows us to define a metric to choose which path types will be included as explanations, reducing the size of the feature vector significantly and speeding up computations.

Gardner e Mitchell (2015) already demonstrated that the performance of SFE running BFS is better than the version using random-walks. So, C-XKE allows us to perform BFS in denser graphs in applications closer to real-life scenarios of large KBs. Moreover, as we will show later, explanations produced by the method are more interpretable from a human point of view.

Formally, we consider a knowledge graph $\mathcal{G}$, formed by the set of enties $\mathcal{E} = \{e_1, ..., e_{Ne}\}$, and the set of relations $\mathcal{R} = \{r_1, ..., r_{Nr}\}$, with N_e, N_r representing the number of entities and relations, respectively. Also, we denote as $\mathcal{T} = \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ the set of all possible triples in $\mathcal{G}$. The function of the embedding black-box classifier is denoted by $g : \mathcal{T} \rightarrow \{0, 1\}$. Again, we denote by $\Pi_\mathcal{G}$ the set of all possible path types interconnectig two entities of a KG, and by $\mathcal{P}(\Pi_\mathcal{G})$ its power set. Let $x_{h,r,t} \in \mathcal{T}$ denote a triple within the KG and $SFE : \mathcal{T} \rightarrow P(\Pi_\mathcal{G})$ the feature extraction performed by SFE. The set of all paths encountered by SFE for a triple $x_{h,r,t}$ will be denoted by $\Pi_{h,r,t|\mathcal{G}} \in \mathcal{P}(\Pi_\mathcal{G})$.

Inspired by C-PR, we compute the contextual similarity between two relations using distributed semantics. We denote as W_{ra}, W_{rb} the word vectors of the labels of two relations $r_a, r_b \in \mathcal{R}$, given a corpus $\mathcal{C}$, and compute the contextual

similarity using *cosine similarity*:

$$sim(r_a, r_b) = \frac{W_{ra} \cdot W_{rb}}{||W_{ra}|| \cdot ||W_{rb}||}. \tag{4.1}$$

For a given triple $x_{h,rk,t}$, we represent a path type, of length l, interconnecting the head and tail entities by $\pi_i = \langle r_1, ..., r_l \rangle$, with $\pi_i \in \mathcal{P}(\Pi_\mathcal{G})$ and $r_1, ..., r_l \in \mathcal{R}$. We define *path contextual-relatedness (pc)*, a metric that provides a measurement of how closely related the labels of the edges of a path are related to the label of the relation r_k, leveraging distributed semantics. We use the simple average of the textual similarity between each edge label with the relation r_k:

$$pc(r_k, \pi_i) = \sum_{r \in \pi_i} \frac{sim(r_k, r)}{\mid r \mid}; \tag{4.2}$$

where $\mid r \mid$ represents the number os edges in π_i.

We state that explanations that are more context-related to the relation subject to explanation have more explainability power; therefore, we can use the metric *pc* as a measurement of the explainability power of each feature.

We can use concepts of Equations 4.1 and 4.2 to reduce feature explosion in two distinct steps of the XKE pipeline. The first one is by applying *pc* after the feature extraction performed by SFE, when the machine learning practitioner needs to reduce the size of the feature vector, aiming to foster the quality of explanations and speed up the computations during the logistic regressor phase. The second is wisely selecting relations to build a subset of the original graph before applying SFE. We can perform this by using the *sim* metric to select only relations related to the context of the relation of the explained triple. The advantages are twofold: first, the paths encountered by the SFE function will have more explainability power, and second, the graph loaded in memory will be smaller, allowing XKE to be applied to large-scale datasets.

4.1.1 C-XKE After Subgraph Feature Extraction (SFE)

Recal that we denoted by $\Pi_{h,r,t|\mathcal{G}} \in \mathcal{P}(\Pi_\mathcal{G})$ the set of all paths encountered while performing SFE function in $\mathcal{G}$, for a triple $x_{h,r,t} \in \mathcal{T}$. We select a subset of

the paths considering the explainability power, pc:

$$\dot{\Pi}_{h,r,t|\mathcal{G}} = \{\pi'_{h,r,t|\mathcal{G}} | \pi'_{h,r,t|\mathcal{G}} \in \Pi_{h,r,t|\mathcal{G}}, pc(\pi'_{h,r,t|\mathcal{G}}, r) \geq \lambda\}; \tag{4.3}$$

where λ is a hyper-parameter of the model that defines the minimum pc value for a given path not to be filtered out from the feature vector used to fit the logistic regressor. The pipeline than follows what was described in Section 3.1: first, we construct an arbitrary subset of triples $\mathcal{D} \subseteq \mathcal{T}$, and build a training set, $\mathbb{D}'$, to fit the interpretable classifier, $g' : P(\Pi_{\mathcal{G}}) \rightarrow \{0, 1\}$, as follows:

$$\mathbb{D}' = \{(\dot{\Pi}_{h,r,t|\mathcal{G}}, g(x_{h,r,t})) | x_{h,r,t} \in \mathcal{D}\}. \tag{4.4}$$

We then draw the explanations from g'. This approach is suitable when machine learning practitioners have already performed the SFE feature extraction for a given graph and would like to fit XKE logistic regressor with a reduced set of features to produce better explanations and speed up computations. In this sense, we can regard C-XKE as a tool to perform a feature selection step between SFE and logistic regression steps.

4.1.2 C-XKE Before Subgraph Feature Extraction (SFE)

We can apply the concepts of C-XKE before running SFE to wisely select only the relations with similar context to the target relation. By doing so, we can perform the SFE feature extraction in a reduced graph, containing only relations of interest that will result in more interpretable explanations.

Formally, let us consider all definitions of a graph $\mathcal{G}$ stated at the beginning of this section. We want to perform SFE on this graph to produce XKE explanations for a given triple $x_{h,rk,t}$. We can use the *similarity* (Equation 4.1) to select a subset of the graph containing only relations that are more contextually related to the relation we want to explain. We build a subset of $\mathcal{G}$ as follows:

$$\mathcal{G}' = \{x_{h,r,t} \in \mathcal{G} | sim(r, rk) \geq \gamma\}, \tag{4.5}$$

where γ is a hyper-parameter of the model that helps to tune the size of the subset of the graph by acting as a cutoff value for the context-relatedness of the relations

of the graph and the relation under explanation. The pipeline proceeds as above: we define by $\Pi_{h,r,t|\mathcal{G}'} \in \mathcal{P}(\Pi_{\mathcal{G}'})$ the set of all paths encountered while performing SFE function in $\mathcal{G}'$, for a triple $x_{h,r,t} \in \mathcal{T}$. Then, we build a training set $\mathbb{D}'$ from an arbitrary set of triple instances $\mathcal{D} \subseteq \mathcal{T}$, in which we train an interpretable classifier $g' : P(\Pi_{\mathcal{G}'}) \rightarrow \{0, 1\}$, as follows:

$$\mathbb{D}' = \{(\Pi_{h,r,t|\mathcal{G}'}, g(x_{h,r,t}))|x_{h,r,t} \in \mathcal{D}\}. \tag{4.6}$$

In the same way, we draw explanations from g'.

4.1.3 Discussion

Both C-XKE improvements presented above can be regarded as domain-oriented feature selection that improves XKE explanations and reduces computation time. One could question why we should use a word-embedding model to help to provide explanations for predictions of Knowledge Base embeddings. We argue that word-embeddings using distributed semantics are a valuable tool to evaluate the context of labels of entities and relations of a knowledge base. Mazumder e Liu (2017) demonstrated a high correlation between contextual similarity for a pair of entities and the existence of a relation between them. We want XKE to provide explanations that make sense from the human point of view; such explanations only make sense when relations are somehow context-related (recall the example provided at the beginning of this chapter, where relation *gender* and *plays_for* are not closely context-related, and the resulting explanation makes no sense).

A possible alternative to word-embeddings would be to use some distance measurement to compare relation representations to capture some context from the results. At first sight, this appears to be an elegant approach because one could argue that the embedded low-dimensional representation of relations indeed captures the distributed semantics from the graph topology, and similar relations would be interconnecting similar entities. In other words, synonyms relations would have similar vector representations in the low-dimensional space. Nevertheless, using such an approach has a significant drawback: we would need to design a different metric for each type of embedding model. So, applying word embeddings

to compute contextual similarity among features and relations allows XKE to remain a model-agnostic approach.

4.2 XKE-e

In this section, we address the low fidelity of XKE, proposing an algorithm called XKEe, which we will later demonstrate improves results and can be applied in conjunction with C-XKE.

A significant deficiency of PRA-based methods is low connectivity of the knowledge graphs (GARDNER et al., 2014). As discussed above, Gardner, Talukdar e Mitchell (2015) proposed SFE using BFS instead of random walks, which reduced this deficiency. Nevertheless, for triples with no interconnecting paths to be used as features, SFE still fails to provide explanations. In our XKE scenario, which relies heavily on SFE-BFS, the effects are twofold: either the interpretable model wrongly mimics the knowledge embedding prediction, or it provides no explanation for the prediction due to the absence of paths to be used as features. This hinders the fidelity of the resulting logistic regressor. One way to overcome this issue would be to increase L to find paths of greater length, but in large KGs, this would be computationally intractable.

Instead, we propose to leverage the information obtained while applying XKE (SFE + logistic regression + embedding labels). For the triples with inconsistent prediction between the logistic regression and the embedding model, we reconstruct path types (features) that the logit assigned coefficients different than zero using the embedding model itself.

More formally, for a given graph $\mathcal{G}$, we denote by $\mathcal{T}$ the set of all possible triples. Let $\Pi_{\mathcal{G}}$ represent the set of all possible paths between two entities in $\mathcal{G}$, and $P(\Pi_{\mathcal{G}})$ its power set, and $SFE : \mathcal{T} \rightarrow P(\Pi_{\mathcal{G}})$ represent the feature extraction function performed by SFE for an arbitrary triple $x_{h,r,t} \in \mathcal{T}$ given $\mathcal{G}$. A single path type (that will be translated to a feature) is represented by π. The set of all encountered paths after performing SFE feature extraction, in $\mathcal{G}$, for a given triple $x_{h,r,t} \in \mathcal{T}$ is represented by $\Pi_{h,r,t|\mathcal{G}} \in P(\Pi_{\mathcal{G}})$. Again, $g : \mathcal{T} \rightarrow \{0, 1\}$ is the embedding classifier function.

To apply the explainer model XKE-TRUE, we build the training set $\mathbb{D}$ as defined in Equation 3.3, and train the interpretable classifier $g'' : P(\Pi_{\mathcal{G}}) \rightarrow \{0, 1\}$ as discussed in Section 3.1.2. In our case, we are applying a logistic regressor as the interpretable classifier, hence, for each path type π_i, the regressor will assign a weight w_{π_i}. Furthermore, we define the subset of all path types with weights different than zero as $\Pi_{h,r,t|\mathcal{G}}^{(w\neq 0)} = \{\pi_i \in \Pi_{h,r,t|\mathcal{G}} \mid w_{\pi_i} \neq 0\}$.

Let us denote by $\hat{y}_g = g(x_{h,r,t})$ the result of a prediction made by the embedding classifier for a given triple $x_{h,r,t} \in \mathbb{D}$. Similarly, we denote $\hat{y}_{XKE} = g''(x_{h,r,t})$ as the result predicted by XKE for the same triple.

When applying an explainer model to interpret the predictions of a black-box classifier, one looks for a model that is faithful to the predictions of the black-box model and can provide explanations for all predictions. Formally, we want that for every triple $x_{h,r,t} \in \mathbb{D}$:

- 1) $\hat{y}_{XKE} = \hat{y}_g$ (the explainer model faithfully predicts the black-box model);
- 2) $\Pi_{h,r,t|\mathcal{G}}^{(w\neq 0)} \neq \varnothing$ (there is at least one feature with assigned weight different than zero to be used as explanation).

Processing all the triples $x_{h,r,t} \in \mathbb{D}$ where one of the two goals above are not met, we build an enhanced graph, $\dot{\mathcal{G}}$, comprised by the original graph $\mathcal{G}$ augmented with new edges that will form new paths to be used as features. The algorithm to build the enhanced graph is described in Algorithm 1. This procedure is sensible because we are operating under the *open world assumption*; hence, we assume that the non-existing path types connecting two entities are only unknown and, therefore, not present in $\mathcal{G}$.

Two final points about the proposed method: the first one is that our approach is closely related to the one applied in **CrossE** (ZHANG et al., 2019), where they construct paths using the embedding to serve as *support* for producing explanations about the predictions. The main difference between the proposals is that CrossE only produces explanations for true facts, while XKEe can explain both true and false predictions. The second point is that we can apply XKEe in both XKE variants, TRUE or PRED. The first step is to apply XKE using $\mathcal{G}$ for

Algorithm 1 XKE-e Algorithm (RUSCHEL et al., 2019)

procedure BUILD-EXTENDED-SET(g, $\mathcal{G}$, $x_{h,r,t}$)
  $\hat{\Pi} \leftarrow \{\}$ ▷ Set of path types for the triple
  $\hat{g} \leftarrow \{\}$ ▷ Set of new found facts
  for all $\pi_{h,r,t} \in \Pi^{(w \neq 0)}_{h,r,t|\mathcal{G}}$ **do**
    check existence of path $\pi_{h,r,t}$ between h and t
  return $\hat{\Pi}$, $\hat{g}$
procedure XKE-E
  $\Pi_{h,r,t|\mathcal{G}} \leftarrow SFE(x_{h,r,t} \mid \mathcal{G})$
  $w \leftarrow g'' : P(\Pi_{\mathcal{G}}) \rightarrow \{0, 1\}$ ▷ train XKE and obtain feature weights
  $\dot{\mathcal{G}} \leftarrow \mathcal{G}$
  for all $x_{h,r,t} \in \mathbb{D}$ **do**
    if $\hat{y}_{XKE} \neq \hat{y}_g$ or $\Pi_{h,r,t|\mathcal{G}} = \varnothing$ **then**
      $\hat{\Pi}, \hat{g} \leftarrow$ BUILD-EXTENDED-SET(g, $\mathcal{G}, x_{h,r,t}$)
      $\Pi_{h,r,t|\mathcal{G}} \leftarrow \hat{\Pi}$
      $\dot{\mathcal{G}} \leftarrow \dot{\mathcal{G}} \cup \hat{g}$
  Update all predictions using the new $\Pi_{h,r,t|\mathcal{G}}$
  Draw explanations from *g"* in the form of Horn clauses

the TRUE version or $\hat{\mathcal{G}}$ for the PRED version, and then use the weights assigned by the logistic regressor as heuristics to build the predicted (and complementary) graph, in our case, $\dot{\mathcal{G}}$.

5 Experiments

5.1 Benchmark Datasets

To evaluate the results, we performed the experiments using two of the most common benchmark datasets used for the evaluation of embedding models, namely *NELL186* and *FB15K-237*.

• **NELL186:** It is a subset of the NELL knowledge base (MITCHELL et al., 2015) introduced by Guo et al. (2015). It contains 14.463 entities and 186 relations obtained from the original KB.

• **FB15K-237:** It is a subset of the original benchmark dataset *FB15K* (BORDES et al., 2013), which in turn is a subset of Freebase (BOLLACKER et al., 2008). Toutanova e Chen (2015) noted that the original dataset suffered from data leakage from the training to the test set due the presence of inverse relations. To avoid that, they filtered out inverse and near-duplicate relations from the *FB15K*, resulting in a more challenging dataset (AKRAMI et al., 2018). This dataset is approximately six times denser than *NELL186*, thus increasing the likelihood of feature explosion during the feature extraction performed by SFE.

Table 1 – Benchmark Datasets

Dataset	$\mathcal{E}$	$\mathcal{R}$	$\mathcal{T}$	**# Train**	**# Test**	**# Valid**	**Density**
NELL186	14,463	186	$39 \cdot 10^9$	31,134	5,000	5,000	$1.1 \cdot 10^{-6}$
FB15K-237	14,541	237	$50 \cdot 10^7$	272,115	17,535	20,466	$6.2 \cdot 10^{-6}$

Dataset characteristics: $\mathcal{E}$, $\mathcal{R}$ and $\mathcal{T}$ represent the number of entities, relations, and all possible triples, respectively. # Train, # Test, # Valid represent the number of triples in each subset. Test and validation subsets contain one negative example per positive one, not included in the count. Density is the actual number of triples considering all subsets divided by all possible triples, $\mathcal{T}$.

5.2 Implementation

We implemented a software package containing *C-XKEe* in Python and made it publicly available.[1] We trained embeddings using the open-source package for multigraph KBs called OpenKE[2] (HAN et al., 2018), with several modifications of our own. We also implemented our version of SFE based on Gardner e Mitchell (2015) using Python with multiprocessing capabilities to parallel-processed subgraph extraction. Also, we implemented one crucial modification to *BFS*. To keep feature extraction tractable while performing the *breadth-first search*, the original implementation uses a filter to prevent the expansion of nodes with high outdegree. For instance, if BFS accesses a node with an outdegree of 1,000, and the maximum allowed outdegree is 100, the node would not be expanded further in the search (refer to section 2.4.1.2 for more details). In contrast, our implementation randomly selects 100 nodes from the original set of 1,000 nodes and further expands those selected nodes. Ruschel et al. (2019) already demonstrated that this modification fosters SFE performance and hence increases XKE fidelity. To properly manage and run calculations with word embeddings, we used the open-source library Gensim[3]. Logistic regression and grid search were performed using sklearn[4].

5.2.1 Hardware

We performed the experiments on a desktop equipped with an Intel(R) Core(TM) i7-8700K CPU 3.70GHz, 64Gb of RAM, GPU Nvidia GeForce 1080Ti with 8Gb of RAM, with Ubuntu 18.04 operational system.

5.2.2 Model Training

We performed a grid search to train TransE with FB15K-237, as proposed by Nguyen et al. (2016a). We selected a 1-to-1 ratio for negative examples generation, using a normal distribution for triple corruption. The training process used SGD with AdaGrad adaptative learning rate, with epochs limited to 1,000. The

[1] http://www.github.com/aruschel/CXKEe
[2] http://openke.thunlp.org/
[3] http://radimrehurek.com/gensim/
[4] https://scikit-learn.org/

hyperparameters that led to the best accuracy on the test dataset were: number of dimensions $k = 50$, learning rate $\eta = 1$, ℓ_1 norm, margin $\gamma = 1$, and mini-batch size $B = |\mathcal{D}_{train}|/100$, where $\mathcal{D}_{train}$ represents the training dataset including positive and negative triples.

To fairly compare each SFE strategy, we used the same TransE embeddings trained by Gusmão et al. (2018) for NELL186; we generated negative examples via Bernoulli distribution at a 1-1 rate. Model training was limited to 1,000 epochs, split into 100 mini-batches, and using SGD with AdaGrad optimizer. The best model accuracy was obtained with a learning rate $\eta = 1$, ℓ_2 norm, margin $\gamma = 1$ and embedding dimension $k = 50$.

We performed grid searches to train ANALOGY embeddings for both FB15K-237 and NELL186 datasets. We set a 6-to-1 ratio for negative examples generation, where four examples were generated by either corrupting the head or the tail of the triple, and 2 of them were generated by corrupting the relation. For FB15K-237, we used Bernoulli distribution to perform the corruption, while for NELL, a normal distribution led to better results. SGD was used to carry out the training, using AdaGrad adaptative learning rate, and the number of epochs was 500 for NELL and 100 for FB. Other hyperparameters that led to best accuracy in the test dataset were: number of dimensions $k = 200$, learning rate $\eta = 0.01$, and mini-batch size $B = |\mathcal{D}_{train}|/10$ for NELL186, and number of dimensions $k = 200$, learning rate $\eta = 0.1$, and mini-batch size $B = |\mathcal{D}_{train}|/100$ for FB15K-237.

We used a publicly available set of pre-trained 300-dimension word2vec word embedding[5] trained on a corpus of 3 billion words from Google News. The word library contained approximately 3 million English words. Almost all relations from both datasets had multi-word labels, so we computed the centroid of the words for each relation, disregarding some repetitive words such as *concept*, found in all text descriptions of NELL186. Using *Gensim*, we created word vectors for each relation and stored them in a different file to avoid the need for loading the entire Google News dataset, saving RAM usage during the C-XKE pipeline.

For the feature extraction process, we applied SFE in both datasets setting

[5] https://code.google.com/archive/p/word2vec/

the maximum body rule length to four and the maximum node outdegree for node expansion to 1,000 nodes. FB15K-237 is a denser KB, so while extracting features for each relation, we selected only the top-180 most similar ones, following the method C-XKE Before SFE described in Section 4.1.2. After feature extraction, we repeated the process without C-XKE for 46 out of 237 relations to increase the number of extracted features.

Before running XKE, we performed another feature selection step using C-XKE after SFE, as described in Section 4.1.1. For each relation r, we ranked the features according to the *pc* metric and selected the most similar ones. To simplify the notation, instead of selecting the actual number of features, we defined a parameter *top_pc_rels* in terms of the proportion of selected features compared with the number of instances in the training subset, $|D_{train}|$. Formally, for a given relation, r, we define the number of selected features by $| p_{selected} |= \text{top\_ pc\_ rels} \cdot | \mathcal{D}_{train} |$. To evaluate the results of C-XKE and its ability to select more interpretable features, we also built some experiments considering the most popular features, following Gardner et al. (2015). We sorted the features by the number of training instances they appeared and selected the top ones according to the metric named *top_pop*. Formally, the number of selected features is calculated by $| p_{selected} |= \text{top\_ pop} \cdot | \mathcal{D}_{train} |$.

The explainer model used was a logistic regressor trained using SGD to minimize the log-loss function with elasticnet regularization, setting the stopping criteria $\epsilon = 0.001$. For each relation, we performed a grid-search to select the hyperparameters that resulted in the best model, from the following ranges: elasticnet ℓ_1 regularization ratio $\lambda = \{0.1, 0.5, 0.7, 0.9, 0.95, 1.0\}$, regularization weight $\alpha = \{0.01, 0.001, 0.0001\}$.

We performed all experiments only for the XKE-TRUE version of XKE, but all proposed approaches are suitable to enhance explanations of XKE-PRED as well.

5.3 Evaluation Metrics

5.3.1 Accuracy

Denote by $\hat{y}$ a classifier prediction and y as the binary random variable modeling the outputs of the model; to evaluate the accuracy, we consider a set of previously unseen triples $\mathcal{D}_{test}$ as a test set. Accuracy is the rate of correct outputs predicted by the model over the entire test set:

$$Accuracy = \sum_{x \in \mathcal{D}_{test}} \frac{1 - \mid \hat{y} - y \mid}{\mid \mathcal{D}_{test} \mid}. \tag{5.1}$$

5.3.2 Fidelity

Since XKE is a *pedagogical* model, it is important to measure its ability to mimic the embedding model's behavior correctly. For a given test set $\mathcal{D}_{test}$, we denote an embedding prediction as $\hat{y}_g$, for an arbitrary input triple $\langle e_h, r_r, e_t \rangle \in \mathcal{D}_{test}$. We also denote $\hat{y}_{XKE}$ the prediction made by the pedagogical model for the same input triple. Fidelity captures the probability of XKE providing the same prediction as to the embedding model, and we compute it by:

$$Fidelity = \sum_{x \in \mathcal{D}_{test}} \frac{1 - \mid \hat{y}_{XKE} - \hat{y}_g \mid}{\mid \mathcal{D}_{test} \mid}. \tag{5.2}$$

5.3.3 F1 Score

This metric is commonly used when there are large class imbalances, and it is obtained from the harmonic mean between *precision* and *recall* as a way to ensure that the score will be high only when both metrics are high. Let us denote TP as the number of true positive predictions, FP as false positive predictions, TN as the number of true negative predictions and FN as the false negative ones. The *F1 score* is:

$$F1 = \frac{2 \times TP}{2 \times TP + FP + FN}. \tag{5.3}$$

5.3.4 Mean Number of Rules

We use this metric to assess the interpretability of the pedagogical model. It conveys the number of features that the logistic regressor assigns weights different

than zero, in other words, the number of active rules of the regressor. Let $\Pi_{h,r,t|\mathcal{G}} = SFE(x_{h,r,t} \mid \mathcal{G})$ represent the set of path types extracted by SFE. For each path type $\pi_i \in \Pi_{h,r,t|\mathcal{G}}$, the logistic regressor will assign a weight w_{π_i}. The subset of path types with weight different than zero will be denoted by $\Pi^{(w\neq 0)}_{h,r,t|\mathcal{G}} = \{\pi_i \in \Pi_{h,r,t|\mathcal{G}} \mid w_{\pi_i} \neq 0\}$. The mean number of rules will then be calculated by

$$\textit{Mean Number of Rules} = \sum_{x_{h,r,t|\mathcal{G}} \in \mathcal{D}_{test}} \frac{\mid \Pi^{(w\neq 0)}_{h,r,t|\mathcal{G}} \mid}{\mid \mathcal{D}_{test} \mid}. \tag{5.4}$$

5.3.5 Interpretability Index

We now propose a metric to compute the interpretability of the explanations provided by XKE, using the *pc* metric previously defined. The ideia of the *interpretability index* is to use the contextual relatedness of the explanations of a given triple as regards the relation subject of explanation, weighted by logit coefficients. Let us again define $\Pi_{h,r,t|\mathcal{G}} = SFE(x_{h,r,t} \mid \mathcal{G})$ as the set of features extracted by SFE for a triple $x_{h,r,t} \in \mathcal{T}$. Let $\Phi_{h,r,t}$ be the vector where each component ϕ_i, either equal to 0 or 1, indicates the existence of a path π_i for the given triple. Furthermore, let $W_{h,r,t}$ be the vector of weights assigned by the logistic regressor, and w_{π_i} represent the coefficient for each feature $\pi_i \in \Pi_{h,r,t|\mathcal{G}}$. Also, we denote $|\Phi_{h,r,t} \odot W_{h,r,t}|_{\ell 1}$ the $\ell 1$ norm of the vector of weights multiplied elementwise by the vector indicating the existence of each feature. Using Equation 4.2, we can calculate the *path context-relatedness* metric, $pc(r, \pi_i)$ for each π_i with regards to relation $r \in \mathcal{R}$. We define the *Interpretability Index* for the given triple as

$$\mathcal{II}_{x_{h,r,t}} = \sum_{\pi_i \in \Pi_{h,r,t|\mathcal{G}}} \frac{\mid w_{\pi_i} \mid \cdot \phi_i \cdot pc(r, \pi_i)}{|\Phi_{h,r,t} \odot W_{h,r,t}|_{\ell 1}}. \tag{5.5}$$

The denominator[6] yields a ratio that conveys the relative importance of each active feature for the triple $x_{h,r,t}$ in terms of the absolute value of its coefficient w_i. In other words, we normalize the vector of weights of active features before multiplying by the *path context-relatedness* of each feature.

Once we have defined a way of calculating how interpretable the set of active explanations for a triple is, we can evaluate the overall interpretability of the model

[6] In cases that $\Phi_{h,r,t} \cdot W_{h,r,t} = 0$, $\mathcal{II}$ is set to zero to avoid division by zero.

for the entire test set that can be used in conjunction with the performance metrics highlighted above to evaluate the pedagogical model. The *overall interpretability index* is then:

$$\mathcal{OII} = \sum_{x_{h,r,t} \in \mathcal{D}_{test}} \frac{\mathcal{II}_{x_{h,r,t}}}{|\mathcal{D}_{test}|}, \tag{5.6}$$

where $\mathcal{D}_{test}$ represents the set of triples of the test subset.

Although the metric we have just proposed is excellent in providing insights into the quality of the explanations for the entire test set (or whatever subset, for that matter), it is highly influenced by triples without any active explanation. So, we propose a variation of the metric by considering only triples that contain active features. First, we build a subset of $\mathcal{D}_{test}$ containing only triples with active explanations: $\mathcal{D}_{test\_filtered} = \{x_{h,r,t} \mid x_{h,r,t} \in \mathcal{D}_{test}, (\Phi_{h,r,t} \cdot W_{h,r,t}) \neq 0\}$, where $(\cdot)$ represents the dot product between the two vectors. Then, we obtain the *overall interpretability index* filtered by triples with active features:

$$\mathcal{OII}_{filtered} = \sum_{x_{h,r,t} \in \mathcal{D}_{test\_filtered}} \frac{\mathcal{II}_{x_{h,r,t}}}{|\mathcal{D}_{test\_filtered}|}. \tag{5.7}$$

This metric allows us to assess the quality of the explanations disregarding the effect of triples that do not have any active rule, helping to grasp how good provided explanations are.

5.4 Results

We present the results of C-XKEe in Table 2 for both TransE and ANALOGY embedding models. For each model, we evaluate the results of the embeddings trained in NELL186 and FB15K-237 datasets. With NELL186, we have performed three different experiments, the first one selecting the features according to the C-XKE procedure as described in section 5.2.2; the second experiment was performed using the most frequent features, and the third one with no future selection. For FB15K-237, we only performed experiments with feature selection to avoid feature explosion and high training time. The table presents the results grouped for each main phase of the experiment pipeline: feature extraction, C-XKE for feature selection, SFE + logit, XKE, and finally, XKEe. This way, we can keep track of

one crucial measure, the *% of test triples with features*, as it plays an essential role in the model's performance.

Regarding the performance of the embedding models evaluated against a test dataset, we can see that both ANALOGY and TransE performed well in the NELL186 dataset and not so well in FB15K-237 since the latter is a much more challenging dataset.

The *feature extraction* phase of the SFE focuses on the dataset itself, regardless of the embedding model. This is why we spot identical figures for each of the datasets when looking at Table 2. For NELL186, we performed the SFE feature extraction loading the entire graph, resulting in a ratio of almost 6-to-1 unique features per training example and approximately 60% of the training triples with at least one feature. In contrast, for FB15K-237, we resorted to C-XKE to select only 180 most context-related relations to load the subgraphs and then perform the feature extraction for approximately 80% of the 237 dataset relations. Despite that, the ratio of unique features per training example was approximately 39-to-1, meaning roughly 35 million unique features. In the same fashion, almost 99% of the test set had at least one feature.

In the next step, we performed the feature selection using C-XKE, as mentioned at the beginning of this chapter. As expected, in scenarios where we picked the most popular features, we had a smaller decrease in % of test triples with at least one feature. The experiments using context-relatedness feature selection had a significant drop in the percentage of test triples with features. One thing worth mentioning is the fact that in NELL186, without feature selection experiment, the decrease in the rate of test triples with features is caused by the fact that several test triples had features that were only present in the test set, and for obvious reasons, were disregarded in the counting and training process as well.

We trained an interpretable classifier *SFE + logit* to benchmark performance, and for most of the experiments with NELL186, the accuracy performance turned out to be better than the accuracy of the embedding itself. In this scenario, the machine learning practitioner could disregard the embedding and use a more interpretable model. For FB15K-237, the results were the opposite, with SFE +

logit performing worse in terms of accuracy. When we evaluate the accuracy of each experiment, we can see that it increases in sets with a greater % of test triples with feature, demonstrating one of the shortcomings of PRA-based methods mentioned before. Another point worth noting is that, for NELL186 with context-aware feature selection, there is an almost neglectable drop in the proportion of test triples with features after we run the classifier. In other words, most of the features fed into the classifier turned out to become active features, with logit assigning non-zero coefficients. Unfortunately, this effect does not translate into model performance, as the model accuracy was much worse than in the other experiments.

Moving to the next pipeline phase, where we fitted XKE, we can see that for NELL186, the explainer model had better performance than FB15K-237. The model's fidelity achieved a result above 82% for all six experiments. Another interesting point is that for the cases in which we restricted the number of features to 20% of the number of training examples, ANALOGY achieved a fidelity almost 5% better than TransE. This difference dropped to only 2% for the experiments with no feature selection. Additionally, we can see that the F1-score for fidelity was more in line with fidelity for ANALOGY. These two facts indicate that perhaps the class imbalance caused by the embedding behavior was much worse with TransE than with ANALOGY, making the logit fitting process more difficult. Looking at the metric *average number of features per triple with active feature* we have another indication of this phenomenon. For the cases with no limitation in the number of features, TransE required almost twice the number of features than ANALOGY, possibly indicating some overfitting tendency.

Regarding the *% of triples with active feature*, it performed as expected, with the number increasing as we do not limit the number of features before fitting the model. For both embedding models, the performance was very similar concerning this metric. Finally, we see that the *Overall Interpretability Index* achieved for NELL186, for the experiments without feature selection, was higher than the experiments using C-XKE, with a strong influence from the percentage of triples with active features. On the other hand, the figures perform as expected in the filtered version of the *Overall Interpretability Index*. The interpretability index is much higher for the experiments that used C-XKE to filter the features than in

other experiments for both embedding models.

In both datasets' XKE results for FB15K-237, two facts draw our attention: fidelity figures are worse than NELL186, and the F1-score for fidelity is greater than fidelity itself for all four experiments, suggesting that XKE was able to deal better with class imbalances generated by the embedding models. Also, we can see that the interpretable model, in this case, XKE, performed worse than the embedding models regarding the accuracy, indicating that the machine learning practitioner would still rely on the embedding model and use the interpretable model to produce explanations. Also, for both embedding models, we can see that XKE fidelity has a high correlation with the percentage of triples with at least one feature. We can also see that, in the experiments using C-XKE to select the features, the average amount of features per triple with feature is almost 1/5 of the other two scenarios, indicating that C-XKE can help to make the explainer model more interpretable by a reduction of the number of active rules. Indeed we can see that both versions of the *overal interpretability index* are better for the experiments using C-XKE.

Finally, looking at the XKEe results, we see that it successfully increased the fidelity for all experiments, to the point that, except for NELL186/Analogy, the fidelity is higher than the embedding accuracy itself. The increase in fidelity was more significant in the experiments that used C-XKE as a feature selection tool. For all combinations of embedding models and datasets, it led to greater fidelity among all other scenarios. The results for F1 Fidelity also followed the same trend, with the experiments with C-XKE as feature selection performing better for each combination. Interestingly, F1 Fidelity figures for FB15K-237 for both TransE and ANALOGY were better than fidelity, whereas, for NELL186, we can see precisely the opposite effect.

Again, there is a high correlation between the increase in the percentage of triples with at least one feature and the fidelity of XKEe, and we see XKEe not only adding features to triples that had no feature but also adding additional features to triples that previously had at least one active feature. But with XKEe, for all combinations of dataset and embedding model, the experiments with feature selection using C-XKE achieved greater fidelity despite the smaller proportion of

triples with active features, suggesting that the XKEe method was more efficient in the presence of more interpretable features. We can also spot the same effect in the number of test triples with predictions corrected by XKEe.

Table 2 – Consolidated Results for **TransE** and **Analogy** Embedding Models.

	Embedding Model	**TransE**					**Analogy**				
	Dataset	**NELL186**			**FB15K237**		**NELL186**			**FB15K237**	
	Embedding Accuracy (%)	**86.40**			**77.13**		**92.85**			**74.21**	
Feature Extraction	% of test triples with feature	61.39			[1]98.63		61.39			[1]98.63	
	# of features / # of triples	5.72			38.64		5.72			38.64	
C-XKE[2]	top_pc_rels	0.20	-	-	0.20	-	0.20	-	-	0.20	-
	top_pop	-	0.20	-	-	0.20	-	0.20	-	-	0.20
	% of test triples with feature (after feat select)	40.92	47.76	55.03	78.82	96.39	40.92	47.76	55.02	78.86	96.40
SFE + logit	Accuracy (%)	87.45	92.83	93.93	68.49	70.96	87.45	92.76	94.05	68.43	71.00
	F1 (%)	85.88	92.39	93.68	71.27	75.72	85.91	92.32	93.81	71.28	75.85
	% of triples with active feature	40.68	46.81	51.36	75.91	94.70	40.73	46.92	51.93	76.39	94.54
XKE	Fidelity (%)	82.51	84.35	86.22	69.92	75.07	87.10	89.42	88.85	68.95	70.66
	F1 (Fidelity) (%)	70.36	75.75	78.70	73.06	79.05	82.29	86.51	86.02	72.69	76.35
	% of test triples with active feature	39.92	46.56	51.57	76.76	94.65	40.39	46.79	51.00	75.64	94.78
	Average # of features per triple with active feature	10.56	19.25	68.33	25.23	109.98	9.68	15.67	36.67	19.92	98.25
	Overall Interpretability Index (%)	34.52	33.93	35.84	63.41	54.71	35.38	34.56	36.16	62.56	55.79
	Overall Interpretability Index (Filtered) (%)	84.61	71.93	69.67	80.53	57.50	85.52	72.89	70.92	80.97	58.52
XKEe	Fidelity (%)	90.96	89.73	90.45	81.39	79.69	92.13	90.77	89.98	80.20	75.34
	F1 (Fidelity) (%)	85.81	84.27	85.15	84.23	83.24	89.86	88.38	87.54	83.21	80.22
	% test triples with active feature	46.26	49.20	53.00	84.92	95.63	44.95	47.63	51.43	83.21	95.62
	Average # of features per triple with active feature	23.48	32.98	263.80	79.50	201.45	18.31	22.43	78.72	55.54	185.71
	# of test predictions corrected	841	536	421	4,660	1,878	502	135	112	4,579	1,901

Results of SFE + logit, XKE and XKEe for TransE and ANALOGY embedding models trained both with NELL186 and FB15K-237.

[1] FB15K-237 required the use of C-XKE during SFE feature extraction so as to be computationally viable.

[2] Each column represents a different feature selection strategy for each one of the datasets. For NELL186 we have 1) the top 20% most similar features, 2) the top 20% most frequent features and 3) no feature selection; For FB15K-237 we generated only first and second versions.

5.5 Qualitative Analysis

In this section, we present some explanations produced by XKE, using C-XKE to perform feature selection, comparing them with examples without context-aware feature selection, aiming to understand the differences in the explanations from each model. Each of the Tables 3, 4, 5, 6 and 7 contains a triple and two different explanation settings, and each example is labeled to be easily referenced. The tables display the embedding model and dataset, the name of the triple, the true label, the embedding prediction, the feature selection approach, and the predictions made by XKE and XKEe.

Also, for each case, the top-5 most important explanations (active rules) are presented, alongside the *logit weight* and the *path context-relatedness* metric as a way to access the interpretability of the explanation.

Examples #1, #2, #3, #4, #8, #9, and #10 are all examples of positive triples where XKE faithfully conveyed embedding predictions; therefore, it was not necessary to run XKEe to find new active rules to correct predictions. Examples #5 and #6 contain a negative triple with one case of a correct XKE prediction and another where XKE misses the prediction. Example #7 carries a positive triple for which XKE failed to provide an accurate prediction, and XKEe successfully corrected it.

We begin our detailed qualitative analysis with examples #1 and #2 as presented in Table 3. In these instances, it is possible to see the increase in interpretability of explanations using C-XKE. Example #1 contains active rules closely related with the *people/ethnicity/languages spoken* relation from FB15K-237, as we can see, all of them showing relatively high *pc* metrics. In example #2, on the other hand, even though XKE successfully predicted the embedding, explanations are not as interpretable. Looking at the active rule #1, for instance, the fact that a movie was shot in a certain country is a weak explanation for the fact that *Indian People* from India speak English. Active rule #2 is even worse, as the fact that two persons share the same gender does not explain why *Indian People* speak *English*. However, strongly indicative of the existence of a triple, the latter type of explanation is not a good type of active rule to explain the fact at hand.

Looking at Table 4, we see instances #3 and #4, two cases of XKE correctly predicting the same label of the embedding prediction. Here we see that the active rules for example #2, are all related to sports, a sensible outcome. Also, all top-5 explanations have high *path context-relatedness* metric. Example #4 did not use C-

XKE for feature selection, but that does not necessarily imply that the explanations are not interpretable, as indeed, we can see that all five active rules help to build trust in the embedding prediction.

We bring examples #5 and #6 in Table 5, for two reasons. The first one is that they are a negative example, and we can see, especially for example #5, that it only has one active rule with a negative coefficient, which is relatively interpretable, as one can find likely that two siblings are not likely to die for the same cause. The second reason is what we see in instance #6: the initial XKE prediction was positive, and XKEe found several active rules for the triple, but this was not enough to revert the wrong prediction.

Table 6 displays example #7 in which we can see that XKEe corrected the original XKE prediction, and the active rules added by XKEe are all good explanations for the fact that *Newark* city lies on *River Hudson*.

Examples #9 and #10 are two additional examples: feature selection with C-XKE and the other without any feature selection. For this case, we can see that the feature selection strategy did not bring any advantage in terms of the interpretability of the explanations. One reason is that, for this specific relation from NELL186, there was no significant number of unique features for either case, reducing the effect of the feature selection operation.

Table 3 – Explanations by XKE and XKEe.

ID	**#1 Analogy / FB15K237**			**#2 Analogy / FB15K237**		
Head	Indian people			Indian people		
Relation	people ethnicity languages spoken			people ethnicity languages spoken		
Tail	English Language			English Language		
True Label	1			1		
Embedding Prediction	1			1		
Feature Selection	top pc rels: 0.2			top pop: 0.2		
XKE Prediction	1			1		
XKEe Prediction	1			1		
	pc	**w**	**explanations**	**pc**	**w**	**explanations**
Active rule #1	0.69	4.21	people ethnicity people → people person nationality ← people person nationality → people person languages	0.50	0.47	people ethnicity people → people person nationality ← film country → film language
Active rule #2	0.69	2.75	people ethnicity people → people person languages → language human language countries spoken in → location country official language	0.66	0.43	people ethnicity people → people person gender ← people person gender → people person languages
Active rule #3	0.75	2.66	people ethnicity people ← people ethnicity people → people ethnicity people → people person languages	0.60	0.41	people ethnicity people → people person spouse s. people marriage type of union ←people person spouse s. people marriage type of union → people person languages
Active rule #4	0.71	2.23	people ethnicity people ← people ethnicity people → people ethnicity geographic distribution ← language human language countries spoken in	0.58	0.36	people ethnicity people → people person spouse s. people marriage type of union → people marriage union type unions of this type. people marriage location of ceremony ←language human language countries spoken in
Active rule #5	0.70	2.09	people ethnicity people → people person nationality ← language human language countries spoken in	0.69	0.32	people ethnicity people → people person nationality ← people person nationality → people person languages
Logit bias		-0.55			-0.64	

Explanations produced by XKE and XKEe for the same triple using two different feature selection methods indicated in the **Feature Selection** field. In the bottom part we can see the top-5 most important explanations (active rules) for each model, with the corresponding *path context-relatedness (pc)*, and *logit weight (w)*. [e] indicates that the feature was not present in the original graph and was found in the embedding by XKEe.

Table 4 – Explanations by XKE and XKEe.

ID	**#3 TransE / NELL186**			**#4 TransE / NELL186**		
Head	athlete: allan houston			athlete: allan houston		
Relation	athlete plays sport			athlete plays sport		
Tail	sport: basketball			sport: basketball		
True Label	1			1		
Embedding Prediction	1			1		
Feature Selection	top pc rels: 0.2			top pop: 0.2		
XKE Prediction	1			1		
XKEe Prediction	1			1		
	pc	**w**	**explanations**	**pc**	**w**	**explanations**
Active rule #1	0.74	6.54	athlete plays in league → league stadiums ← athlete home stadium → athlete plays sport	0.89	1.82	athlete plays for team → team plays sport
Active rule #2	0.95	4.30	athlete plays for team ← athlete plays for team → athlete plays sport	0.95	1.63	athlete plays for team ← athlete plays for team → athlete plays sport
Active rule #3	0.90	3.03	athlete plays in league ← athlete plays in league → athlete plays for team → team plays sport	0.90	1.62	athlete plays in league ← athlete plays in league → athlete plays for team → team plays sport
Active rule #4	0.92	1.79	athlete plays in league ← athlete plays in league ← athlete beat athlete → athlete plays sport	0.52	1.59	athlete plays for team → organization headquartered in state or province ← organization headquartered in state or province → team plays sport
Active rule #5	0.73	1.72	athlete plays in league → organization hired person ← athlete coach → athlete plays sport	0.69	1.56	athlete plays in league ← team plays in league ← team also known as → team plays sport
Logit bias		-3.16			-2.88	

Explanations produced by XKE and XKEe for the same triple using two different feature selection methods indicated in the **Feature Selection** field. In the bottom part we can see the top-5 most important explanations (active rules) for each model, with the corresponding *path context-relatedness (pc)*, and *logit weight (w)*. For each explanation, the preceding arrow → (or no arrow) indicates that the relation is walked forwards, whereas ← indicates the relation is walked in reverse. **[e]** indicates that the feature was not present in the original graph and was found in the embedding by XKEe.

Table 5 – Explanations by XKE and XKEe.

ID	**#5 Analogy / FB15K-237**			**#6 Analogy / FB15K-237**		
Head	Stroke			Stroke		
Relation	people cause of death people			people cause of death people		
Tail	Robert F. McGowan			Robert F. McGowan		
True Label	0			0		
Embedding Prediction	0			0		
Feature Selection	top pc rels: 0.2			top pop: 0.2		
XKE Prediction	0			1		
XKEe Prediction	0			1		
	pc	**w**	**explanations**	**pc**	**w**	**explanations**
Active rule #1	0.86	-1.16	people cause of death people → people person sibling s people sibling relationship sibling ← people cause of death people → people cause of death people	0.56	-0.91	[e] people cause of death people → people person spouse s people marriage type of union → people marriage union type unions of this type people marriage location of ceremony → location location contains
Active rule #2	0.74	0.02	people cause of death people ← people cause of death people → medicine disease risk factors ← people person gender	0.46	-0.23	[e] people cause of death people → people person places lived people place lived location ← film film featured film locations ← film actor film film performance film
Active rule #3	0.84	0.15	people cause of death people → people deceased person place of death ← people deceased person place of death	0.42	-0.20	[e] people cause of death people ← education educational institution students graduates education education student → common topic webpage common webpage category ← common topic webpage common webpage category
Active rule #4	0.71	0.83	people cause of death people → people person gender ← people person gender	0.51	-0.18	[e] people cause of death people → people person place of birth → common topic webpage common webpage category ← common topic webpage common webpage category
Active rule #5	0.74	0.87	people cause of death people → people person profession ← people person profession	0.52	-0.17	[e] people cause of death people ← base cultural event event entity involved → military military conflict combatants military military combatant group combatants ← people person nationality
Logit bias		-2.66			-0.70	

Table 6 – Explanations by XKE and XKEe.

ID	**#7 TransE / NELL186**			**#8 TransE / NELL186**		
Head	city: newark			city: newark		
Relation	city lies on river			city lies on river		
Tail	river: hudson			river: hudson		
True Label	1			1		
Embedding Prediction	1			1		
Feature Selection	top pc rels: 0.2			top pop: 0.2		
XKE Prediction	0			1		
XKEe Prediction	1			1		
	pc	**w**	**explanations**	**pc**	**w**	**explanations**
Active rule #1	0.63	2.67	[e] at location ← city located in state → city lies on river	0.51	13.11	at location ← automobile maker car dealers in state or province → automobile maker dealers in city → city lies on river
Active rule #2	1.00	2.62	[e] city lies on river ← city lies on river → city lies on river	0.52	9.71	location located within location → state or province is bordered by state or province ← at location → city lies on river
Active rule #3	0.62	2.47	[e] location located within location ← city located in geopolitical location → city lies on river	0.60	9.71	location located within location → state or province is bordered by state or province ← city located in state → city lies on river
Active rule #4	0.65	2.39	[e] city located in country ← location located within location → city lies on river	0.54	4.88	location located within location ← at location → city lies on river
Active rule #5	0.71	2.23	[e] city also known as → city lies on river	0.49	3.70	at location → state or province is bordered by state or province ← sub part of → city lies on river
Logit bias		-0.54			-0.87	

Explanations produced by XKE and XKEe for the same triple using two different feature selection methods indicated in the **Feature Selection** field. In the bottom part we can see the top-5 most important explanations (active rules) for each model, with the corresponding *path context-relatedness (pc)*, and *logit weight (w)*. [e] indicates that the feature was not present in the original graph and was found in the embedding by XKEe.

Table 7 – Explanations by XKE and XKEe.

ID	**#9 Analogy / NELL186**			**#10 Analogy / NELL186**		
Head	country: united kingdom			country: united kingdom		
Relation	country currency			country currency		
Tail	currency: pounds			currency: pounds		
True Label	1			1		
Embedding Prediction	1			1		
Feature Selection	top pc rels: 0.2			no feature selection		
XKE Prediction	1			1		
XKEe Prediction	1			1		
	pc	**w**	**explanations**	**pc**	**w**	**explanations**
Active rule #1	0.59	3.87	← bank bank in country → bank bank in country → country also known as → country currency	0.69	0.96	← language of country → language of country → country currency
Active rule #2	0.69	2.80	← language of country → language of country → country currency	0.59	0.60	← country also known as ← bank bank in country → bank bank in country → country currency
Active rule #3	0.67	2.51	← bank bank in country → bank bank in country → country currency	0.62	0.57	← language of country → language of country ← country located in geopolitical location → country currency
Active rule #4	0.59	2.00	country also known as ← bank bank in country → bank bank in country → country currency	0.62	0.56	← agricultural product came from country → agricultural product came from country → country currency
Active rule #5	0.62	1.94	← language of country → language of country ← country located in geopolitical location → country currency	0.67	0.50	← bank bank in country → bank bank in country → country currency
Logit bias		-0.63			-2.10	

Explanations produced by XKE and XKEe for the same triple using two different feature selection methods indicated in the **Feature Selection** field. In the bottom part we can see the top-5 most important explanations (active rules) for each model, with the corresponding *path context-relatedness (pc)*, and *logit weight (w)*. [e] indicates that the feature was not present in the original graph and was found in the embedding by XKEe.

6 Conclusions and Future Work

This report contains novel approaches to enhance the performance and the quality of explanations produced by XKE, a seminal method that produces explanations for predictions made by embedding models. We provided the background knowledge required to understand the technique and some of the related works tackling the issue of interpretability in the context of machine learning. Then, we discussed some of the shortcomings of XKE: low fidelity and the absence of explanations for certain predictions, and the quality of the explanations. We addressed the first issue with the XKEe *(Enhanced XKE)*, a method that produces new paths in the latent space allowing the explainer model to correct wrong predictions, increasing the overall model fidelity. The second issue was addressed by C-XKE *(Context-aware XKE)*, a method relying on word embeddings that help to select relations and path types that are more context-related with the target relation increasing the interpretability of the explanations. With this method, we proposed a novel metric to objectively measure the interpretability of the explanations, allowing us to compare the quality of the explanations for different settings. Both approaches can be added to the toolbox of the machine learning practitioner to be used separately or together, and both work well with the TRUE and PRED variants of the XKE algorithm.

The novel approaches proposed in our work were strongly supported by experiments evaluating the performance of C-XKEe *(Context-aware enhanced XKE)* against benchmark datasets, demonstrating the increased fidelity and also qualitative analysis of several explanations, combined with the interpretability metric we have developed.

We believe our work is another step towards explainable AI and provides some base for further research and investigation. The field of embedding models is still growing, and interpretability will remain a major concern. In future work, we foresee the further development of interpretability metrics, helping to reduce its still subjective nature. Yet, since XKE helps to shed light on how embedding models are internally organized, further evaluation of XKE with different types of embedding models would also be necessary.

Bibliography

AKRAMI, Farahnaz; GUO, Lingbing; HU, Wei; LI, Chengkai. Re-evaluating Embedding-Based Knowledge Graph Completion Methods. ACM, 2018. Cited in page 57.

ANDREWS, Robert; DIEDERICH, Joachim; TICKLE, Alan B. Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-Based Systems*, p. 373–389, 1995. Cited in page 40.

BIANCHI, Federico; ROSSIELLO, Gaetano; COSTABELLO, Luca; PALMONARI, Matteo; MINERVINI, Pasquale. Knowledge Graph Embeddings and Explainable AI. apr 2020. Cited in page 43.

BIRAN, Or; COTTON, Courtenay. Explanation and Justification in Machine Learning: A Survey. In: *IJCAI 2017 Workshop on Explainable Artificial Intelligence (XAI)*. [S.l.: s.n.], 2017. p. 8–13. Cited 2 times in pages 37 and 43.

BOLLACKER, Kurt; EVANS, Colin; PARITOSH, Praveen; STURGE, Tim; TAYLOR, Jamie. Freebase: A Collaboratively Created Graph Database For Structuring Human Knowledge. In: *ACM SIGMOD International Conference on Management of Data*. [S.l.: s.n.], 2008. p. 1247–1250. Cited 2 times in pages 17 and 57.

BORDES, Antoine; USUNIER, Nicolas; GARCÍA-DURÁN, Alberto; WESTON, Jason. Translating Embeddings for Modeling Multi-Relational Data. *Advances in Neural Information Processing Systems*, p. 2787–2795, 2013. Cited 5 times in pages 23, 31, 32, 36, and 57.

CHANDRAHAS; SENGUPTA, Tathagata; PRAGADEESH, Cibi; TALUKDAR, Partha. Inducing Interpretability in Knowledge Graph Embeddings. In: . [S.l.: s.n.], 2017. Cited in page 44.

DEMIR, Caglar; NGOMO, Axel Cyrille Ngonga. Convolutional Complex Knowledge Graph Embeddings. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. [S.l.]: Springer Science and Business Media Deutschland GmbH, 2020. v. 12731 LNCS, p. 409–424. ISBN 9783030773847. ISSN 16113349. Cited in page 32.

DETTMERS, Tim; MINERVINI, Pasquale; STENETORP, Pontus; RIEDEL, Sebastian. Convolutional 2D Knowledge Graph Embeddings. In: *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*. [S.l.]: AAAI press, 2017. p. 1811–1818. ISBN 9781577358008. Cited in page 32.

DORAN, Derek; SCHULZ, Sarah; BESOLD, Tarek R. What Does Explainable AI Really Mean? A New Conceptualization of Perspectives. In: *CEUR Workshop Proceedings*. [S.l.: s.n.], 2017. Cited 2 times in pages 18 and 38.

EBISU, Takuma; ICHISE, Ryutaro. TorusE: Knowledge Graph Embedding on a Lie Group. In: *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*. [S.l.]: AAAI press, 2017. p. 1819–1826. ISBN 9781577358008. Cited in page 32.

GARDNER, Matt; MITCHELL, Tom. Efficient and Expressive Knowledge Base Completion Using Subgraph Feature Extraction. In: *Proceedings of the 2015 Conference on Empirical*

Methods in Natural Language Processing. [S.l.]: Association for Computational Linguistics, 2015. p. 1488–1498. Cited 6 times in pages 26, 28, 45, 48, 49, and 58.

GARDNER, Matthew; MITCHELL, Tom; COHEN, William; FALOUTSOS, Christos; BORDES, Antoine. *Reading and Reasoning with Knowledge Graphs*. Tese (Doutorado), 2015. Cited in page 60.

GARDNER, Matt; TALUKDAR, Partha; KRISHNAMURTHY, Jayant; MITCHELL, Tom. Incorporating Vector Space Similarity in Random Walk Inference over Knowledge Bases. In: *Proceedings of EMNLP. Association for Computational Linguistics*. [S.l.: s.n.], 2014. Cited in page 53.

GARDNER, Matt; TALUKDAR, Partha; MITCHELL, Tom. Combining vector space embeddings with symbolic logical inference over open-domain text. *AAAI spring symposium series*, v. 6, p. 1, 2015. Cited in page 53.

GETOOR, Lise; TASKAR, Ben. *Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning)*. [S.l.]: The MIT Press, 2007. ISBN 0262072882. Cited 2 times in pages 17 and 24.

GUNNING, D. *Broad Agency Announcement Explainable Artificial Intelligence (XAI)*. [S.l.], 2016. DARPA–BAA–16–53 p. Cited in page 18.

GUO, Lingbing; SUN, Zequn; HU, Wei. Learning to Exploit Long-term Relational Dependencies in Knowledge Graphs. In: *36th International Conference on Machine Learning, ICML 2019*. [S.l.]: International Machine Learning Society (IMLS), 2019. v. 2019-June, p. 4441–4450. ISBN 9781510886988. Cited in page 32.

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*. [S.l.: s.n.], 2015. p. 84–94. Cited in page 57.

GUSMÃO, Arthur C.; CORREIA, Alvaro C.; De Bona, Glauber; COZMAN, Fabio G. Interpreting Embedding Models of Knowledge Bases : A Pedagogical Approach. In: *2018 ICML Workshop on Human Interpretability in Machine Learning (WHI 2018)*. [S.l.: s.n.], 2018. p. 79–86. Cited 5 times in pages 18, 41, 42, 43, and 59.

HAN, Xu; CAO, Shulin; LV, Xin; LIN, Yankai; LIU, Zhiyuan; SUN, Maosong; LI, Juanzi. *OpenKE: An Open Toolkit for Knowledge Embedding*. [S.l.], 2018. 139–144 p. Cited in page 58.

JI, Guoliang; HE, Shizhu; XU, Liheng; LIU, Kang; ZHAO, Jun. Knowledge Graph Embedding via Dynamic Mapping Matrix. *ACL-IJCNLP 2015 - 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, Proceedings of the Conference*, Association for Computational Linguistics (ACL), v. 1, p. 687–696, 2015. Cited in page 32.

JIANG, Xiaotian; WANG, Quan; WANG, Bin. Adaptive Convolution for Multi-Relational Learning. In: *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*. [S.l.]: Association for Computational Linguistics (ACL), 2019. v. 1, p. 978–987. ISBN 9781950737130. Cited in page 32.

KAZEMI, Seyed Mehran; POOLE, David. SimplE Embedding for Link Prediction in Knowledge Graphs. In: *Advances in Neural Information Processing Systems 31*. [S.l.: s.n.], 2018. p. 4284–4295. Cited 2 times in pages 31 and 44.

LAO, Ni; COHEN, William. Relational Retrieval Using a Combination of Path-Constrained Random Walks. *Machine Learning*, v. 81, n. n.1, p. 53–67, 2010. Cited in page 25.

LAO, Ni; MITCHELL, Tom; COHEN, William. Random Walk Inference and Learning in A Large Scale Knowledge Base. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Edinburgh, United Kingdom: Association for Computational Linguistics, 2011. p. 529—-539. Cited 2 times in pages 26 and 28.

LI, Jun; HOU, Jie; ZHOU, Chunyu. An Improved Capsule Network-based Embedding Model for Knowledge Graph Completion. In: *Proceedings of the 33rd Chinese Control and Decision Conference, CCDC 2021*. [S.l.]: Institute of Electrical and Electronics Engineers Inc., 2021. p. 2247–2251. ISBN 9781665440899. Cited in page 32.

LIN, Yankai; LIU, Zhiyuan; SUN, Maosong; LIU, Yang; ZHU, Xuan. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In: *29th AAAI Conference on Artifitial Intelligence*. [S.l.: s.n.], 2015. p. 2181–2187. Cited in page 32.

LIPTON, Zachary C. The Mythos of Model Interpretability. In: *ICML Workshop on Human Interpretability in Machine Learning*. [S.l.: s.n.], 2016. p. 96–100. Cited in page 37.

LIU, Hanxiao; WU, Yuexin; YANG, Yiming. Analogical Inference for Multi-relational Embeddings. 2017. Cited 2 times in pages 31 and 36.

LUNDBERG, Scott M; ALLEN, Paul G; LEE, Su-In. A Unified Approach to Interpreting Model Predictions. In: *Advances in Neural Information Processing Systems*. [S.l.: s.n.], 2017. p. 4765–4774. Cited in page 18.

MAZUMDER, Sahisnu; LIU, Bing. Context-aware Path Ranking for Knowledge Base Completion. In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17) Context-aware*. [S.l.: s.n.], 2017. p. 1195–1201. Cited 5 times in pages 28, 29, 48, 49, and 52.

MIKOLOV, Tomas; CHEN, Kai; CORRADO, Greg; DEAN, Jeffrey. Distributed Representations of Words and Phrases and their Compositionality. In: *NIPS'13: Proceedings of the 26th International Conference on Neural Information Processing Systems*. [S.l.: s.n.], 2013. p. 3111–3119. Cited in page 29.

MIKOLOV, Tomas; CHEN, Kai; CORRADO, Greg; DEAN, Jeffrey. Efficient Estimation of Word Representations in Vector Space. *CoRR*, abs/1301.3, p. 1–12, 2013. Cited 2 times in pages 29 and 36.

MILLER, Tim. Explanation in Artificial Intelligence: Insights from the Social Sciences. *Artificial Intelligence*, v. 267, 2017. Cited in page 37.

MITCHELL, Tom; COHEN, William; HRUSCHKA, E; TALUKDAR, Partha; YANG, Bishan; BETTERIDGE, Justin; CARLSON, Andrew; DALVI, B; GARDNER, Matt; KISIEL, Bryan; KRISHNAMURTHY, J; LAO, Ni; MAZAITIS, K; MOHAMED, T; NAKASHOLE, N; PLATANIOS, E; RITTER, A; SAMADI, M; SETTLES, B; WANG, R; WIJAYA, D; GUPTA, A; CHEN, X; SAPAROV, A; GREAVES, M; WELLING, J. Never-Ending Learning. *Communications of the Acm*, v. 61, n. 1, p. 2302–2310, 2015. Cited 2 times in pages 17 and 57.

MITTELSTADT, Brent; RUSSELL, Chris; WACHTER, Sandra. Explaining Explanations in AI. In: *FAT '19: Proceedings of the Conference on Fairness, Accountability, and Transparency*. [S.l.: s.n.], 2019. p. 279–288. ISBN 9781450361255. Cited in page 37.

MOHAMED, Sameh K; NOVÁEK, Vít; VANDENBUSSCHE, Pierre-Yves. Knowledge Base Completion Using Distinct Subgraph Paths. In: *33rd Annual ACM symposium on Applied Computing*. [S.l.: s.n.], 2018. ISBN 9781450351911. Cited in page 45.

NEIL, Daniel; BRIODY, Joss; LACOSTE, Alix; SIM, Aaron; CREED, Paidi; SAFFARI, Amir. Interpretable Graph Convolutional Neural Networks for Inference on Noisy Knowledge Graphs. In: *NeurIPS 2018 - Machine Learning for Health (ML4H)*. [S.l.: s.n.], 2018. Cited in page 44.

NGUYEN, Dat Quoc. An overview of embedding models of entities and relationships for knowledge base completion. *CoRR*, abs/1703.0, 2017. Cited in page 30.

NGUYEN, Dai Quoc; NGUYEN, Tu Dinh; NGUYEN, Dat Quoc; PHUNG, Dinh. A Novel Embedding Model for Knowledge Base Completion Based on Convolutional Neural Network. In: *NAACL HLT 2018 - 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*. [S.l.]: Association for Computational Linguistics (ACL), 2017. v. 2, p. 327–333. Cited in page 32.

NGUYEN, Dat Quoc; SIRTS, Kairit; QU, Lizhen; JOHNSON, Mark. Neighborhood Mixture Model for Knowledge Base Completion. In: *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*. Berlin, Germany: [s.n.], 2016. p. 40–50. Cited in page 58.

NGUYEN, Dat Quoc; SIRTS, Kairit; QU, Lizhen; JOHNSON, Mark. STransE: a novel embedding model of entities and relationships in knowledge bases. In: *2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2016 - Proceedings of the Conference*. [S.l.]: Association for Computational Linguistics (ACL), 2016. p. 460–466. ISBN 9781941643914. Cited in page 32.

NGUYEN, Dai Quoc; VU, Thanh; NGUYEN, Tu Dinh; NGUYEN, Dat Quoc; PHUNG, Dinh. A Capsule Network-based Embedding Model for Knowledge Graph Completion and Search Personalization. In: *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*. [S.l.]: Association for Computational Linguistics (ACL), 2018. v. 1, p. 2180–2189. ISBN 9781950737130. Cited in page 32.

NICKEL, Maximilian; MURPHY, Kevin; TRESP, Volker; GABRILOVICH, Evgeniy. A Review of Relational Machine Learning for Knowledge Graphs. *IEEE*, v. 104, n., p. 11–33, 2015. Cited 7 times in pages 17, 21, 22, 23, 24, 25, and 30.

NICKEL, Maximilian; ROSASCO, Lorenzo; POGGIO, Tomaso. Holographic Embeddings of Knowledge Graphs. *30th AAAI Conference on Artificial Intelligence, AAAI 2016*, AAAI press, p. 1955–1961, oct 2015. Cited in page 31.

NICKEL, Maximilian; TRESP, Volker; KRIEGEL, Hans-Peter. A Three-Way Model for Collective Learning on Multi-Relational Data. In: *ICML'11: Proceedings of the 28th International Conference on International Conference on Machine Learning*. [S.l.: s.n.], 2011. p. 809–816. Cited in page 31.

PEZESHKPOUR, Pouya; TIAN, Yifan; SINGH, Sameer. Investigating Robustness and Interpretability of Link Prediction via Adversarial Modifications. In: *NAACL HLT 2019 - 2019*

Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference. [S.l.: s.n.], 2019. Cited in page 45.

POLLETI, Gustavo Padilha; COZMAN, Fabio Gagliardi. Faithfully Explaining Predictions of Knowledge Embeddings. In: *Brazilian Conference on Inteligent Systems*. [S.l.: s.n.], 2019. Cited in page 44.

RIBEIRO, Marco Tulio; SINGH, Sameer; GUESTRIN, Carlos. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In: *22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. [S.l.: s.n.], 2016. p. 1135–1144. Cited 5 times in pages 18, 38, 39, 40, and 45.

ROBNIK-ŠIKONJA, Marko; BOHANEC, Marko. Perturbation-Based Explanations of Prediction Models. *Human and Machine Learning*, p. 159–175, 2018. Cited in page 47.

ROSSI, Andrea; BARBOSA, Denilson; FIRMANI, Donatella; MATINATA, Antonio; MERIALDO, Paolo. Knowledge Graph Embedding for Link Prediction. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, ACM PUB27 New York, NY, USA, v. 15, n. 2, jan 2021. ISSN 1556472X. Cited in page 31.

RUSCHEL, Andrey; GUSMÃO, Arthur C.; POLLETI, Gustavo Padilha; COZMAN, Fabio Gagliardi. Explaining Completions Produced by Embeddings of Knowledge Graphs. In: KERN-ISBERNER, Gabriele; OGNJANOVIĆ, Zoran (Ed.). *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*. [S.l.]: Springer International Publishing, 2019. p. 324–335. ISBN 978-3-030-29765-7. Cited 3 times in pages 48, 55, and 58.

SOCHER, Richard; CHEN, Danqi; MANNING, Christopher D; NG, Andrew Y. Reasoning With Neural Tensor Networks for Knowledge Base Completion. In: *Neural Information Processing Systems (2003)*. [S.l.: s.n.], 2013. p. 926–934. Cited in page 23.

STADELMAIER, Josua; PADÓ, Sebastian. Modeling Paths for Explainable Knowledge Base Completion. In: *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. [S.l.: s.n.], 2019. p. 147–157. Cited in page 45.

SUCHANEK, Fabian M.; KASNECI, Gjergji; WEIKUM, Gerhard. YAGO: A Core of Semantic Knowledge Unifying WordNet and Wikipedia. *WWW 2007*, 2007. Cited in page 17.

SUN, Zhiqing; DENG, Zhi Hong; NIE, Jian Yun; TANG, Jian. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. In: *7th International Conference on Learning Representations, ICLR 2019*. [S.l.]: International Conference on Learning Representations, ICLR, 2019. Cited in page 32.

TAKAHASHI, Ryo; TIAN, Ran; INUI, Kentaro. Interpretable and Compositional Relation Learning by Joint Training with an Autoencoder. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. [S.l.: s.n.], 2018. p. 2148–2159. ISBN 1805.09547v1. Cited in page 44.

TOUTANOVA, Kristina; CHEN, Danqi. Observed versus latent features for knowledge base and text inference. *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, p. 57–66, 2015. Cited 2 times in pages 28 and 57.

TROUILLON, Théo; WELBL, Johannes; Sebastian Riedel, Csuclacuk; Eric Gaussier, Csuclac U; Guillaume Bouchard, Imagfr. Complex Embeddings for Simple Link Prediction. In: *33rd International Conference on Machine Learning*. [S.l.: s.n.], 2016. Cited in page 31.

W3C. *RDF 1.1 Concepts and Abstract Syntax*. 2014. Disponível em: <https://www.w3.org/RDF/>. Cited in page 21.

WANG, Quan; MAO, Zhendong; WANG, Bin; GUO, Li. Knowledge Graph Embedding : A Survey of Approaches and Applications. *IEEE Transactions on Knowledge and Data Engineering*, v. 29, n. 12, p. 2724–2743, 2017. Cited 3 times in pages 17, 30, and 31.

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. p. 1112, 1119. Cited in page 32.

WEST, Robert; GABRILOVICH, Evgeniy; MURPHY, Kevin; SUN, Shaohua; GUPTA, Rahul; LIN, Dekang. Knowledge base completion via search-based question answering. In: *Proceedings of the 23rd international conference on World wide web - WWW '14*. [S.l.: s.n.], 2014. ISBN 9781450327442. ISSN 21508097. Cited in page 23.

XIAO, Han; HUANG, Minlie; ZHU, Xiaoyan. From One Point to A Manifold: Knowledge Graph Embedding For Precise Link Prediction. In: *IJCAI'16: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. [S.l.: s.n.], 2016. p. 1315–1321. Cited in page 32.

XIAO, Han; HUANG, Minlie; ZHU, Xiaoyan. Knowledge Semantic Representation: A Generative Model for Interpretable Knowledge Graph Embedding. In: . [S.l.: s.n.], 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. p. 950–962. Cited in page 44.

YANG, Bishan; YIH, Wen-Tau; HE, Xiaodong; GAO, Jianfeng; DENG, Li. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In: *ICLR 2015 3rd International Conference on Learning Representations*. [S.l.: s.n.], 2015. Cited in page 31.

YOGENDRAN, Nivetha; KANAGARAJAH, Abivarshi; CHANDIRAN, Kularajini; THAYASIVAM, Uthayasanker. Hybrid Approach for Accurate and Interpretable Representation Learning of Knowledge Graph. In: *MERCon 2020 - 6th International Multidisciplinary Moratuwa Engineering Research Conference, Proceedings*. [S.l.]: Institute of Electrical and Electronics Engineers Inc., 2020. p. 1–6. ISBN 9781728190594. Cited in page 45.

ZHANG, Wen; DENG, Shumin; WANG, Han; CHEN, Qiang; ZHANG, Wei; CHEN, Huajun. XTransE: Explainable knowledge graph embedding for link prediction with lifestyles in e-Commerce. *Communications in Computer and Information Science*, Springer, v. 1157 CCIS, p. 78–87, 2020. ISSN 18650937. Cited in page 45.

ZHANG, Wen; PAUDEL, Bibek; ZHANG, Wei; BERNSTEIN, Abraham; CHEN, Huajun. Interaction Embeddings for Prediction and Explanation in Knowledge Graphs. In: . [S.l.: s.n.], 2019. ISBN 9781450359405. Cited 3 times in pages 32, 44, and 54.