Explanation of Link Predictions on Knowledge Graphs via Levelwise Filtering and Graph Summarization

Roberto Barile$^{1[0009-0007-3058-8692]}$, Claudia d'Amato$^{1,2[0000-0002-3385-987X]}$, and Nicola Fanizzi$^{1,2[0000-0001-5319-7933]}$

^{1} Dipartimento di Informatica – University of Bari Aldo Moro, Italy
{roberto.barile, claudia.damato, nicola.fanizzi}@uniba.it
^{2} CILA – University of Bari Aldo Moro, Italy

Abstract. Link Prediction methods aim at predicting missing facts in Knowledge Graphs (KGs) as they are inherently incomplete. Several methods rely on Knowledge Graph Embeddings, which are numerical representations of elements in the Knowledge Graph. Embeddings are effective and scalable for large KGs; however, they lack explainability. KELPIE is a recent and versatile framework that provides post-hoc explanations for predictions based on embeddings by revealing the facts that enabled them. Problems have been recognized, however, with filtering potential explanations and dealing with an overload of candidates. We aim at enhancing KELPIE by targeting three goals: reducing the number of candidates, producing explanations at different levels of detail, and improving the effectiveness of the explanations. To accomplish them, we adopt a semantic similarity measure to enhance the filtering of potential explanations, and we focus on a condensed representation of the search space in the form of a quotient graph based on entity types. Three quotient formulations of different granularity are considered to reduce the risk of losing valuable information. We conduct a quantitative and qualitative experimental evaluation of the proposed solutions, using KELPIE as a baseline.

Keywords: Knowledge Graphs · Link Prediction · Explanation

1 Introduction

Knowledge Graphs (KGs) emerged as a tool to represent, navigate and query the growing flood of data by encapsulating knowledge of complex domains into a form accessible to both humans and machines. A KG is a multi-relational graph composed of entities and relations, represented as nodes and edges, respectively. KGs are often integrated with ontologies, that formally define classes and relations, which allow for advanced inference capabilities [18]. Several examples of large KGs exist, including enterprise products, e.g., see [30,13], and various well-known open sources, e.g., [5,1,23].

Despite their effectiveness, working with KGs often suffers from their inherent incompleteness [18], due also to the open-world semantics generally assumed in scenarios that involve them as the result of a complex, incremental and distributed building process. This underpins the importance of tasks like *Link Prediction* (LP) and *triple classification* aiming, respectively, at inferring missing relationships between existing nodes and deciding on the truth of (new) triples.

Among the many LP methods grounded on *Machine Learning* (ML) models, those based on *Knowledge Graph Embeddings* (KGEs) have emerged as a prevalent approach, especially for their superior scalability (see [25] for a recent survey). KGEs map elements in the KGs to low-dimensional vector spaces, streamlining complex tasks via linear algebra.

Nevertheless, such models tend to operate as "opaque boxes" whose predictions are difficult to explain, thus undermining their credibility and trust. Indeed, this opacity can be particularly problematic in contexts where understanding the reasons supporting the predictions is critical, such as healthcare or financial decision-making. For example, LP might help to find out potential connections between specific drugs and their side effects [9]. In such cases, not only the predictions but also their underlying rationale is essential, as these may influence decisions about investments on a drug.

To address this opacity, the field of *Explainable Artificial Intelligence* (XAI) [21] has come into the spotlight. XAI aims at making the decisions of ML models more transparent and understandable, enhancing human trust and comprehension across all AI applications. Following [17], XAI methods can be divided into two categories: a) *post-hoc* methods, for addressing specific model outcomes; b) *global interpretability* methods, for offering a holistic understanding of the entire inference process.

We will focus on *post-hoc* methods, which are suitable for scalable LP models based on numerical representations. In the post-hoc setting, given a prediction, the generated explanation typically comprises a specific set of facts that have enabled the inference through the model. A recent, effective, and versatile framework providing such kind of explanation to LP tasks is KELPIE [26]. Specifically, KELPIE provides explanations through three steps: (1) the pre-filter/extraction of a sub-graph designated as the search space, (2) the combination of facts within this sub-graph into candidate explanations, (3) the ranking of such possible explanations. Nevertheless, the pre-filter phase is grounded on the exploitation of a topological measure that could lead to discarding facts that can make up potentially valid explanations. Furthermore, the successive combination of facts for building candidate explanations leads to an extremely large number of possible explanations. Even more so, KELPIE is not able to offer explanations with an adaptable level of detail. Some users might prefer a brief, high-level explanation, while others might require a more detailed account.

Recognizing these limitations, we identified three specific research goals. Firstly, adopting an alternative metric for the pre-filtering phase. Second, decreasing the number of assessments of candidate explanations necessary to identify the optimal ones. Third, to be able to generate explanations at different levels of detail. Overall, as a final goal, we aim at improving the results for the metrics of end-to-end effectiveness.

In agreement with these goals, we formulate our contributions relying on formal ontologies often available in KGs. Firstly, we propose to adopt in the pre-filter/extraction phase a measure of semantic similarity between entities so to focus on facts semantically related to the prediction, thus potentially more suitable as part of an explanation. Secondly, our proposal is to compute a summarization of the search space using different formulations of quotient graphs so to speed up the search, ground it on clusters of semantically related facts and offering explanations at different levels of granularity. We also experimentally prove that these contributions are actually able to improve effectiveness and efficiency.

The rest of this work is organized as follows. Sect. 2 introduces basic notions that are essential for the paper. Sect. 3 reviews existing approaches for explaining link predictions. Sect. 4 provides details on KELPIE, then outlines the proposed contributions. The experimental evaluation of the resulting approach is illustrated in Sect. 5. Finally, Sect. 6 summarizes achievements and limitations, delineating future works.

2 Fundamentals

2.1 Knowledge Graphs and Embedding Models

A KG is a graph-based data structure $\mathcal{G}(\mathcal{V}, \mathcal{E})$ with $\mathcal{V}$ representing a set of nodes, also known as entities, and $\mathcal{E}$ representing a set of edges, labeled with relationships which connect pairs of entities.

In the adopted RDF model, a KG is a collection of triples in the format $\langle s, p, o\rangle$, i.e., *subject*, *predicate*, and *object* where $s, o \in \mathcal{V}$ and $p \in \mathcal{E}$. In RDF, the terms are denoted by the elements of the sets $\mathcal{U}$ (URIs), $\mathcal{B}$ (blank nodes) and $\mathcal{L}$ (literals). Consequently, an RDF graph is a set of triples with: $s \in \mathcal{U} \cup \mathcal{B}$, $r \in \mathcal{U}$, and $o \in \mathcal{U} \cup \mathcal{B} \cup \mathcal{L}$.

Various models have been proposed for representing KGs in low-dimensional vector spaces, by learning a unique distributed representation (or *embedding*) for each entity and predicate in the KG and considering different representation spaces (e.g., point-wise, complex, discrete, Gaussian, manifold). Here we focus on vector embeddings in the set of real numbers.

In an ML setting, a KG $\mathcal{G}$ may be further split into a training set $\mathcal{G}_{train}$, a validation set $\mathcal{G}_{val}$ and a test set of triples $\mathcal{G}_{test}$. Irrespective of the specific learning approach, these models all represent each entity $x \in \mathcal{V}$ by means of a continuous embedding vector $\mathbf{e}_x \in \mathbb{R}^k$, where $k \in \mathbb{N}$ is a user-defined hyper-parameter. Similarly, each predicate $p \in \mathcal{E}$ is associated to a scoring function $f_p : \mathbb{R}^k \times \mathbb{R}^k \to \mathbb{R}$. For each pair of entities $s, o \in \mathcal{V}$, the score $f_p(\mathbf{e}_s, \mathbf{e}_o)$ is a measure of the plausibility of the statement encoded by $\langle s, p, o\rangle$. The embedding of all entities (and predicates) in $\mathcal{G}$ is learned by minimizing a loss function, often a margin-based one.

2.2 Quotient Graph

In the context of KGs, quotient graphs aim at summarizing the data graph into a higher-level topology [8]. They are based on the concept of a quotient set and equivalence classes. Given a set X and an equivalence relation $\sim$ on it, X is partitioned into disjoint subsets of equivalent elements, the equivalence classes. The *quotient set* $X/_\sim$ contains all equivalence classes.

Before diving into the specifics of the quotient graph formation, we introduce the notions of simulation and bisimulation, and we report the definitions as provided in [18]. A simulation is a binary relation from a graph $\mathcal{G}$ to a graph $\mathcal{G}'$ that maintains the existence of a path between connected nodes in $\mathcal{G}$ to $\mathcal{G}'$. Formally:

Definition 1. *A simulation from a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ to a graph $\mathcal{G}'(\mathcal{V}', \mathcal{E}')$ is a relation $R \subseteq \mathcal{V} \times \mathcal{V}'$ such that for every edge $\langle x, y, z \rangle$ in $\mathcal{G}$, if $x \; R \; x'$ for some $x' \in \mathcal{V}'$, then there exists $z' \in \mathcal{V}'$ such that $z \; R \; z'$ and $\langle x', y, z' \rangle$ is an edge in $\mathcal{G}'$. $\mathcal{G}'$ simulates $\mathcal{G}$ when a simulation exists from $\mathcal{G}$ to $\mathcal{G}'$.*

A stronger, symmetric extension of this relation can be defined as follows:

Definition 2. *A bisimulation between $\mathcal{G}$ and $\mathcal{G}'$ is a relation that is both a simulation from $\mathcal{G}$ to $\mathcal{G}'$ and a simulation from $\mathcal{G}'$ to $\mathcal{G}$. When a bisimulation exists on $\mathcal{G}$ and $\mathcal{G}'$, they are bisimilar.*

Given a KG $\mathcal{G}(\mathcal{V}, \mathcal{E})$, its *quotient graph* $\mathcal{Q}(\mathcal{V}/_\sim, \mathcal{E}/_\sim)$ is a graph with node set $\mathcal{V}/_\sim$ as the quotient set of $\mathcal{V}$ according to the equivalence relation of choice, and edge set $\mathcal{E}/_\sim$, that can be defined in different ways, depending on the desired level of preservation of the structure. In the case of simulation, an edge $\langle X, y, Z \rangle$ in the quotient graph exists if and only if there exists $x \in X$ and $z \in Z$ such that an edge $\langle x, y, z \rangle$ is in the original graph $\mathcal{G}$. Bisimulation, on the other hand, imposes a stronger condition: $\langle X, y, Z \rangle$ is an edge in the quotient graph when, for each $x \in X$, there exists a $z \in Z$ such that $\langle x, y, z \rangle$ is in the input graph.

Bisimulation also involves refining the quotient nodes to split the ones that contain nodes with different outgoing edges in the original graph. To clarify, two nodes belong to the same quotient node if they share identical outgoing edges in the original graph and are in the same equivalence class. This is equivalent to finding the *Relational Stable Coarsest Partition* (RSCP) [15].

To summarize, the steps to compute a quotient graph *bisimilar* to $\mathcal{G}$ are the following: 1) compute $\mathcal{V}/_\sim$ as seen before; 2) compute the RSCP to refine $\mathcal{V}/_\sim$; 3) add the edges to $\mathcal{E}/_\sim$. If an equivalence relation is not available, the RSCP can be computed directly on $\mathcal{V}$.

Several algorithms to compute RSCP are available [22,14], however, such implementations are not suitable for *multi-graphs*, i.e., graphs that allow multiple parallel edges between the same pair of nodes, and therefore KGs. Indeed, these algorithms take into account only one of the multiple edges between the same pair of nodes, since all edges are assumed to be of the same type. To address the issue, the input graph can be pre-processed as formalized in [7]. Specifically, each triple $\langle s, p, o \rangle$ is represented as an unlabeled edge connecting s to the node (p, o).

3 Related Works

In this section, we specifically focus on post-hoc methods for explaining LP. Recent solutions are grounded on the exploitation of a data poisoning technique [33] in order to determine a single fact that, if inserted or eliminated, would most significantly poison the prediction. Another example is CRIAGE [24], which addresses the computational challenges of earlier models [19], but lacks clarity on the model adaptability and focuses on limited cases (solely those facts where the object is either the subject or the object of the prediction).

An objective that is receiving increasing attention and that is targeted in this paper is the development of model-independent explainability solutions. CROSSE [35] and APPROXSEMANTICCROSSE [10] analyze the KG topology instead of the model's behavior and find the paths from the subject to the object supported by analogous situations in the graph, i.e., with similar paths connecting similar entities. The LP model is used only to exclude relations and entities.

KELPIE [26] explains a prediction by returning a specific set of relevant facts rather than a single fact. To evaluate the relevance of a candidate explanation, KELPIE employs a novel *post-training* process which can be tailored to different KGE models. Similarly to KELPIE, KGEX [2] provide sets of facts as explanations, but employs sub-graph sampling and *Knowledge Distillation* of surrogate models. A complementary approach is proposed in [3], it provides explanation through abductive reasoning on a logical theory learned through a symbolic rule learning approach.

KELPIE represents a very significant advance in the class of model-independent explanation solutions. Nevertheless, it has some drawbacks: (i) in the initial filtering of facts it employs a topological measure which may be a limited insight on the correlation between the training facts and the prediction to explain, (ii) the combination of facts for building candidate explanations leads to a very large number of possible explanations, (iii) the explanations are solely provided as sets of facts, thus not encompassing more general insights such as the classes of the entities. This paper intends to overcome all these limitations.

In parallel to post-hoc explanation solutions, also inherently interpretable LP models are available as for the case of XTRANSE [34], the work in [4], and GNNEXPLAINER [32] targeting specifically *Graph Neural Networks*. These methods aim at making the model itself interpretable, whereas our goal is to develop a model-agnostic post-hoc solution for explaining LP results.

Also, general-purpose XAI methods have been developed, i.e., solutions that are independent of the task to explain. Such general-purpose approaches can be distinguished between those that identify relevant features of the input data as explanations, called *saliency explanations*, and those that identify training samples as explanations. Saliency-based frameworks (like SHAP [20] grounded on *Shapley values* [28]) are rather popular, but they are not easily adaptable to the LP task since they require interpretable input attributes, which are suitable for images or text, but fall short for LP using graph embeddings, where input samples are numerical. Indeed, in this case, saliency-based methods merely highlight the most significant components of vectors in relation to the outcome, lacking human interpretability. As for the methods identifying training samples as explanations [19], the notion of *Influence Functions* from robust statistics [27] has been exploited, resulting in a very high resource demanding solution.

4 The Proposed Approach

We propose a method for providing post-hoc explanations for LP on KGs exploiting their semantics. Specifically, moving from KELPIE, we aim at the enhancement of its main components. Our contributions capitalize on the exploitation of schema-level information from the shared OWL ontologies adopted by the KGs. The resulting new method remains independent of the KGE model. In the following, we summarize KELPIE in Sect. 4.1, then we delineate the proposed extensions in Sects. 4.2, 4.3.

4.1 KELPIE

KELPIE [26] stands out in the related works overview as it provides effective explanations including multiple triples, it adapts to any KGE model, and it is supplied with the resources to replicate the experiments. Therefore, we adopt it as the ground of our approach. We now delve into details on how it works. Given a predicted triple $\langle s, p, o\rangle$, a KELPIE explanation consists in a set of training triples featuring s that have enabled to predict the object o through the model. KELPIE, is structured around three main components:

- The Pre-Filter selects the most useful triples featuring s
- The Explanation Builder combines the pre-filtered triples into candidate explanations and identifies the most relevant ones.
- The Relevance Engine computes the relevance of a candidate explanation adopting a ML technique called *post-training*, coined by the authors.

We further detail each component individually.

The Pre-Filter aims at decreasing the complexity for the subsequent stages. Firstly, it extracts the sub-graph $\mathcal{G}^s_{train}$ of all the training triples featuring s as subject or as object. Then, it filters this sub-graph to obtain $\mathcal{F}^s_{train}$ by selecting the top-k most useful triples. The utility measure is based on graph topology. Specifically, for any given $\langle s, q, r\rangle$ (or $\langle r, q, s\rangle$), it computes the length of the shortest path connecting r to the predicted object o, ignoring the direction of the edges.

The Explanation Builder's task is to find a set of triples in $\mathcal{F}^s_{train}$ representing an optimal explanation X^* according to their relevance. It combines the triples in $\mathcal{F}^s_{train}$ into candidate explanations (each denoted as X), and determines whether any X can be accepted as X^* based on its relevance. Identifying X^* is a search problem within a space ($\mathcal{S}$) of candidate explanations of varying lengths, but exhaustive search is impractical. Indeed, Kelpie implements heuristic conditions to prune the search space.

The process is summarized in Algorithm 1. The method first computes the relevance of each triple in $\mathcal{F}^s_{train}$ used as a 1-triple explanation (Line 1). Before exploring any $\mathcal{S}_i$ with, $i > 1$ the algorithm computes the *preliminary relevance* of each explanation as the average relevance of its triples (Line 4). The subset $\mathcal{S}_i$ is then traversed in descending order of preliminary relevance. For each explanation, the Relevance Engine computes its *true relevance* (Line 8) and the algorithm checks if it meets the acceptance criteria (Line 9). The decision on whether to continue exploring the current subset or to move on to the next one is guided by ρ_i, which is defined as the ratio of the relevance of the current explanation to the highest relevance found so far in the subset (Line 14). If ρ_i becomes too small, indicating likely less relevant future explanations, the algorithm decides whether to move on to the next subset, with probability $1 - \rho_i$ (Lines 15-16).

The Relevance Engine adopts two alternative, yet complementary methods, namely *necessary relevance* and *sufficient relevance*. Both methods ground on a 4 steps approach: (a) create s' as a duplicate of s, (b) compute the set $\mathcal{F}^{s'}_{train}$ as $\mathcal{F}^s_{train} - X$ for *necessary relevance* and as $\mathcal{F}^s_{train} \cup X$ for *sufficient relevance*, (c) learn the entity embedding of s' through *post-training*, (d) compute the difference between the scores of the triples $\langle s, p, o\rangle$ and $\langle s', p, o\rangle$. Moreover, for *sufficient relevance*, instead of being applied on s, the process is iterated on the elements of a set C of random entities c for which the model does not lead to the object o for predicting the filler of $\langle c, p, ?\rangle$.

Algorithm 1: Algorithm for identifying the explanation X^*

```
Input: the triples in $\mathcal{F}^s_{train}$; the Relevance Engine $engine$;
       the acceptance threshold $\xi_0$; the explanation size limit $i_{max}$;
Output: The smallest combination $X^*$ whose relevance exceeds $\xi_0$;
1  $triple\_to\_relevance \leftarrow \{t : engine.compute([t])$ for $t$ in $F^s_{train}\}$;
2  for $i \leftarrow 2$ to $i_{max}$ do
3      $\mathcal{S}_i \leftarrow combinations(F^s_{train}, i)$;
4      $pre\_relevances \leftarrow [avg([triple\_to\_relevance[t]$ for $t$ in $X])$ for $X$ in $\mathcal{S}_i]$;
5      $\mathcal{S}_i \leftarrow sort(\mathcal{S}_i, pre\_relevances)$;
6      $best\_relevance \leftarrow None$;
7      foreach $X \in \mathcal{S}_i$ do
8          $cur\_relevance \leftarrow engine.compute(X)$;
9          if $cur\_relevance > \xi_0$ then
10             return $X$;
11         else
12             if $best\_relevance = None$ or $cur\_relevance > best\_relevance$ then
13                 $best\_relevance \leftarrow cur\_relevance$;
14             $\rho_i \leftarrow cur\_relevance/best\_relevance$;
15             if $random(0, 1) > \rho_i$ then
16                 break;
```

4.2 Injecting Semantics in the Pre-Filter

The first new component of our proposed methodology is Semantic Pre-Filter, a refined version of the Pre-Filter. We refine the method for assessing the utility of the triples in $\mathcal{G}^s_{train}$. We recall that in KELPIE the Pre-Filter calculates the utility of a triple $\langle s, q, r\rangle$ (or $\langle r, q, s\rangle$) with respect to the prediction $\langle s, p, o\rangle$ based on a breadth-first search (*bfs*) measuring the length of the shortest non-oriented path connecting r to o. Now, we base this calculation on the path (connecting r to o) with the least cumulative weight instead of the shortest one, making the *bfs* weighted (*wbfs*). Specifically, the weight of an edge connecting r to o is $1 - sim(r, o)$. Such weight is meant as a measure of the semantic similarity between the two connected nodes. This enhancement involves the integration of the approximated semantic similarity measure proposed in [10] as a measure over pairs of entities. Namely, in [10] a function Cl is defined as returning classes to which an entity can be proven to belong to, and its approximated version $\tilde{Cl}$ that simplifies the needed *realization* service and allows bypassing the usage of *retrieval* required in Cl.

The semantic similarity measure sim is formally defined as a *Jaccard* measure on a couple of entities r, o. Specifically, $sim(r, o) = \frac{|\tilde{Cl}(r) \cap \tilde{Cl}(o)|}{|\tilde{Cl}(r) \cup \tilde{Cl}(o)|}$.

For instance, consider a very simple KG with the following triples: $\{$ $\langle$*Barack Obama*, *signed*, *Obamacare*$\rangle$, $\langle$*Barack Obama*, *born in*, *Honolulu*$\rangle$, $\langle$*Honolulu*, *located in*, *United States*$\rangle$, $\langle$*United States*, *signed*, *Obamacare*$\rangle$ $\}$ and the prediction $\langle$*Barack Obama*, *nationality*, *United States*$\rangle$. Both *Honolulu* and *Obamacare* are one-hop distant from *United States*. Nevertheless, *Honolulu* has higher semantic similarity with *United States* than *Obamacare*.

Algorithm 2: Algorithm for identifying X^* in the quotient graph

```
Input: The set $\mathcal{F}^s_{train}$ of training triples; $Cl$ function;
       the Relevance Engine object $engine$;
       the acceptance threshold $\xi_0$; the explanation size limit $i_{max}$;
Output: The smallest combination $X^*$ whose relevance exceeds $\xi_0$;
1  $\mathcal{Q}^s_{train} \leftarrow compute\_quotient\_graph(F^s_{train}, Cl)$;
2  $quot\_to\_orig \leftarrow compute\_mapping(\mathcal{Q}^s_{train}, F^s_{train})$;
3  $triple\_to\_relevance \leftarrow \{\}$;
4  foreach $triple_{quot} \in Q^{train}_s$ do
5      $triples_{orig} \leftarrow quot\_to\_orig([triple_{quot}])$;
6      $triple\_to\_relevance[triple_{quot}] \leftarrow engine.compute(triples_{orig})$;
7  for $i \leftarrow 2$ to $i_{max}$ do
8      $\mathcal{S}_i \leftarrow combinations(\mathcal{Q}^s_{train}, i)$;
9      $\mathcal{S}_i \leftarrow preliminary\_sort(\mathcal{S}_i, triple\_to\_relevance)$;
10     foreach $X_{quot} \in \mathcal{S}_i$ do
11         $X_{orig} \leftarrow quot\_to\_orig(X_{quot})$;
12         $cur\_relevance \leftarrow engine.compute(X_{orig})$;
13         $check\_accept\_threshold(cur\_relevance, \xi_0)$;
14         $update\_best\_relevance(best\_relevance, cur\_relevance)$;
15         $check\_early\_exit(best\_relevance, cur\_relevance)$;
```

This integration of semantics fundamentally changes how the Pre-Filter assesses the utility of triples within the $\mathcal{G}^s_{train}$ set. The intuition behind this enhancement is to acknowledge the semantic relationships between nodes (entities). Adopting this refined version, $\mathcal{F}^s_{train}$ will eventually contain triples that are not only topologically, but also semantically related to the prediction. This contribution may result in a more accurate extraction of $\mathcal{F}^s_{train}$ enhancing the effectiveness of the explanations; thus addressing our final research goal.

4.3 Injecting Semantics in the Explanation Builder

We propose Quotient Explanation Builder to enhance the Explanation Builder component of KELPIE. In brief, given a prediction $\langle s, p, o\rangle$ to explain, the Explanation Builder combines the pre-filtered training triples ($\mathcal{F}^s_{train}$) into candidate explanations and then identify optimal ones (details in Algorithm 1).

Our enhanced version tackles the goal of limiting the combinatorial explosion when computing candidate explanations by introducing a suitable summarization step of the input subgraph. For the purpose, the method relies on the notion of quotient graph, presented in Subsect. 2.2.

We formalize Quotient Explanation Builder in Algorithm 2. The algorithm requires the same parameters as the original formulation, along with the inclusion of the $\tilde{Cl}$ function. Initially, the algorithm computes the quotient graph $\mathcal{Q}^s_{train}$ of $\mathcal{F}^s_{train}$ exploiting $\tilde{Cl}$ to determine the equivalence relation (Line 1).

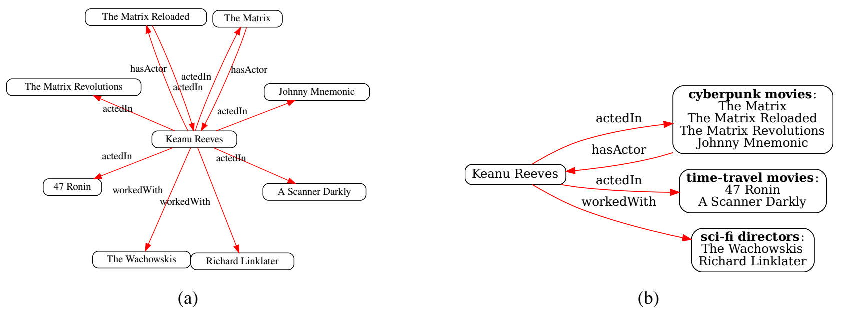


Fig. 1: sub-graph on *Keanu Reeves* (a) and its (simulation) quotient graph (b)

We refer to a triple in the quotient graph as a *quotient triple* ($triple_{quot}$), and to a triple in the original graph as an *original triple* ($triple_{orig}$). The nodes in the quotient graph are equivalence classes of nodes in the original graph; hence, each quotient triple is mapped to its inherently equivalent original triples (Line 2).

For example, we consider *Keanu Reeves* as the subject s, for which we report $\mathcal{F}^s_{train}$ in Fig. 1a and its quotient graph in Fig. 1b. The quotient triple $\langle s, \mathit{actedIn}, \mathit{cyberpunk\ movies}\rangle$ (where *cyberpunk movies* is a class) is mapped to the original triples: $\langle s, \mathit{actedIn}, \mathit{The\ Matrix}\rangle$, $\langle s, \mathit{actedIn}, \mathit{The\ Matrix\ Reloaded}\rangle$, $\langle s, \mathit{actedIn}, \mathit{The\ Matrix\ Revolutions}\rangle$, $\langle s, \mathit{actedIn}, \mathit{Johnny\ Mnemonic}\rangle$.

The subsequent steps in Algorithm 2 follow a procedure similar to Algorithm 1, but performing all the computations on $\mathcal{Q}^s_{train}$. It computes the relevance of each quotient triple used as a 1-quotient-triple explanation. It first retrieves the original triples that it corresponds to (Line 5), then it computes the relevance of such original triples using the Relevance Engine (Line 6).

The process then continues by exploring combinations of quotient triples from the smallest size (2) up to a specified limit i_{max} (Line 7). $\mathcal{S}_i$ is sorted in descending order of preliminary relevance, then the algorithm computes the relevance of each X_{quot}. Similarly to the case of 1-quotient-triple explanations, it retrieves the corresponding original triples (Line 11), then it computes their relevance (Line 12). The remaining lines check the heuristic conditions.

The quotient graph condenses the original graph, thus addressing the goal of decreasing the number of candidate explanations. Moreover, quotient graphs contribute to the other goal of generating explanations at varying levels of detail, as both the quotient triples and the entities within the quotient nodes can be output. Finally, the inherent grouping of entities in a quotient graph may enhance the effectiveness of explanations addressing the overall final goal.

A quotient triple represents a collection of similar original triples, so it provides multiple pieces of evidence of the same kind. Hence, a set of quotient triples represents different kinds of evidence, each backed by multiple pieces. This pattern furthers heterogeneity and robustness in the explanation. In essence, these goals hinge on grouping of semantically related triples, an aspect effectively handled by quotient graphs.

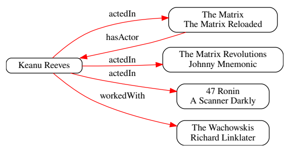
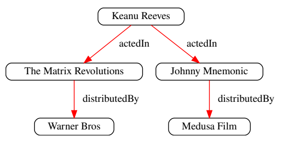
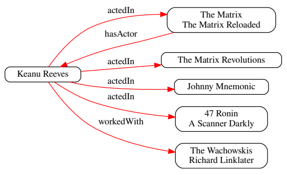


Fig. 2: *bisimulation quotient* of $\mathcal{G}^{s}_{train}$



(a)



(b)

Fig. 3: fragment of $\mathcal{G}^{s(+)}_{train}$ (a) and its *bisimulation quotient* (b)

The introduced abstraction potentially enhances the method. However, it increases the risk to bypass relevant smaller sets or individual triples. Hence, the use of the quotient graph allows for a trade-off between abstraction and detail.

To optimize the choice, we propose three alternative formulations of the quotient graph: (i) *simulation quotient*, (ii) *bisimulation quotient*, and (iii) *depth-1 bisimulation quotient*. Algorithm 2 is agnostic of the formulation.

The *simulation quotient* computes the quotient graph according to the simulation relation. Being the most abstract formulation, it is the most prone to ignore small sets and individual triples. In Fig. 1b we already provided an example.

In the *bisimulation quotient*, we adopt the bisimulation relation. It splits some equivalence classes to meet the stricter condition on edges, mitigating the risk of the first method. For instance, in Fig. 2 the equivalence class *cyberpunk movies* is split in two nodes according to the presence of the property *hasActor*. This helps when the prediction relies on either the first two movies or the other two. We clarify that, in this formulation, we compute the RSCP of the partition resulting from $\tilde{Cl}$ rather than computing it on the whole set of nodes in $\mathcal{F}^{s}_{train}$, thus maintaining the semantic information provided $\tilde{Cl}$. We also recall that to compute RSCP, a pre-processing step is required converting each triple $\langle s, p, o\rangle$ to an unlabeled edge from s to a new node p, o. However, for the new nodes $\tilde{Cl}$ is not defined. To address this aspect, we have extended $\tilde{Cl}$ so to assign each new node to a separate equivalence class.

So far, information not directly featuring s was not taken into account while looking for local explanations. However, additional information may be employed only to further refine the equivalence classes, leading to the *depth-1 bisimulation quotient*. The algorithm starts by finding the depth-1 sub-graph of s ($\mathcal{G}^{s(+)}_{train}$) which includes neighbors of neighbors of s. In Fig. 3a we exemplify this by reporting a fragment for $s =$ *Keanu Reeves*. Secondly, it computes the *bisimulation quotient* of $\mathcal{G}^{s(+)}_{train}$, as shown in Fig. 3b. The equivalence class $\{$*The Matrix Revolutions*, *Johnny Mnemonic*$\}$ is further split in two nodes according to the property *produced by*. This helps when the prediction solely relies either on *The Matrix Revolutions* or on *Johnny Mnemonic*. In this formulation, quotient triples that do not include s will arise. They can be discarded as the graph is expanded only for refining the equivalence classes rather than increasing the set of candidate explanations.

In conclusion, we remark that *bisimulation quotient* and *depth-1 bisimulation quotient* have a finer granularity than *simulation quotient*, but still explanations are more abstract than those produced by KELPIE. The *depth-1 bisimulation quotient*, that is the finest of our formulations can still bypass individual triples, only KELPIE assesses the relevance of all the triples in $\mathcal{F}^{s}_{train}$. Our assumption is that the bypassed triples are less relevant explanations than the sets resulting from the quotient graphs by virtue of the heterogeneity and robustness of explanations resulting from the combination of quotient triples.

5 Experimental Evaluation

In this section, we illustrate the experiments carried out to assess our contributions with respect to our research goals (stated in Sect. 1). All the code, datasets, and trained models utilized in our study are openly accessible on GitHub[3]. In this section we present our experimental setting, then we provide quantitative and qualitative results.

5.1 Experimental Setting

We performed the experiments on three datasets: DBpedia50 [29] (DB50K), DB100K [12] and YAGO4-20. DB50K and DB100K are both samples of the DBpedia KG. In contrast, we sampled YAGO4-20 from the YAGO4 [23] KG by extracting triples about entities involved in at least 20 triples and then filtering out triples with literal objects. We report statistics on the datasets in Tab. 1.

The common aspect of these datasets is that along with the RDF triples they include or can be integrated with OWL (specifically OWL2-DL) statements including class assertions, and other schema axioms concerning classes and relationships. We exploited the HERMIT [16] reasoner offline to materialize the implicit class assertions in the KGs. Next, we implemented $\tilde{Cl}$ as a simple lookup of class assertions to execute our method without any other adjustment.

We highlight that for DB50k and DB100K, only the RDF triples are available off the shelf. We retrieved the class assertions with custom SPARQL queries to the DBpedia endpoint [4], while we employed the schema axioms provided by the full DBpedia [5]. The resulting KGs DB50K and DB100K proved to be inconsistent, while YAGO4-20 turned out to contain unsatisfiable classes. We manually repaired the KGs (see Appendix A) as such problems hinder the use of the reasoner.

KELPIE and consequently our extension, supports any LP model based on embeddings; therefore, we performed our experiments on TRANSE [6], CONVE [11] and

[3] `https://github.com/rbarile17/kelpiePP`

[4] `https://dbpedia.org/sparql`

[5] `https://databus.dbpedia.org/ontologies/dbpedia.org/ontology--DEV`

Table 1: Statistics of the datasets

	Entities	Relations	Train triples	Valid triples	Test triples
DB50K	24620	351	32194	123	2095
DB100K	98776	464	587688	49172	49114
YAGO4-20	96910	70	555182	69398	69398

Table 2: Performance of the LP models

	DB50K		**DB100K**		**YAGO4-20**	
	H@1	MRR	H@1	MRR	H@1	MRR
TRANSE	0.288	0.380	0.101	0.211	0.081	0.135
CONVE	0.366	0.410	0.285	0.360	0.163	0.213
COMPLEX	0.407	0.464	0.359	0.430	0.195	0.242

COMPLEX [31] as KELPIE adopts the same ones and these represent three prevalent families: Geometric Models, Deep Learning Models, and Tensor Decomposition Models. Tab. 2 reports the LP performance of each model that we measured on each dataset in terms of typical measures, namely Mean Reciprocal Rank (MRR) and Hits-at-1 ($H@1$) in their filtered variant.

We evaluate the approach adopting both *necessary* and *sufficient* relevance (details in Subsect. 4.1). In both scenarios, we select a set P of 100 correct test predictions randomly, for each model. Then, we adopt the methodology utilized by CRIAGE, and subsequently by KELPIE, to assess if we meet our goal of optimizing the effectiveness. Specifically, in the *necessary* scenario, after extracting explanations for all predictions in P, we remove their triples and retrain the model. Since the original model correctly led to those predictions, their initial $H@1$ and MRR are both 1.0; however, if the extracted explanations are indeed necessary, the inference through the retrained model should fail to lead to the predictions in P. Therefore, the effectiveness is the decrease in $H@1$ and MRR over P.

Conversely, in the *sufficient* scenario, for each prediction $\langle s, p, o \rangle \in P$, we draw a set C of 10 random entities c and extract sufficient explanations that should lead the model to predict $\langle c, p, o \rangle$. We define P_C as the set of these hypothetical $10 \times 100 = 1000$ predictions $\langle c, p, o \rangle$. As the original model did not lead to the predictions in P_C by design, their original $H@1$ and MRR are approximately null. However, if the extracted explanations are indeed sufficient, the retrained model should lead to the predictions in P_C. Therefore, the effectiveness is the increase in the $H@1$ and MRR over P_C. In both scenarios, the variation metrics are denoted as $\Delta H@1$ and ΔMRR. Moreover, as for our first research goal, we aim at improving the efficiency by decreasing the number of candidate explanations. Hence, we also measure the number of Relevance Engine invocations (indicated by # r). In Appendix B we report details on the hyper-parameters used for training the models, for the *post-training* in the extraction of the explanations, and for the re-training in the evaluation of the explanations.

5.2 Quantitative Evaluation

In Tab. 3 we report the outcomes of the experiments on KELPIE with the various extensions in terms of the effectiveness and efficiency metrics. Firstly, we measure the

Table 3: Results of the experimental evaluation

		Necessary									**Sufficient**								
		DB50K			**DB100K**			**YAGO4-20**			**DB50K**			**DB100K**			**YAGO4-20**		
		# r	$\Delta H@1$	ΔMRR	# r	$\Delta H@1$	ΔMRR	# r	$\Delta H@1$	ΔMRR	# r	$\Delta H@1$	ΔMRR	# r	$\Delta H@1$	ΔMRR	# r	$\Delta H@1$	ΔMRR
TRANSE	KELPIE	1234	-0.93	**-0.867**	2568	**-0.69**	-0.546	2614	-0.48	-0.356	1086	0.521	0.580	3109	0.317	0.443	6872	0.260	0.365
	wbfs	1241	**-0.94**↑	-0.857	2420	-0.64	-0.511	2771	-0.49↑	-0.357↑	1097	0.499	0.562	3065	**0.323**↑	0.441	6953	0.299↑	0.391↑
	sim	510	-0.93	-0.856	1953	-0.63	-0.518	**1437**	**-0.57**↑	**-0.450**↑	362	0.449	0.515	2464	0.298	0.426	3819	**0.377**↑	**0.475**↑
	bisim	752	-0.91	-0.815	1954	-0.58	-0.481	1470	-0.57↑	-0.433↑	622	0.508	0.568	**2345**	0.287	0.422	3789	0.335↑	0.441↑
	bisim_d1	925	-0.91	-0.836	2555	-0.61	-0.481	2461	-0.46	-0.332	804	**0.528**↑	**0.587**↑	3210	0.320↑	0.440	6858	0.270	0.390↑
	wbfs + sim	**502**	**-0.94**↑	-0.861	**1860**	-0.68	**-0.566**↑	1458	-0.56↑	-0.443↑	**361**	0.475	0.531	2465	0.301	0.432	**3774**	0.324↑	0.434↑
	wbfs + bisim	762	-0.90	-0.818	2037	-0.65	-0.551	1474	-0.55↑	-0.439↑	632	0.482	0.556	2517	0.32↑	**0.446**↑	4035	0.343↑	0.434↑
	wbfs + bisim_d1	938	-0.91	-0.824	2555	-0.65	-0.533	2422	-0.46	-0.333	813	0.499	0.565	3288	0.307	0.429	6239	0.278	0.387↑
CONVE	KELPIE	818	-0.42	-0.385	1590	-0.60	-0.483	5349	-0.39	-0.263	1177	0.629	**0.648**	6090	0.141	0.184	9550	0.349	0.378
	wbfs	817	-0.44↑	-0.397↑	1558	**-0.91**↑	**-0.833**↑	5481	-0.36	-0.256	1177	0.602	0.605	5485	0.140	0.201↑	9514	0.345	0.381↑
	sim	267	**-0.50**↑	**-0.463**↑	1083	-0.84↑	-0.771↑	3360	-0.41↑	-0.308↑	**520**	0.610	0.599	**4380**	0.254↑	0.333↑	**4953**	0.386↑	0.399↑
	bisim	341	-0.48↑	-0.427↑	1229	-0.64↑	-0.481	**3269**	**-0.43**↑	**-0.322**↑	636	0.615	0.607	5230	0.381↑	0.444↑	5077	**0.387**↑	0.396↑
	bisim_d1	518	-0.45↑	-0.390↑	1891	-0.80↑	-0.697↑	5258	-0.38	-0.282↑	864	0.626	0.622	5158	**0.423**↑	**0.468**↑	8315	0.343	0.365
	wbfs + sim	**258**	-0.42	-0.389↑	**1047**	-0.79↑	-0.716↑	3373	-0.39	-0.294↑	530	0.565	0.576	5173	0.403↑	0.467↑	5013	0.379↑	**0.403**↑
	wbfs + bisim	350	-0.41	-0.375	1701	-0.85↑	-0.764↑	3362	-0.41↑	-0.308↑	634	0.635↑	0.631	4580	0.247↑	0.321↑	5086	0.349	0.366
	wbfs + bisim_d1	518	-0.46↑	-0.396↑	1919	-0.81↑	-0.727↑	5030	-0.39	-0.277↑	1026	**0.658**↑	0.639	5717	0.386↑	0.433↑	9072	0.347	0.368
COMPLEX	KELPIE	626	**-0.98**	**-0.944**	2682	-0.82	-0.723	3882	-0.65	-0.499	261	0.795	0.720	1263	0.658	0.705	1310	**0.543**	**0.519**
	wbfs	626	**-0.98**	**-0.944**	2760	-0.84↑	-0.726↑	3939	-0.59	-0.461	261	0.795	0.720	1263	0.659↑	**0.708**↑	1315	0.520	0.503
	sim	549	-0.96	-0.895	1783	-0.80	-0.684	**1515**	-0.62	-0.510	219	0.777	0.716	774	0.526	0.624	**576**	0.475	0.463
	bisim	571	-0.90	-0.872	1952	**-0.85**↑	**-0.733**↑	1606	-0.62	-0.507	225	0.772	0.705	897	0.581	0.666	595	0.461	0.449
	bisim_d1	617	-0.98	-0.921	2419	-0.81	-0.693	2930	-0.61	-0.505	259	**0.804**↑	**0.726**↑	1228	0.657	0.697	1153	0.494	0.476
	wbfs + sim	**548**	-0.97	-0.935	**1758**	-0.80	-0.710	1527	-0.64	**-0.551**↑	**217**	0.790	0.723↑	**740**	0.499	0.598	578	0.498	0.482
	wbfs + bisim	573	-0.90	-0.872	1910	-0.81	-0.704	1584	**-0.66**↑	-0.534↑	222	0.779	0.718	857	0.556	0.646	597	0.471	0.463
	wbfs + bisim_d1	617	-0.98	-0.921	2506	-0.83↑	-0.722	2900	-0.57	-0.471	259	0.804↑	0.726↑	1228	**0.660**↑	0.699	1148	0.481	0.472

impact of the Semantic Pre-Filter (*wbfs*). Moreover, we gauge the effects of the Quotient Explanation Builder in its various formulations: *sim* (*simulation quotient*), *bisim* (*bisimulation quotient*), and *bisim_d1* (*depth-1 bisimulation quotient*). Finally, we assess how KELPIE performs when equipped with both contributions reporting *wbfs* + *sim*, *wbfs* + *bisim*, and *wbfs* + *bisim_d1*. In all experiments, we set the Pre-Filter k threshold to 20 as in KELPIE.

Across almost all configurations of scenario, model, and dataset, the Quotient Explanation Builder improves the effectiveness through at least one of the alternative formulations. Exceptions were observed in the *necessary* scenario with TRANSE on DB50K and DB100K, the *sufficient* scenario with CONVE on DB50K, in the *necessary* scenario with COMPLEX on DB50K, and in the *sufficient* scenario with COMPLEX on YAGO4-20. The Semantic Pre-Filter improves its effectiveness either when adopted independently or when combined with the Quotient Explanation Builder. It has no benefit in the *sufficient* scenario with TRANSE on DB50K, in the *necessary* scenario with CONVE on YAGO4-20, in the *necessary* scenario with COMPLEX on DB50K, and in both scenarios with COMPLEX on YAGO4-20. We posit that the cases of limited performance can be attributed to the selected predictions. Specifically, many predictions may have a subject s associated with a limited number of triples in G^s_{train}, thus leading to a limited search space. Explaining such predictions may be challenging also for KELPIE itself as it looks for explanations in G^s_{train} where few potentially effective explanations are available when explaining such predictions.

Particularly, for predictions associated with less than k (Pre-Filter parameter) triples the Pre-Filter selects all the triples in G^s_{train}; hence, any pre-filter is equivalent except for the order of triples. Furthermore, in instances of graphs characterized by 2 or 3 triples, the compressive potential of quotient graphs is restricted, as the resulting quotient graph precisely mirrors the original structure.

Table 4: Examples of explanations with Quotient Explanation Builder and plain KELPIE

Simulation quotient	
quotient triples	**entities**
⟨*Movie*, *actor*, *Gabourey Sidibe*⟩	*Casting By*, *Top Five*, *Seven Psychopaths*, *White Bird in a Blizzard*, *Tower Heist*, *Precious*
⟨*TVSeries*, *actor*, *Gabourey Sidibe*⟩	*Empire (2015 TV Series)*
Depth-1 bisimulation quotient	
quotient triples	**entities**
⟨*Movie - subset 1*, *actor*, *Gabourey Sidibe*⟩	*Top Five*
⟨*Movie - subset 2*, *actor*, *Gabourey Sidibe*⟩	*White Bird in a Blizzard*
⟨*TVSeries*, *actor*, *Gabourey Sidibe*⟩	*Empire (2015 TV Series)*
KELPIE	
⟨*Empire (2015 TV Series)*, *actor*, *Gabourey Sidibe*⟩ ⟨*Top Five*, *actor*, *Gabourey Sidibe*⟩ ⟨*White Bird in a Blizzard*, *actor*, *Gabourey Sidibe*⟩	

For instance, for DB50K 98 out of 100 predictions with TRANSE are associated with less than 20 triples, with 91 predictions associated with less than 5 triples. We focused on streamlining the search in cases of very extensive search spaces, rather than expanding the search space when it is limited. Indeed, for predictions on highly connected entities for which the space of candidate explanations becomes very extensive, our approach enhances KELPIE.

As regards # r, the impact of the Semantic Pre-Filter appears nearly negligible. In contrast, the Quotient Explanation Builder is intended to optimize # r, indeed the lowest value was observed in the cases of *sim* or *wbfs* + *sim* in all configurations. Next values are those reported for *bisim*, *bisim_d1*, and finally for KELPIE, that showed the lowest efficiency, thus aligning with the quotient formulations that feature incremental granularity, with *simulation quotient* being the most abstract. In only two cases, the lowest value is reported for *bisim*, specifically in the *necessary* scenario with CONVE on YAGO4-20 and in the *sufficient* scenario with TRANSE on DB100K. These cases are likely due to meeting exceptionally early the heuristic conditions in KELPIE.

5.3 Qualitative Evaluation

In this part, we show a typical example of explanation output by the proposed method, focusing on the impact of the Quotient Explanation Builder. More specifically, we report *necessary explanations* for the prediction ⟨*Gabourey Sidibe*, *nationality*, *United States*⟩ found in the experiments with COMPLEX on YAGO4-20.

In Tab. 4 we show the explanation computed using the *simulation quotient* and the *depth-1 bisimulation quotient* and the baseline KELPIE. We omit *bisimulation quotient*, as for this prediction it provides the same explanation of *simulation quotient*. The *simulation quotient* provides as explanation both *quotient triples* (first column), and specific entities (second column) supporting our third research goal. Indeed, the *quotient triples* represent relationships between a class expression and the subject of the prediction; then, the specific entities within such a class are explicated. For the given prediction, the first form suggests that the model predicted the correct *nationality* because in the training phase it had seen works like movies and TV series starring *Gabourey Sidibe* as *actor* which is useful to understand the inference made through the model with a general insight based on classes. The second form indicates specific works, which is useful to inspect specific instances and verify their commonalities. Indeed, all the works have the *United States* as country of origin or are related to a producer or actor who, in turn, is related to the *United States*. In the *depth-1 bisimulation quotient* the quotient triples involve two specific subsets of movies, as in this formulation the equivalence classes in the quotient can be split into multiple nodes according to outgoing edges. Indeed, the two sets of movies feature different actors.

The baseline explanation is grounded on the same rationale, retrieving works related to the *United States*. However, it provides a narrower array of evidence. The insight is more precise and more easily comprehensible given its brevity, but limited in terms of coverage. We claim that, despite its length, the explanation with the *simulation quotient* is still intelligible because the effort needed to understand a single triple is approximately the same needed for a *quotient triple* with the corresponding entities. Thus, our approach seems able to provide richer explanations with little extra effort required for their interpretation.

6 Conclusions

We introduced a novel approach for explaining predictions made on KGs that enhances the existing framework KELPIE. Specifically, we identified three research goals: decreasing the number of candidates, providing explanations on different levels of detail, and improving explanation effectiveness. We employed a semantic similarity measure to focus the process on triples semantically related to the prediction. Furthermore, we employed quotient graphs to compact the search space for explanations. We experimentally proved that our solutions furthers the goals on three datasets endowed with semantic information and three representative LP models. We also qualitatively assessed the impact of the quotient graph with respect to the third goal.

A natural progression of this work involves incorporating additional knowledge on a semantic level, e.g., about relationships. Moreover, we plan to explore other graph summarization techniques, such as those in [8]. Furthermore, we aim at improving the explanations for the predictions associated with a limited number of triples. Finally, we may involve users in the evaluation.

Acknowledgments. This work was partially supported by project *FAIR - Future AI Research* (PE00000013), spoke 6 - Symbiotic AI (`https://future-ai-research.it/`), under the NRRP MUR program funded by the NextGenerationEU and by project *HypeKG - Hybrid Prediction and Explanation with Knowledge Graphs* (H53D23003700006), under PRIN 2022 program funded by MUR.

Disclosure of Interests. The authors declare to have no competing interests.

References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: A nucleus for a web of open data. In: ISWC 2007. pp. 722–735. Springer (2007). `https://doi.org/10.1007/978-3-540-76298-0_52`

2. Baltatzis, V., Costabello, L.: Kgex: Explaining knowledge graph embeddings via subgraph sampling and knowledge distillation. arXiv preprint arXiv:2310.01065 (2023)
3. Betz, P., Meilicke, C., Stuckenschmidt, H.: Adversarial explanations for knowledge graph embeddings. In: IJCAI-22. pp. 2820–2826. International Joint Conferences on Artificial Intelligence Organization (2022). `https://doi.org/10.24963/ijcai.2022/391`
4. Bhowmik, R., de Melo, G.: Explainable link prediction for emerging entities in knowledge graphs. In: ISWC 2020. pp. 39–55. Springer (2020). `https://doi.org/10.1007/978-3-030-62419-4_3`
5. Bollacker, K., Cook, R., Tufts, P.: Freebase: A shared database of structured general human knowledge. In: AAAI 2007. pp. 1962–1963. AAAI Press (2007). `https://doi.org/10.5555/1619797.1619981`
6. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. Advances in neural information processing systems **26** (2013). `https://doi.org/10.5555/2999792.2999923`
7. Buneman, P., Staworko, S.: Rdf graph alignment with bisimulation. Proc. VLDB Endow. **9**(12), 1149–1160 (2016). `https://doi.org/10.14778/2994509.2994531`
8. Čebirić, Š., Goasdoué, F., Kondylakis, H., Kotzinos, D., Manolescu, I., Troullinou, G., Zneika, M.: Summarizing semantic graphs: a survey. The VLDB journal **28**(3), 295–327 (2019). `https://doi.org/10.1007/s00778-018-0528-3`
9. Cohen, S., Hershcovitch, M., Taraz, M., Kißig, O., Wood, A., Waddington, D., Chin, P., Friedrich, T.: Drug repurposing using link prediction on knowledge graphs with applications to non-volatile memory. In: Complex Networks 2021. pp. 742–753. Springer (2022). `https://doi.org/10.1007/978-3-030-93413-2_61`
10. d'Amato, C., Masella, P., Fanizzi, N.: An approach based on semantic similarity to explaining link predictions on knowledge graphs. In: WI-IAT 2021. pp. 170–177 (2021). `https://doi.org/10.1145/3486622.3493956`
11. Dettmers, T., Minervini, P., Stenetorp, P., Riedel, S.: Convolutional 2D knowledge graph embeddings. In: AAAI 2018. AAAI Press (2018). `https://doi.org/10.5555/3504035.3504256`
12. Ding, B., Wang, Q., Wang, B., Guo, L.: Improving knowledge graph embedding using simple constraints. In: ACL 2018. pp. 110–121. ACL (2018). `https://doi.org/10.18653/v1/P18-1011`
13. Dong, X.L.: Building a broad knowledge graph for products. In: IEEE-ICDE 2019. pp. 25–25. IEEE (2019). `https://doi.org/10.1109/ICDE.2019.00010`
14. Dovier, A., Piazza, C., Policriti, A.: A fast bisimulation algorithm. In: CAV 2001. pp. 79–90. Springer (2001). `https://doi.org/10.1007/3-540-44585-4_8`
15. Gentilini, R., Piazza, C., Policriti, A.: From bisimulation to simulation: Coarsest partition problems. Journal of Automated Reasoning **31**, 73–103 (2003). `https://doi.org/10.1023/A:1027328830731`
16. Glimm, B., Horrocks, I., Motik, B., Stoilos, G., Wang, Z.: HermiT: an OWL 2 reasoner. Journal of automated reasoning **53**(3), 245–269 (2014). `https://doi.org/10.1007/s10817-014-9305-1`
17. Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., Pedreschi, D.: A survey of methods for explaining black box models. ACM Comput. Surv. **51**(5), 1–42 (2018). `https://doi.org/10.1145/3236009`
18. Hogan, A., Blomqvist, E., Cochez, M., d'Amato, C., de Melo, G., Gutiérrez, C., Kirrane, S., Labra Gayo, J.E., Navigli, R., Neumaier, S., Ngonga Ngomo, A.C., Polleres, A., Rashid, S.M., Rula, A., Schmelzeisen, L., Sequeda, J.F., Staab, S., Zimmermann, A.: Knowledge Graphs. No. 22 in Synthesis Lectures on Data, Semantics, and Knowledge, Springer (2021). `https://doi.org/10.2200/S01125ED1V01Y202109DSK022`, `https://kgbook.org/`

19. Koh, P.W., Liang, P.: Understanding black-box predictions via influence functions. In: ICML'17. pp. 1885–1894. PMLR (2017). https://doi.org/10.5555/3305381.3305576
20. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. Advances in neural information processing systems **30** (2017). https://doi.org/10.5555/3295222.3295230
21. Monroe, D.: AI, explain yourself. Communications of the ACM **61**(11), 11–13 (2018). https://doi.org/10.1145/3276742
22. Paige, R., Tarjan, R.E.: Three partition refinement algorithms. SIAM Journal on computing **16**(6), 973–989 (1987). https://doi.org/10.1137/0216062
23. Pellissier Tanon, T., Weikum, G., Suchanek, F.: Yago 4: A reason-able knowledge base. In: ESWC 2020. pp. 583–596. Springer (2020). https://doi.org/10.1007/978-3-030-49461-2_34
24. Pezeshkpour, P., Tian, Y., Singh, S.: Investigating robustness and interpretability of link prediction via adversarial modifications. arXiv preprint arXiv:1905.00563 (2019)
25. Rossi, A., Barbosa, D., Firmani, D., Matinata, A., Merialdo, P.: Knowledge graph embedding for link prediction: A comparative analysis. ACM Transactions on Knowledge Discovery from Data (TKDD) **15**(2), 1–49 (2021). https://doi.org/10.1145/3424672
26. Rossi, A., Firmani, D., Merialdo, P., Teofili, T.: Explaining link prediction systems based on knowledge graph embeddings. In: SIGMOD'22. pp. 2062–2075 (2022). https://doi.org/10.1145/3514221.3517887
27. Rousseeuw, P.J., Hampel, F.R., Ronchetti, E.M., Stahel, W.A.: Robust statistics: the approach based on influence functions. John Wiley & Sons (2011). https://doi.org/10.1002/9781118186435
28. Shapley, L.S.: A value for n-person games. In: Contributions to the Theory of Games II (1953). Princeton University Press (1997). https://doi.org/10.1515/9781400829156-012
29. Shi, B., Weninger, T.: Open-world knowledge graph completion. In: AAAI 2018. vol. 32 (2018). https://doi.org/10.1609/aaai.v32i1.11535
30. Singhal, A.: Introducing the knowledge graph: things, not strings, https://blog.google/products/search/introducing-knowledge-graph-things-not
31. Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., Bouchard, G.: Complex embeddings for simple link prediction. In: ICML'16. pp. 2071–2080. PMLR (2016). https://doi.org/10.5555/3045390.3045609
32. Ying, Z., Bourgeois, D., You, J., Zitnik, M., Leskovec, J.: GNNExplainer: Generating explanations for graph neural networks. Advances in neural information processing systems **32** (2019), https://dl.acm.org/doi/10.5555/3454287.3455116
33. Zhang, H., Zheng, T., Gao, J., Miao, C., Su, L., Li, Y., Ren, K.: Data poisoning attack against knowledge graph embedding. In: International Joint Conference on Artificial Intelligence. pp. 4853–4859 (08 2019). https://doi.org/10.24963/ijcai.2019/674
34. Zhang, W., Deng, S., Wang, H., Chen, Q., Zhang, W., Chen, H.: Xtranse: Explainable knowledge graph embedding for link prediction with lifestyles in e-commerce. In: JIST 2019. pp. 78–87. Springer (2020). https://doi.org/10.1007/978-981-15-3412-6_8
35. Zhang, W., Paudel, B., Zhang, W., Bernstein, A., Chen, H.: Interaction embeddings for prediction and explanation in knowledge graphs. In: WSDM'19. pp. 96–104 (2019). https://doi.org/10.1145/3289600.3291014

Table 5: Hyper-parameters of the models

	DB50K	**DB100K**	**YAGO4-20**
TransE	D: 256, p: 2, Ep: 65, Lr: 0.0017, B: 2048, γ: 5, N: 5	D: 256, p: 2, Ep: 133, Lr: 0.0036, B: 2048, γ: 5, N: 5	D: 128, p: 2, Ep: 59, Lr: 0.0098, B: 2048, γ: 10, N: 5
ConvE	D: 200, $Drop$: $\{in$: 0, h: 0, $feat$: $0\}$, Ep: 69, Lr: 0.018, B: 512	D: 200, $Drop$: $\{in$: 0, h: 0.2, $feat$: $0\}$, Ep: 109, Lr: 0.0432, B: 512	D: 200, $Drop$: $\{in$: 0.2, h: 0.1, $feat$: $0.3\}$, Ep: 512, Lr: 0.0427, B: 512
ComplEx	D: 200, Ep: 43, Lr: 0.043, B: 512	D: 200, Ep: 83, Lr: 0.0814, B: 512	D: 200, Ep: 88, Lr: 0.0425, B: 512

A Appendix: Repair of Inconsistent and Unsatisfiable Ontologies

We integrated the datasets DB100K and DB50K with OWL schema axioms retrieved from various sources. The resulting ontologies were inconsistent; hence, we manually repaired them. Moreover, YAGO4-20 turned out to contain unsatisfiable classes. Specifically, we identified the causes of such problems by running the explanation facility of the reasoner. In this appendix, we report some insights on the adjustments that we performed to make DB100K consistent and YAGO4-20 satisfiable, hence "reasonable". In our GitHub repository, we report all the adaptations.

For DB100K, we modified the types for certain instances. For instance, our SPARQL query to retrieve the class assertions returned *Politician*, and *TimePeriod* for several entities. Such types led to inconsistencies as *Politician* is a subclass of *Person* which, in turn, is disjoint with *TimePeriod*. We modified these entities, keeping only the type *Politician*.

We also modified schema axioms in certain cases to preserve several triples, which otherwise would have led to inconsistencies. For instance, we modified the range of the property *location*. We recall that the range of a relationship p specify the classes whose instances can occur as object in a triple with predicate p. We changed the range from *Place* to *Place*$\sqcup$*Company*. Through this adjustment, we accommodate also triples having *location* as predicate and an instance of *Company* as object.

In other cases, we needed to remove triples. For instance, we removed the triple $\langle$*Subramanian_Swamy*, *region*, *Economics*$\rangle$. These triples caused an inconsistency because the type of *Economics* is, *University* which in turn is a descendant of *Agent* in the class hierarchy. However, the range of *region* is *Place* which is disjoint with *Agent*.

For YAGO4-20, we removed certain *subClassOf* axioms. For instance, the class *Districts_of_Slovakia* was a subclass of *AdministrativeArea*, *Product*, and *CreativeWork*. The class *Districts_of_Slovakia* was unsatisfiable as *AdministrativeArea* is subclass of *Localization* which is disjoint with *CreativeWork*. We modified it keeping only *AdministrativeArea* and *Product* as super-classes.

B Appendix: Hyper-parameters

In this appendix, we report in Tab. 5 the hyper-parameters that we adopted to train each model on each dataset. Furthermore, we employed the same set of hyper-parameters to execute the *post-training* in the extraction of the explanations and to retrain the models in the evaluation of the explanations.

Note that:

- D is the embedding dimension, in the models that we adopted entity and relation embeddings always have same dimension
- p is the exponent of the p-norm
- Lr is the learning rate
- B is the batch size
- Ep is the number of epochs
- γ is the margin in the Pairwise Ranking Loss
- N is the number of negative triples generated for each positive triple
- ω is the size of the convolutional kernels
- $Drop$ is the training dropout rate, specifically:
  - in is the input dropout
  - h is the dropout applied after a hidden layer
  - $feat$ is the feature dropout

We adopted random search to find the values of the hyper-parameters. Exceptions are given by B and Ep. For B we adopted the value leading to optimize execution times and parallelism. For Ep we adopted early stopping with 1000 as maximum number of epochs and 5 as patience threshold during the training of the models, and we reported the epoch on which the training stopped. Hence, we used such value as number of epochs in the *post-training* and in the evaluation. Furthermore, as in KELPIE, for TRANSE we adopted the learning rate (Lr) values in Tab. 5 during training and evaluation, but for the *post-training* we used a different value. For TRANSE the batch size (B) is particularly large (2048) and usually exceeds by far the number of triples featuring an entity. This affects *post-training* because in any *post-training* epoch the entity would only benefit from one optimization step. We easily balanced this by increasing the Lr to 0.01.