Explaining Link Prediction Systems based on Knowledge Graph Embeddings

Andrea Rossi
andrea.rossi3@uniroma3.it
Roma Tre University
Rome, Italy

Donatella Firmani
donatella.firmani@uniroma1.it
Sapienza University
Rome, Italy

Paolo Merialdo
paolo.merialdo@uniroma3.it
Roma Tre University
Rome, Italy

Tommaso Teofili
tommaso.teofili@uniroma3.it
Roma Tre University
Rome, Italy

ABSTRACT

Link Prediction (LP) aims at tackling Knowledge Graph incompleteness by inferring new, missing facts from the already known ones. The rise of novel Machine Learning techniques has led researchers to develop LP models that represent Knowledge Graph elements as vectors in an embedding space. These models can outperform traditional approaches and they can be employed in multiple downstream tasks; nonetheless, they tend to be opaque, and are mostly regarded as black boxes. Their lack of interpretability limits our understanding of their inner mechanisms, and undermines the trust that users can place in them. In this paper, we propose the novel Kelpie explainability framework. Kelpie can be applied to any embedding-based LP models independently from their architecture, and it explains predictions by identifying the combinations of training facts that have enabled them. Kelpie can extract two complementary types of explanations, that we dub *necessary* and *sufficient*. We describe in detail both the structure and the implementation details of Kelpie, and thoroughly analyze its performance through extensive experiments. Our results show that Kelpie significantly outperforms baselines across almost all scenarios.

CCS CONCEPTS

• Computing methodologies → Machine learning; **• Information systems**;

KEYWORDS

Knowledge Graphs; Machine Learning; XAI; Link Prediction

ACM Reference Format:
Andrea Rossi, Donatella Firmani, Paolo Merialdo, and Tommaso Teofili. 2022. Explaining Link Prediction Systems based on Knowledge Graph Embeddings. In *Proceedings of the 2022 International Conference on Management of Data (SIGMOD '22), June 12–17, 2022, Philadelphia, PA, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3514221.3517887

SIGMOD '22, June 12–17, 2022, Philadelphia, PA, USA.
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9249-5/22/06…$15.00
https://doi.org/10.1145/3514221.3517887

1 INTRODUCTION

Knowledge Graphs (KGs) are structured representations of real-world information where nodes embodying *entities* are linked by directed edges denoted by labels conveying semantic *relations* forming triples called *facts*. In time, several KGs have achieved web-scale size, e.g., Freebase [7], DBPedia [4], Yago [50], and, in industry, the Google KG [48]. Nonetheless, all KGs are plagued by *incompleteness*, as they only hold a small fraction of the information they should encompass [2, 58]. *Link Prediction* (LP) tackles this issue by analyzing the already known facts to infer the missing ones; for instance, knowing facts ⟨*Barack_Obama*, *born_in*, *Honolulu*⟩ and ⟨*Honolulu*, *located_in*, *USA*⟩, one may deduce ⟨*Barack_Obama*, *nationality*, *USA*⟩ (assuming it was unknown in the KG).

Most LP systems map KG elements to low-dimensional vectors, dubbed *KG embeddings*, that are automatically learned applying Machine Learning (ML) techniques to leverage the known facts in the KG. Since the seminal work by Bordes *et al.* [9] embedding-based LP has rapidly become a sparkling topic, with dozens of new models proposed every year (see the works by Nguyen [37] and Wang *et al.* [56], for recent surveys). The most effective models relying on embeddings have been shown to outperform traditional rule-based counterparts on most research datasets [43], thus gaining even further traction. These systems have already been successfully applied to numerous downstream tasks, such as fact checking [21] and recommendation [61]. Approaches related to LP have also influenced other linking problems in the Data Management field: in Entity Matching, whose goal is identifying which entries in a data store refer to the same concepts [14, 28, 36], word embeddings have been recently coupled with entity embeddings with promising results [26, 35]. Moreover, embedding-based LP models have proved effective in Entity Alignment, finding pairs of entities from different KGs that refer to the same notion [53]. Given the progress witnessed so far, this trend is likely to grow even stronger in the future.

Like most ML systems, embedding-based LP models are not directly interpretable, and despite the growing body of literature on novel LP approaches, so far not much research has been devoted to explaining their outcomes. Shedding light on the behaviour and predictions of opaque AI systems is the purpose of *eXplainable AI (XAI)* [34]: with the widespread adoption of non-transparent ML-based models in data management, explaining their predictions has become an urgent, yet often very challenging, task [20].

In the case of LP, identifying which training facts have been most influential to a prediction amounts to revealing which pieces of evidence the model is leveraging: this, in turn, enables the users to assess whether the model can be trusted or not. For instance, in the previous example ⟨*Barack_Obama, nationality, USA*⟩, obtaining an explanation that matches human intuition, e.g., ⟨*Barack_Obama, born_in, Honolulu*⟩, would increase our trust in the model; on the contrary, an obscure explanation like ⟨*Barack_Obama, handedness, left-handed*⟩ would reveal that the model is probably influenced by spurious correlations.

This kind of interpretability is not just useful in KG completion, where we need to ensure the reliability of our sources: in several domains of application of LP research it may even be an inherent requirement. For example, as reported by Bonner *et al.* [8], LP systems have been recently used in the biomedical field with promising results. Tasks such as drug discovery and repurposing can be successfully modeled as the prediction of missing links between drugs and diseases [32]; LP models have been shown to correctly predict therapeutical relations between various gene proteins and Rheumatoid Arthritis [38]. In these scenarios, as highlighted by Gaudelet *et al.* [16], the capability to explain predictions is strongly desired, but seldom provided by the current LP models. As yet another example, it has been recently highlighted by Pezeshkpour *et al.* [40] that LP explainability frameworks can support the identification of biases and even errors in the original KGs. In this regard, we report in our experimental section an example of bias revealed by explanations.

In this paper, we propose a full-fledged framework for explaining embedding-based link predictions, that we name *Kelpie* (Knowledge graph Embeddings for Link Prediction: Interpretable Explanations). The Kelpie framework can be applied to any LP model based on embeddings, independently of their architecture and components. Given a prediction, Kelpie explains it by computing the subset of training facts enabling the model to return it. In this regard, we identify two complementary approaches: the explanation can be seen as either the set of facts in absence of which the model would not have been capable to yield that prediction; or as the set of facts that, if featured by any entity, would lead the model to yield that prediction. We dub these two settings *necessary* and *sufficient* scenarios, and we design Kelpie to support both of them.

Following the taxonomy by Guidotti *et al.* on XAI methods [17], Kelpie belongs to the category of local Black Box Explanation, also called local post-hoc interpretability methods. In other words, Kelpie explanations address specific outcomes of a trained model rather than its global behaviour.

Contributions Our main contributions are the following:

- We provide a formal definition of explanation for the LP task, and introduce the concepts of *necessary* and *sufficient* explanations in this context.
- We introduce the Kelpie framework, which is specifically designed to extract both sufficient and necessary explanations, and which can support any embedding-based LP model.
- We accompany our framework with a full fledged implementation suited for the vast majority of LP systems in literature.
- We report extensive experiments assessing the effectiveness of Kelpie on multiple models and datasets, and compare its results to those obtained by pre-existing techniques.

Our code, datasets, trained models, and all the resources used in our work are publicly available in our GitHub repository. [1]

Paper Outline Section 2 provides an overview of how embedding-based models tackle the LP task, and formulates the concept of explanations for this setting. Section 3 discusses related work. Section 4 describes the Kelpie framework, both in its architecture and in its implementation. Section 5 reports the experimental evaluation, discussing the obtained outcomes also in comparison to pre-existing methods in literature. Section 6 provides concluding remarks.

2 PROBLEM OVERVIEW

This section defines key preliminary concepts that we refer to in our work, and introduces the problem of explaining Link Prediction.

2.1 Link Prediction in Knowledge Graphs

We define a Knowledge Graph (KG) as a labeled directed graph $KG = (\mathcal{E}, \mathcal{R}, \mathcal{G})$: $\mathcal{E}$ is a set of nodes representing *entities*; $\mathcal{R}$ is a set of labels representing *relations*; and $\mathcal{G} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ is a set of edges representing *facts* that link entities via relations. Each fact is thus a triple $\langle h,\ r,\ t\rangle$ where h is the *head*, r is the *relation*, and t is the *tail*.

Link Prediction (LP) leverages the known facts in a graph to infer the missing ones. The currently most popular approach to LP is by far to use ML techniques to learn *KG embeddings*, i.e., vectorized representations of entities and relations. Embeddings embody the semantics of the original KG elements, and can thus be used to predict new links in the graph. In the following, we will denote KG elements in *italics* and the corresponding embeddings in **bold**.

Embedding-based LP models typically define a *scoring function* ϕ to estimate the plausibility of facts based on the embeddings of their elements. In training, models learn embeddings that optimize the scores of the known facts: ideally, they should generalize and deem unseen true facts as highly plausible as well. In the following, we report formulations where higher ϕ values convey better plausibility; symmetric formulations can be derived for models that work in the opposite way. In addition to entity and relation embeddings, models may learn *shared parameters* not directly linked to any KG element (e.g., the weights of neural layers). Once the training is over, predictions are performed by identifying which entities, if added to incomplete triples as heads or tails, yield the best scores. A *tail prediction* $\langle h,\ r,\ t\rangle$ is the process that finds t as the best scoring tail (i.e., the most plausible answer) for the incomplete triple $\langle h,\ r,\ ?\rangle$:

$$t = \underset{e \in \mathcal{E}}{\operatorname{argmax}}\, \phi(\mathbf{h}, \mathbf{r}, \mathbf{e}). \tag{1}$$

Head predictions can be defined analogously. For the sake of simplicity, in the following we mostly refer to tail predictions; all our methods and algorithms can be applied to head predictions as well.

Research on LP is typically run on datasets sampled from real-world KGs; in each dataset the set of facts $\mathcal{G}$ is further split into a training set $\mathcal{G}_{train}$, a validation set $\mathcal{G}_{valid}$, and a test set $\mathcal{G}_{test}$. Evaluation is performed by running head and tail prediction on each fact in $\mathcal{G}_{test}$: for each prediction the entity actually featured in the test fact, that we dub *target entity*, is ranked against all the other entities in $\mathcal{E}$. The equation for the tail rank of any fact is thus:

$$tailRank(h, r, t) = |\{e \in \mathcal{E} | \phi(\mathbf{h}, \mathbf{r}, \mathbf{e}) >= \phi(\mathbf{h}, \mathbf{r}, \mathbf{t})\}| \tag{2}$$

[1] https://github.com/AndRossi/Kelpie

Head ranks are computed analogously. Ideally, the target entity should obtain the best plausibility, so its rank should be 1.

In a prediction, multiple entities may be "correct" answers, i.e., multiple entities, if used to answer the prediction, may result in a fact in $\mathcal{G}$. This leads to two possible settings:

- *raw setting*: correct answers outscoring the target entity are still deemed wrong, so they are treated as any other entity;
- *filtered setting*: correct answers outscoring the target entity are not seen as mistakes and do not contribute to its rank.

Filtered metrics are generally preferred in literature [9]; in our work we always use filtered metrics.

LP models are usually evaluated by aggregating the head and tail ranks obtained on $\mathcal{G}_{test}$ into global metrics:

- *Hits@K (H@K)*: it is the fraction of ranks T with value $\leq k$:
$$H@K = \frac{|\{t \in T : t \leq k\}|}{|T|}. \tag{3}$$
Kadlec *et al.* [23] have recently suggested to focus on $H@1$, as it better highlights the differences among models.
- *Mean Reciprocal Rank (MRR)*: it corresponds to the average of the inverse of all the obtained ranks T:
$$MRR = \frac{1}{|P|}\sum_{t \in T}\frac{1}{T}. \tag{4}$$

Both metrics are always between 0 and 1, and the higher their value, the better the result they convey.

2.2 Explaining Link Predictions

Given a tail prediction $\langle h, r, t\rangle$, a Kelpie explanation consists intuitively in the smallest set of training facts featuring h that have enabled us to predict the tail t. For instance, when explaining why the top ranking tail for $\langle Barack\_Obama, nationality, ?\rangle$ is USA, Kelpie searches for the smallest set of *Barack_Obama* facts allowing the model to predict USA. Analogously, to explain a head prediction $\langle h, r, t\rangle$ we search for the most relevant combination of t facts. Under this broad definition we define the following two scenarios:

Necessary Explanations. Given a tail prediction $\langle h, r, t\rangle$ we define a *necessary explanation* as the smallest set of training facts featuring h that, if removed from $\mathcal{G}_{train}$, lead the model to change the top ranking prediction for $\langle h, r, ?\rangle$ to any entity $e \neq t$. In other words, a necessary explanation is the smallest set of facts featuring h that made possible for the model to pick the correct tail t in the first place. Analogous explanations can be extracted for head.

Sufficient Explanations. Given a tail prediction $\langle h, r, t\rangle$ and a set C of random entities for which the model does not predict $\langle c, r, t\rangle$, $c \in C$, we define a *sufficient explanation* as the smallest set of training facts featuring h that, when added to any $c \in C$, lead the model to switch the top ranking tail for $\langle c, r, ?\rangle$ to t. In other words, a sufficient explanation contains the h training facts that, if added to any $c \in C$ by replacing h with c, lead the model to switch its tail prediction for c to the same tail t predicted for h. The formulation for head predictions is analogous. We call the prediction switch undergone by entities $c \in C$ a *conversion*. For example, when explaining the tail prediction ⟨*Barack_Obama*, *nationality*, *USA*⟩, we identify the *Barack_Obama* training facts that, if transferred to other entities, make the model predict them as American. For example, we may find that adding the fact ⟨*Xi_Jinping*, *president_of*, *USA*⟩ to $\mathcal{G}_{train}$ is enough to change the predicted *nationality* of *Xi_Jinping* to *USA*, accomplishing the conversion. Despite having a global scope in regard to the entities c to convert, sufficient explanations are still a *local* approach: they only take into account training facts featuring h and thus only identify which parts of h enable the prediction to explain.

As observed by Watson, Gultchin *et al.* [57], necessity and sufficiency are the building blocks of any successful explanation. Necessary and sufficient explanations are complementary to each other: given a source entity, a relation and the predicted target entity:

- *necessary explanations* provide a deeper insight on *the source entity* with respect to that prediction: they investigate which, among the training facts of the source entity, lead to the prediction of the target entity;
- *sufficient explanations* provide a deeper insight on *the prediction* with respect to that source entity: they investigate how the same prediction can be replicated across the entire dataset using the facts of the source entity.

Kelpie can successfully extract *both* necessary and sufficient explanations. The type of explanation to choose depends on the user's goal: necessary explanations can find why a specific entity has been predicted in a certain way, whereas sufficient explanations embody rules that imply that prediction, describing the behaviour of the model on a broader scale. Both scenarios can also be leveraged to explain wrong predictions: necessary explanations can identify which training facts of the wrongly predicted entities have misled the model; sufficient explanations can isolate which facts those entities may have lacked, i.e., which facts, if transferred to them from other entities, can convert them to the correct prediction.

We indicate with X the generic *candidate explanation* and with X^* the explanation we identify and return. When discussing the *necessary* scenario we denote X as X_n and X^* as X_n^*; in the *sufficient* scenario we denote them as X_s and X_s^* respectively.

3 RELATED WORKS

Among post-hoc explanation methods, the works most related to ours form two main categories: *general purpose* and *LP-specific.*

3.1 General Purpose Frameworks

General Purpose XAI frameworks propose techniques that can be applied in a variety of domains; among the most popular systems in this family, we discuss LIME [41], SHAP [30] and ANCHOR [42].

The LIME framework [41] perturbs the input features of the predicted sample, and checks the effects on the prediction confidence. It then uses Lasso regression [51] to obtain an interpretable local approximation of the original model, with a relevance weight for each feature. Identifying which features of a sample have led to a certain prediction corresponds to our concept of *necessary explanation.*

Recently, LIME seems to have been outclassed by frameworks that formulate the relevance of input features in terms of Shapley values [46]. As observed by Watson, Gultchin *et al.* [57] these methods, such as SHAP [30], have gained popularity due to their solid theoretical backing derived from Game Theory. We provide a detailed comparison between SHAP and our approach in Section 4.3.

The ANCHOR framework [42] explains a prediction by identifying *anchors*, i.e., sets of input features that, if transferred to other samples, "lock" the model to yield the same prediction. Anchors are closely related to the *sufficient explanations* we use in our work: if a sample displays the features of the anchor, the model will return the same prediction, regardless of what the other features look like.

The aforementioned frameworks identify *saliency* explanations, i.e., the most relevant features from the input samples [3]. They have been successfully adapted to many vertical scenarios: in the case of Entity Matching (EM), for example, Mojito [11] works as a direct extension of LIME, adding a new type of perturbation in which an attribute of one of the two entries to compare is copied into the other entry. Analogously, Landmark [5] extends the LIME-like post-hoc approach with the idea of introducing perturbations to just one entry at a time, keeping the other as a term for comparison.

Unfortunately, those frameworks cannot be similarly adapted to the LP task. Saliency explanations require the input features to be human-interpretable, otherwise feature-based explanations will not be understandable by humans. While this requirement is met when dealing with images, sentences, or even pairs of entries as in EM, it does not apply to LP, where the input samples are just triplets of embeddings representing the head, the relation, and the tail of the fact to score. In our scenario, saliency-based approaches would just identify which components in those vectors are most relevant to the outcome to explain, which is not truly informative from a human point of view. Kelpie, on the contrary, is specifically tailored for embedding-based LP: when explaining a prediction, rather than focusing on the input features, it identifies which *training facts* have been most responsible for that prediction in the first place.

In this regard Kelpie is more akin to the framework by Koh and Liang [25], which identifies the most influential training samples for the prediction to explain. Their approach is not based on local perturbations, but rather on Influence Functions, a classic technique from robust statistics [19]. Unfortunately, this makes the algorithm computationally very expensive; despite the authors' efforts of optimization, Pezeshkpour *et al.* [40] have observed that computation times degrade rapidly when the number of entities in the dataset exceeds even a few hundreds. Ultimately, this makes the framework impractical for explaining LP on KGs.

3.2 Link Prediction Specific Tools

Despite the extensive body of literature regarding LP models for KGs, so far only a few works have dealt with their explainability.

The most recent approaches in this regard follow a *data poisoning* technique: given a prediction $\langle h, r, t\rangle$, their purpose is to identify the one fact that, if added or removed to the training set, worsens $\phi(h, r, t)$ the most (thus "poisoning" the prediction).

The Criage framework [40] achieves this goal by approximating the variation in $\phi(h, r, t)$ with Influence Functions, similarly to the above mentioned work by Koh and Liang [25]. Criage successfully overcomes the computational issues that plagued its predecessors by applying first order Taylor approximations. Unfortunately, it is unclear how this formulation can be consistently adapted to the scoring functions of non-multiplicative models; another major limitation is that it can only take into account facts whose tail is either the head h or the tail t of the fact to explain.

The framework by Zhang *et al.* [62] performs a similar task by following an embedding perturbation approach. They slightly shift the embedding of h towards $-\frac{\partial\phi(h, r, t)}{\partial h}$, i.e., in the direction that would worsen $\phi(h, r, t)$; they then identify the training facts featuring h whose score worsens the most when using the shifted version of h. These facts agree with $\langle h, r, t\rangle$ on how to improve or worsen ϕ, so they are assumed to work in its favour during training. A symmetric approach can be used to identify the fake adversarial samples that, if added to the dataset, worsen $\phi(h, r, t)$ the most.

While still in the spectrum of interpretability techniques, data poisoning frameworks do not aim at explaining predictions, but rather at investigating the robustness of models to single-fact adversarial modifications. Nonetheless, the way in which they identify the one most influential sample to remove from the dataset can be seen as analogous to our concept of necessary explanations.

When it comes to full-fledged explainability tools, though, LP models can rely on such diverse architectures that researching model-independent techniques is a very challenging task. Authors have thus tried to circumvent the issue in a number of creative ways. For instance, Zhang *et al.* [64] extract explanations using the dataset topology rather than the model behaviour: to explain a prediction $\langle h, r, t\rangle$ they identify the most relevant paths connecting h to t, i.e., the paths that most frequently connect other entities linked by the same relation r. In their approach the model is only used to filter out relations and entities with embeddings too different from the ones of the fact to explain. Other authors have developed inherently interpretable LP models based on embeddings, such as XTransE [63] or the recent model by Bhowmik and De Melo [6]. We also acknowledge the existence of tools that only support models based on a specific architecture, such as GNNExplainer [60], that is only suitable for Graph Neural Networks systems.

4 THE KELPIE FRAMEWORK

The Kelpie framework can be applied by design to LP models based on embeddings. As already mentioned, given any prediction Kelpie can identify the smallest set of training facts enabling it, both in the sufficient and in the necessary scenario.

We find that extracting combinations of facts longer than 1 is often vital for effective explanations. For instance, when extracting necessary explanations for the tail prediction $\langle Barack\_Obama, nationality, USA\rangle$, removing only the one most influential training fact featuring $Barack\_Obama$, e.g., $\langle Barack\_Obama, president\_of, USA\rangle$, will likely not affect the prediction, because it is still enabled by other facts, e.g., $\langle Barack\_Obama, part\_of, 109^{th}\_US\_Congress\rangle$. Models can yield predictions by leveraging multiple pieces of evidence: a correct explanation should encompass them all.

The high-level architecture of the Kelpie framework, as depicted in Figure 1, is based on the interaction of three main modules: a *Pre-Filter*, a *Relevance Engine* and an *Explanation Builder*. In the following we describe them in the context of explaining a tail prediction $\langle h, r, t\rangle$; head predictions can be handled analogously.

When explaining a tail prediction $\langle h, r, t\rangle$, the Pre-Filter analyzes the set of all training facts featuring h: we call it $\mathcal{G}^h_{train}$. The Pre-Filter aims at reducing the search space for the following steps, so it discards the least promising facts from $\mathcal{G}^h_{train}$; we denote the resulting set as $\mathcal{F}^h_{train}$. We discuss this component in Section 4.1.

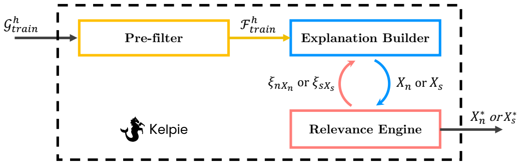


Figure 1: High-level architecture of the Kelpie framework, depicting the interactions to explain a tail prediction ⟨*h*, *r*, *t*⟩.

The Explanation Builder combines the Pre-Filtered facts to build candidate explanations X; it then explores the resulting space searching for the smallest combination impactful enough on the prediction to explain, i.e., the correct explanation X^*. The Explanation Builder analyzes increasingly long combinations with an *ad-hoc* algorithm described in Section 4.3. Each analyzed X is submitted to the Relevance Engine, which estimates its effect on the prediction to explain. The process goes on until either a candidate explanation X is accepted as X^*, or a size limit is exceeded. If multiple same-sized X satisfy the acceptance criteria, they all adhere to the definitions in Section 2, so we just return the first one we have encountered.

The Relevance Engine can estimate how adding or removing training facts from an entity would affect a specific prediction. We call the estimated effect *relevance*; in our work we provide distinct formulations for *necessary relevance*, denoted with ξ_{nX_n}, and *sufficient relevance*, denoted with ξ_{sX_s}. Our implementation of Relevance Engine relies on a ML technique that we dub *post-training*; we describe it in detail in Section 4.2.

4.1 Pre-Filter

Given a tail prediction $\langle h,\ r,\ t\rangle$ the Pre-Filter has the role to preemptively discard the least promising facts in $\mathcal{G}^h_{train}$ to limit the space of candidate explanations to combinations of presumably meaningful h facts. It has been observed that in all LP datasets the distribution of facts per entity is extremely skewed [45]; Pre-Filtering prevents combinatorial explosion when the *degree* of h, i.e., the number of its training facts, is very large (e.g, from hundreds to thousands).

To achieve this goal the Pre-Filter computes, for each fact in $\mathcal{G}^h_{train}$, a *promisingness* value γ. Our implementation measures promisingness based on the graph topology. In a KG, topologically close entities bear a stronger semantic relationship: intuitively, a training fact connecting h to an entity q close to t has a higher chance to be meaningful with respect to $\langle h,\ r,\ t\rangle$. Therefore, for any $\langle h,\ s,\ q\rangle$ (or $\langle q,\ s,\ h\rangle$) in $\mathcal{G}^h_{train}$ we compute $\gamma_{\langle h,\ s,\ q\rangle}$ (or $\gamma_{\langle q,\ s,\ h\rangle}$) as the length of the shortest non-oriented path connecting q to t; lower values convey better promisingness and thus higher priority in the filtering selection. We provide an example in Figure 2 explaining tail prediction ⟨*Barack_Obama*, *nationality*, *USA*⟩. We analyze all *Barack_Obama* training facts:

- ⟨*Barack_Obama*, *president_of*, *USA*⟩ has promisingness 0 (the best possible value), because it features the tail entity *USA* itself.
- ⟨*Barack_Obama*, *born_in*, *Honolulu*⟩ has promisingness 1, because there is a fact linking directly *Honolulu* and *USA*.
- ⟨*Bill_Gates*, *supported*, *Barack_Obama*⟩ has promisingness 2, as the shortest path from *Bill_Gates* to *USA* has length 2: [⟨*Bill_Gates*, *born_in*, *Seattle*⟩, ⟨*Seattle*, *located_in*, *USA*⟩].

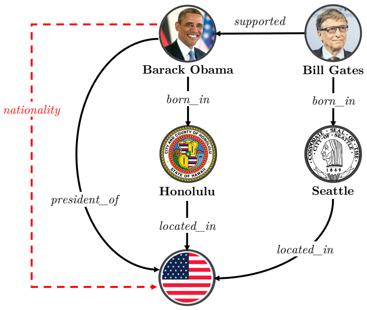


Figure 2: An example of a small KG; the tail prediction ⟨Barack_Obama, nationality, USA⟩ is highlighted in red.

We identify the shortest path from any entity q to t by operating Breadth-First Searches (BFS) on the training graph using q as starting node. We do not take into account fact orientation, and we ignore any paths including $\langle h,\ r,\ t\rangle$ as we want to measure the closeness of q independently of the prediction to explain.

Another possibility to compute promisingness, inspired by the works of Shiralkar, Ciampiglia *et al.* [10, 47], would be to prioritize the facts $\langle h,\ r,\ q\rangle$ in which q has a similar type to t. We have verified that this option yields results similar to our original formulation; we report our experiments in this regard in our online repository.

Independently of how promisingness is measured, we build the set $\mathcal{F}^h_{train}$ of the most promising facts applying a simple *top-k* policy on the promisingness values. The value of k can be tweaked depending on the desired trade-off between the computation time and the certainty to keep all the meaningful facts. In our end-to-end experiments we use $k = 20$, that we have verified to be a fine trade-off; we report tests varying the value of k in our repository.

4.2 Relevance Engine

The Relevance Engine has the responsibility to estimate how the addition or removal of certain training facts would affect a specific prediction if the model was retrained from scratch; we call this estimate *relevance*. Considering as usual a generic tail prediction $\langle h,\ r,\ t\rangle$, the role of the Relevance Engine is to receive from the Explanation Builder candidate explanations X composed of facts from $\mathcal{F}^h_{train}$ and to return the corresponding relevances. The necessary and sufficient scenarios require different formulations of relevance:

- in the necessary scenario, relevance ξ_{nX_n} quantifies the effect of removing from h the facts in X_n;
- in the sufficient scenario, relevance ξ_{sX_s} measures the effect of adding the facts in X_s to a set C of entities c to convert.

Ideally, relevance could be computed by retraining the whole model from scratch after adding or removing the facts in X. In practice, this is clearly unfeasible. We thus design a more scalable methodology that we call *post-training*. Post-training consists in adding a new entity to an already trained model; the corresponding embedding is trained on a set of purposefully chosen facts while keeping all the other embeddings and parameters frozen. For any entity e, running a post-training using a replica of $\mathcal{G}^e_{train}$ results in an alternate variant of e that we dub a *mimic*. Post-training a mimic is a lightweight process, because it optimizes only one embedding (instead of $|\mathcal{E} + \mathcal{R}|$) on a training set comparable to $\mathcal{G}^{e}_{train}$ (instead of the entire $\mathcal{G}_{train}$, which is always orders of magnitude larger). We identify two types of mimics:

- A *homologous mimic* of e approximates the behaviour of e without introducing variations;
- A *non-homologous mimic* of e approximates the behaviour that e would show if its training facts had been slightly different since the beginning.

A *homologous mimic* is created as a new, fictitious entity e' with a set of training facts $\mathcal{G}^{e'}_{train}$ that is an exact replica of $\mathcal{G}^{e}_{train}$. Its embedding $\boldsymbol{e'}$ is initialized randomly (as for any entity), and then post-trained on $\mathcal{G}^{e'}_{train}$. After this process is over e' is expected behave similarly to e. This is a safe assumption because, as described in Section 2, KG Embeddings are trained to optimize the ϕ scores of the training facts mentioning them. This implies that the embedding of any entity is only directly influenced by *(i)* the training samples mentioning it, *(ii)* the embeddings of the neighbouring KG elements, and *(iii)* any potential shared parameters trained alongside said embeddings. Since the samples mentioning e' are the same as those mentioning e, and since the other embeddings and parameters have remained unchanged, we can expect e' to be a good proxy for e. We do not expect from e' an identical behaviour to e: fluctuations may occur, e.g., due to the inherent randomness of training steps such as embedding initialization and random shuffle of training samples. What we are after is just an approximation of the behaviour of e.

A *non-homologous mimic* is created following a similar pipeline, but after initializing $\mathcal{G}^{e'}_{train}$ as a replica of $\mathcal{G}^{e}_{train}$, a specific variation is purposefully injected before post-training e'. The variation can consist in either removing or adding a set M of facts; we denote the mimic as e'_{-M} in the former case, or e'_{+M} in the latter. After the post-training is over, the mimic should approximate the behaviour that the original entity e would have displayed if the injected variation had been present since the very beginning.

We provide in Figure 3 an example of homologous and non-homologous mimics. Given the same ⟨*Barack_Obama*, *nationality*, *USA*⟩ example as in Section 4.1, a homologous *Barack_Obama* mimic will be featured in the same facts as the original; on the contrary, a non-homologous mimic will display variations, such as the removal of the link with relation *president_of* and tail *USA*.

Post-training may remind of the popular technique of *fine-tuning*, as they both operate on a model after its original training is complete; however, while fine-tuning updates most (or all) the parameters in the model, typically to adapt it to a new scenario, post-training just extends the set of KG elements represented by the model without modifying any of the pre-existing parameters.

Intuitively, given a prediction involving an entity e we can estimate the effect of adding (or removing) to e a set M of facts by comparing the outcome obtained using a non-homologous mimic e'_{-M} (or e'_{+M}) with the outcome of the original e. In practice, we find it more effective to compare the result of the non-homologous mimic with the result of a homologous mimic e': this seems to erase small fluctuations that post-training may introduce in some cases, e.g., when e has an exceedingly low degree, or when its embedding had not fully converged in the original training. More specifically, given a tail prediction $\langle h,\ r,\ t\rangle$ and a candidate explanation X containing facts from $\mathcal{G}^{h}_{train}$ we compute relevance as follows.

```
Algorithm 1: Compute necessary relevance $\xi_{nX_n}$
   Input: A tail prediction $\langle h,\ r,\ t\rangle$ to explain;
          a candidate necessary explanation $X_n$
   Output: The corresponding necessary relevance $\xi_{nX_n}$
 1 $\mathrm{h}' \leftarrow postTrain(\mathcal{G}^{h}_{train})$
 2 $\mathrm{h}'_{-\mathrm{X_n}} \leftarrow postTrain(\mathcal{G}^{h}_{train} \setminus X_n)$
 3 $tailRank^{t}_{h'} \leftarrow predictTail(\mathrm{h}', \mathrm{r}, \mathrm{t})$
 4 $tailRank^{t}_{h'_{-X_n}} \leftarrow predictTail(\mathrm{h}'_{-\mathrm{X_n}}, \mathrm{r}, \mathrm{t})$
 5 $\xi_{nX_n} \leftarrow tailRank^{t}_{h'_{-X_n}} - tailRank^{t}_{h'}$
 6 return $\xi_{nX_n}$
```

```
Algorithm 2: Compute sufficient relevance $\xi_{sX_s}$
   Input: A tail prediction $\langle h,\ r,\ t\rangle$ to explain;
          a candidate sufficient explanation $X_s$;
          a set $C$ of entities to convert
   Output: The corresponding sufficient relevance $\xi_{sX_s}$
 1 $individualRelevances \leftarrow [\,]$
 2 for $c \in C$ do
 3     $\mathrm{c}' \leftarrow postTrain(\mathcal{G}^{c}_{train})$
 4     $\mathrm{c}'_{+\mathrm{X_s}} \leftarrow postTrain(\mathcal{G}^{c}_{train} \cup X_s)$
 5     $tailRank^{t}_{c'} \leftarrow predictTail(\mathrm{c}', \mathrm{r}, \mathrm{t})$
 6     $tailRank^{t}_{c'_{+X_s}} \leftarrow predictTail(\mathrm{c}'_{+\mathrm{X_s}}, \mathrm{r}, \mathrm{t})$
 7     $current\xi_{sX_s} \leftarrow (tailRank^{t}_{c'} - tailRank^{t}_{c'_{+X_s}})/(tailRank^{t}_{c'} - 1)$
 8     $individualRelevances.insert(current\xi_{sX_s})$
 9 $\xi_{sX_s} \leftarrow average(individualRelevances)$
10 return $\xi_{sX_s}$
```

Necessary relevance. We define the *necessary relevance* ξ_{nX_n} of X_n as the expected tail rank deterioration caused by the removal of the facts in X_n from h. We report our procedure in Algorithm 1. We first post-train a homologous mimic h' and a non-homologous mimic h'_{-X_n} (lines 1-2). We then compute the tail ranks of $\langle h',\ r,\ t\rangle$ and $\langle h'_{-X_n},\ r,\ t\rangle$ (lines 3-4): their difference is the estimated tail rank deterioration ξ_{nX_n} (line 5). ξ_{nX_n} is always between 0, indicating no expected tail rank variations, and $|\mathcal{E}| - 1$, indicating that the original tail t is expected to become the least predicted entity.

Sufficient relevance. In the sufficient scenario we first identify a set C of random entities c for which the tail prediction $\langle c,\ r,\ t\rangle$ has rank greater than 1: for any $c \in C$, a sufficient explanation should *convert* c, i.e., make t become the top-ranking tail for c. We compute the *sufficient relevance* ξ_{sX_s} of any X_s as shown in Algorithm 2. For each $c \in C$ we estimate how adding the facts of X_s to c improves the rank of t as a tail by: (*i*) post-training a homologous mimic c' and a non-homologous one c'_{+X_s} (Lines 3-4); (*ii*) comparing the tail ranks of $\langle c'\ r\ t\rangle$ and $\langle c'_{+X_s}, r, t\rangle$ (Lines 5-6): their difference is the *actual rank improvement* associated to X_s for c. Unlike necessary explanations, for which any rank worsening is acceptable, sufficient explanations have a specific goal: converting all $c \in C$. We thus formulate the effect of adding the X_s facts to any $c \in C$ as the *ratio* of the *actual rank improvement* over the *ideal rank improvement*, which we know to be $tailRank(c', r, t) - 1$ (line 7). The sufficient relevance ξ_{sX_s} is obtained by averaging such rank improvement ratios across all $c \in C$ (line 9). ξ_{sX_s} is typically between 0 (if X_s is not expected to produce any effects) and 1 (if X is expected to fully convert any $c \in C$); exceptionally, it can be lesser than 0 if X_s is expected to worsen ranks rather than to improve them.

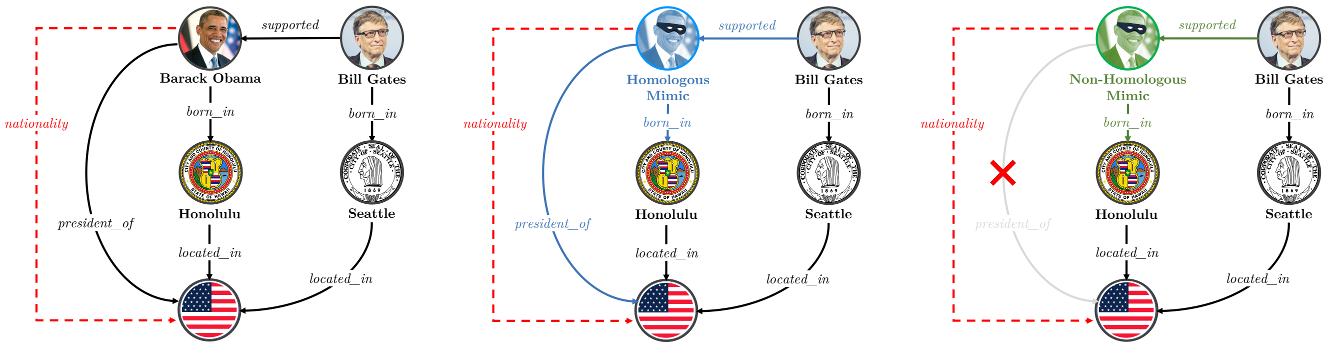


Figure 3: An example of homologous and non-homologous mimics.

Relying on post-training grants to our Relevance Engine some very desirable properties. Pre-existing approaches, such as those mentioned in Section 3, can only estimate the significance of *single facts*. On the contrary, post-training successfully estimates the effects of adding or removing *multiple facts*, enabling Kelpie to extract more expressive explanations. To the best of our knowledge, our technique is the first attempt in LP literature to effectively compute the consequences of adding or removing multiple training facts.

The Relevance Engine is the one Kelpie module that requires awareness over the original model, as its post-training processes take directly into account the underlying embedding mechanism. The other Kelpie modules, on the contrary, are completely model-independent. Treating the model as a transparent box, incidentally, is a trait shared by other LP-interpreting works, such as those mentioned in Section 3.2. We find that, in the case of Kelpie, the best way to add support to new models is to provide model-specific implementations of simple interfaces; we include a detailed technical description in this regard in our online repository.

In describing the post-training process, we have assumed that the original model only leverages individual facts, which is by far the dominant approach in literature. A few recent methods can also exploit contextual information, such as paths [18, 55], temporal details [29], or types [59]. While our current implementation focuses on fact-based models, the formulation of Kelpie can indeed be applied to these *contextual models* too: the Pre-Filter and the Explanation Builder, which are model-independent, would just work as usual, and the Relevance Engine could easily include contextual information in its post-training processes. We further discuss the opportunities of contextual models in Section 6.

4.3 Explanation Builder

Given the usual tail prediction $\langle h, r, t\rangle$ to explain, the role of the Explanation Builder is to guide the search for its explanation X^*. This amounts to: *(i)* combining the Pre-Filtered $\mathcal{F}^h_{train}$ into candidate explanations X; *(ii)* exploring the resulting space, selecting the X to submit to the Relevance Engine; *(iii)* deciding, based on the obtained relevances, whether any X can be accepted as X^*.

Since Kelpie supports combinations of facts longer than 1, the space of candidate explanations can rapidly become overwhelming: even Pre-Filtering the h facts down to the n most promising ones, the number of i-sized combinations is still $\binom{n}{i}$, making brute-force approaches unfeasible. We thus model the identification of X^* as a search problem in the space S of the candidate explanations X sized between 1 and a maximum value i_{max}; in our experiments we set i_{max} to 4, having observed that longer explanations hardly ever provide meaningful contributions to the process. Rather than exploring S as a whole, we partition it into subsets S_i based on the size i of candidate explanations. We treat each S_i as the space of a separate search problem, that we tackle with an algorithm inspired by Adaptive Simulated Annealing [22]. As soon as we find an X satisfying specific acceptance criteria the search is interrupted and we return X as X^*. On the contrary, if evidence is found that no X in S_i satisfies the acceptance criteria, we prematurely stop exploring S_i and we move to S_{i+1}. If the explorations of all S_i up to $S_{i_{max}}$ are unsuccessful, we follow a best-effort policy and return the X with the highest relevance found so far. In the following we describe in detail our acceptance criteria and visit algorithm.

Acceptance Criteria. As described in Section 4.2, the relevance of any candidate explanation X is an estimate of how the prediction would change when applying X to $\mathcal{G}^h_{train}$. For any analyzed X, we can thus use its relevance to decide whether X can be accepted as X^* or not. In practice we set *relevance thresholds*:

- In the *sufficient scenario* our ideal goal would be to achieve, for all entities $c \in C$, the ideal rank improvement, corresponding to $\xi_{sX_s} = 1$. In our experiments we accept room for some approximation and set the threshold ξ_{s0} to 0.9, indicating an expected tail rank improvement of 90% or greater.
- In the *necessary scenario* we just aim at worsening the prediction rank by 1 or more. Hence, the threshold ξ_{n0} can be tweaked depending on the desired effect: higher values can lead more effective explanations at the cost of longer computations. We report in our repository a study on how varying our ξ_{n0} affects explanations; we find that $\xi_{n0} = 5$ is usually a fine trade-off.

Visit Algorithm. As mentioned above, in the search for X^* we explore the subsets S_i of the space S separately. Within any S_i, identifying the most relevant X is a problem of *combinatorial optimization*, a well-known topic addressed by algorithms such as Simulated Annealing, Tabu Search, Evolutionary methods [39], or multi-armed bandit solutions [24] (e.g, in the ANCHOR framework). Unfortunately, such general-purpose solutions tend to require a large number of iterations to converge, as they are designed for scenarios where no assumptions can be made on the composition of the search space. This makes them impractical in our case, where visiting a candidate explanation X involves advanced analyses.

To overcome this issue we observe that relevant explanations are usually combinations of facts that, when used as 1-sized explanations, are individually relevant. For any X we can thus compute a *preliminary relevance* as the average of the individual relevances of its facts. In our experiments, we have observed that preliminary and true relevance do globally correlate: high preliminary relevance usually implies high relevance and *vice versa*. We report in Figure 4 an example for the candidate sufficient explanations of a TransE FB15k prediction; we have found analogous pattern in different predictions and settings as well. We stress that the X with the highest preliminary relevance is not necessarily the one with highest relevance. Furthermore, a longer X is not necessarily more relevant than a shorter one, proving that it not enough to just combine more facts to achieve more effective explanations. These observations, mixed with inspiration from the Adaptive Simulated Annealing technique [22], result in the iterative method in Algorithm 3.

We first compute the individual relevance of each fact in $\mathcal{F}^h_{train}$ when used as a 1-sized explanation (Lines 1-3). Before exploring any S_i with $i > 1$, we compute the *preliminary relevance* of each $X \in S_i$ as the average relevance of its facts. (Lines 7-9) We then traverse S_i by scanning its candidate explanations X in descending order of preliminary relevance; for each visited X we have the Relevance Engine compute its true relevance (Line 13), and we check if it satisfies the acceptance criteria (Line 14). If it does, we stop and return X as X^*; otherwise, we need to decide whether S_i is still worth-exploring, or we should rather move to S_{i+1}.

To make this decision, we need to assess how likely it is to have already met the most relevant X in S_i. As a simple proxy to such likeliness we use the ratio ρ_i between the relevance of the current X and the highest relevance encountered so far in S_i (Line 19).[2] The value of ρ_i is always in $[0, 1]$ as long as relevances are positive; as the iterations go on, due to the correlation between preliminary and true relevance the current X will tend to become less and less relevant, inevitably making ρ_i smaller and smaller. After a while, it may become reasonable to stop exploring S_i and move to S_{i+1}. We decide this stochastically, using $1 - \rho_i$ as the probability to interrupt the search in S_i and to move to S_{i+1} (Lines 20-21). Probabilistic solutions are common in search algorithms due to their flexibility [49]; in our case, we observe that among different models, datasets, or even just different subsets S_i for the same prediction, the relevances of the visited candidate explanations may decay with very different paces; hence other strategies, e.g., fixed thresholds or top-k, do not work well compared to a probabilistic approach.

Comparison With Shapley Value Methods. The Relevance Engine, with its post-training technique, allows us to add or remove from $\mathcal{G}_{train}$ the facts of candidate explanations in a way similar to how saliency-based XAI frameworks inject perturbations into the input features of samples; the Explanation Builder guides the exploration in the vast space of resulting candidate explanations.

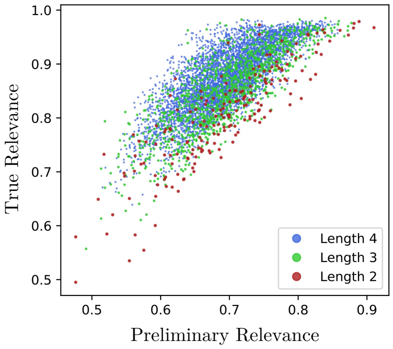


Figure 4: Preliminary vs true relevance correlation: each point is a candidate explanation for the same prediction.

Algorithm 3: Explanation Builder algorithm for identifying the smallest valid explanation X^*

```
Input: The set $\mathcal{F}^h_{train}$ of training samples to combine into explanations;
       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  $fact2rel \leftarrow \{\}$
2  for $fact \in \mathcal{F}^h_{train}$ do
3      $fact2rel[fact] \leftarrow engine.computeFor([fact])$
4  for $i \leftarrow 2$ to $i_{max}$ do
5      $S_i \leftarrow combinations(\mathcal{F}^h_{train}, i)$
6      $preRelevances \leftarrow []$
7      for $X \in S_i$ do
8          $preRelevance \leftarrow avg([fact2rel[f]$ for $f$ in $X])$
9          $preRelevances.add(preRelevance)$
10     $S_i \leftarrow sort(S_i, preRelevances)$
11     $bestRelevance \leftarrow None$
12     for $X \in S_i$ do
13         $curRelevance \leftarrow engine.computeFor(X)$
14         if $curRelevance > \xi_0$ then
15             return $X$
16         if $bestRelevance == None$ or
             $curRelevance > bestRelevance$ then
17             $bestRelevance \leftarrow curRelevance$
18         else
19             $\rho_i \leftarrow curRelevance / bestRelevance$
20             if $random(0, 1) > \rho_i$ then
21                 break
```

As long as the Relevance Engine is employed, it is technically possible to replace the Explanation Builder with the exploration methods of other XAI frameworks. In this regard, we have tested the SHAP framework strategy [30], appreciated for its theoretical guarantees. As mentioned in Section 3.1, SHAP approximates Shapley Values [46] to identify the most relevant input features.

We run tests with KernelSHAP, i.e., the one model-agnostic method introduced by the SHAP authors, using their own implementation.[3] We find that, to explain the same predictions, while our Explanation Builder only takes a few dozens or hundreds visits in the space of candidate explanations (i.e., post-trainings), KernelSHAP always requires hundreds of thousands, making it unfeasible for our task; we report detailed results of these experiments in our repository. Our results confirm the observations of recent works raising concerns on the tractability of SHAP in specific domains [12]. Our Explanation Builder, in contrast, is remarkably efficient. This is mostly due to its preliminary relevance heuristics,

[2] In practice, to achieve greater robustness to outliers, in our experiments we use the average relevance of the last 10 visited candidate explanations

[3] https://github.com/slundberg/shap

	Entities	Relations	Train Facts	Valid Facts	Test Facts
FB15k	14951	1345	483142	50000	50971
FB15k-237	40943	18	141442	5000	5000
WN18	14541	237	272115	17535	20466
WN18RR	40943	11	86835	3034	3134
YAGO3-10	123182	37	1079040	5000	5000

Table 1: Statistics of the LP datasets we employ.

	FB15k		**WN18**		**FB15k-237**		**WN18RR**		**YAGO3-10**	
	H@1	**MRR**	**H@1**	**MRR**	**H@1**	**MRR**	**H@1**	**MRR**	**H@1**	**MRR**
TransE	0.259	0.401	0.371	0.553	0.134	0.208	0.035	0.181	0.066	0.121
ComplEx	0.826	0.857	0.944	0.950	0.270	0.365	0.440	0.484	0.499	0.575
ConvE	0.612	0.703	0.941	0.945	0.225	0.311	0.392	0.424	0.466	0.544

Table 2: LP performance of the models we employ across all datasets, using metrics H@1 and MRR.

tailored specifically for the LP scenario, which allow us to start the search from the facts that will most probably produce the best explanations, and to enact effective early termination policies.

We also point out that most Shapley Values approaches assume feature independence, which may not be always guaranteed in the LP context. To the best of our knowledge, the work by Aas *et al.* [1] is the only SHAP version supporting dependent features; unfortunately, as the authors themselves report, this comes at a significant cost in computation times, which would make it even worse in LP.

5 EXPERIMENTAL RESULTS

In this section, we describe in detail the experiments run in our work. We first discuss our environment and general setup, and then report the performed experiments and the corresponding results.

5.1 Experimental Setup

We provide here an overview of the computational environment of our experiments, as well as the involved LP datasets and models.

Environment. All of our experiments, including the original trainings and evaluations of our models, have been run on a server with 88 CPUs Intel Core(TM) i7-3820 at 3.60GHz, 516GB RAM and 4 NVIDIA Tesla with 16GB VRAM. The operating system is Ubuntu 18.04, with CUDA Version 11.2 (Driver 460.73.01) and PyTorch 1.7.1.

Datasets. We evaluate the Kelpie framework conducting experiments on the 5 best-established datasets in LP literature; we report their main features in Table 1. **FB15k** and **WN18** have been created by the TransE authors [9] focusing on the most mentioned entities in the Freebase [7] and WordNet [33] KGs respectively. These datasets have been proven to suffer from test leakage due to the presence of inverse and equivalent relations; they are thus generally coupled with their subsamples **FB15k-237** and **WN18RR**, created respectively by Toutanova and Chen [52] and by Dettmers *et al.* [13] removing such relations. Finally, **YAGO3-10** has been sampled by Dettmers *et al.* [13] from the YAGO3 [31] KG extracting the facts that mention entities linked by at least 10 different relations.

Models. As already mentioned, the Kelpie framework supports any LP model based on embeddings. To showcase its flexibility we run our experiments on three models representative for the three different families recently identified by Rossi *et al.* [43] in their taxonomy, and trained with three different Loss functions.

- **TransE** [9] is a pioneering *geometric* model that interprets relations as translations in the embedding space; its simple formulation cannot model correctly one-to-many and many-to-one relations [15]. Our implementation is faithful to the original paper, and optimized with *Pairwise Ranking Loss.*
- **ComplEx** [54] is a *tensor decomposition* model that learns embeddings in the $\mathbb{C}$ space. Its scoring function is a Hermitian product, allowing ComplEx to model asymmetric relations. Our implementation is based on the one by Lacroix *et al.* [27], which has been recently shown to surpass most LP models [43]; it is trained with a *Multiclass Negative Log-Likelihood Loss.*
- **ConvE** [13] is a *deep Learning* model; when computing the score for any fact $\langle h, r, t \rangle$ it applies convolutional layers to process the concatenation of $\boldsymbol{h}$ and $\boldsymbol{r}$; the output is combined with $\boldsymbol{t}$ through dot product. Our implementation is faithful to the one by the original authors, and trained with *Binary Cross-Entropy Loss.*

Table 2 reports the LP performance of each model on each dataset.[4]

5.2 Baselines

We compare the performance of Kelpie with two very recent systems for LP interpretation: the work by Zhang *et al.* [62], which we denote as *Data Poisoning (DP)*, and the *Criage* framework by Pezeshkpour *et al.* [40]. The former system is not open source, so we have re-implemented it from scratch; our implementation is available in our code repository. For the latter, we have adapted the code from their Github repository.[5]

As mentioned in Section 3 both systems perform data poisoning: given a prediction $\langle h, r, t \rangle$ they identify the one fact that, if removed or added to $\mathcal{G}_{train}$, worsens $\phi(h, r, t)$ the most. Our necessary scenario is analogous to their removal node, so in this setting we can directly compare our results to theirs. The sufficient scenario is more problematic: these systems do not study how to improve predictions, so, as they are, they cannot identify which facts should be added to other entities c to convert them. Nonetheless, we find that in this regard their formulations and code can be adapted:

- Data Poisoning, in its removal mode, would normally shift $\boldsymbol{h}$ to worsen $\phi(h, r, t)$ and search for the training fact featuring h whose score degrades the most. We adapt this formulation to the sufficient scenario by applying it symmetrically: when trying to convert prediction $\langle c, r, t \rangle$ we shift $\boldsymbol{c}$ in the direction that would *improve* the prediction score, and verify which fact to add to c displays the greatest improvement.
- Criage uses Taylor-approximated Influence Functions to estimate the score variation caused by adding or removing a fact; therefore we can just reprogram Criage to choose the fact that, if added to the entity c to convert, would improve the score of $\langle c, r, t \rangle$ the most, instead of worsening it.

[4] Details on the hyperparameters used in training are available in our online repository.
[5] https://github.com/pouyapez/criage/tree/master/CRIAGE

The code provided by the Criage authors only supports multiplicative models such as ConvE and ComplEx, but not TransE; hence, in our TransE experiments we only compare Kelpie to DP.

5.3 End-to-end Experiments

In this section, we report the end-to-end effectiveness of Kelpie necessary and sufficient explanations, for each model and dataset.

Methodology and Metrics. In both the necessary and the sufficient scenario, for each model and dataset we randomly sample a set P of 100 correct test tail predictions, we extract the corresponding explanations and we apply them to the training set $\mathcal{G}_{train}$; then, following to the methodology used by Criage [40], we measure their effectiveness as the variation of $H@1$ and MRR on the involved predictions, i.e., $\Delta H@1$ and ΔMRR. More specifically:

- In the *necessary scenario*, after extracting explanations for all predictions in P, we remove their facts from $\mathcal{G}_{train}$ and retrain the model. Since the original model correctly inferred those predictions, their original $H@1$ and MRR are both 1.0; if the extracted explanations are indeed necessary, on the contrary, the retrained model should be unable to infer the predictions in P. Therefore, we measure the effectiveness of necessary explanations as the *worsening* in the $H@1$ and MRR over P: the *more negative* $\Delta H@1$ and ΔMRR, the greater the effectiveness of the explanations.
- In the *sufficient scenario*, for each prediction $\langle h, r, t\rangle \in P$ we draw a set C of 10 random entities c and extract sufficient explanations that should convert them, i.e., that should lead the model to predict $\langle c, r, t\rangle$ as described in Section 4.2. We call P_C the set of these fictitious $10 \times 100 = 1000$ predictions $\langle c, r, t\rangle$. Since the original model did not infer the predictions in P_C by construction, their original $H@1$ is 0.0 and their MRR is ~ 0.0; if the extracted explanations are indeed sufficient, on the contrary, after adding their facts to the corresponding entities to convert, the re-trained model should infer the predictions in P_C. Therefore, we measure the effectiveness of sufficient explanations as the *improvement* in the $H@1$ and MRR over P_C: the *more positive* $\Delta H@1$ and ΔMRR, the greater the effectiveness of the extracted explanations.

An analogous methodology can be defined for head predictions.

Results. We report in Tables 3 and 4 the effectiveness of necessary and sufficient explanations respectively, and compare the results obtained by Kelpie with those obtained running analogous pipelines on the baselines DP and Criage. We also include the results of a Kelpie version limited to single-fact explanations, that we call $K1$.

Across all scenarios, datasets and models, Kelpie almost always outperforms baselines, confirming the effectiveness of the proposed framework. Our baselines, and in particular DP, still achieve competitive performance especially in the necessary scenario, while the gap from Kelpie widens in the sufficient scenario.

The $K1$ version of Kelpie usually obtains results comparable to the best baselines, often exceeding them in the sufficient scenario. Since both $K1$ and baselines extract explanations with only 1 fact, this is a further confirmation that the post-training process is indeed effective at identifying the most relevant facts to a prediction. On the other hand, the performance achieved by $K1$ is almost always worse than "full" Kelpie: this proves that supporting combinations longer than 1 fact is key to obtain satisfactory explanations.

		FB15k		WN18		FB15k-237		WN18RR		YAGO3-10	
		ΔH@1	ΔMRR	ΔH@1	ΔMRR	ΔH@1	ΔMRR	ΔH@1	ΔMRR	ΔH@1	ΔMRR
TransE	K1	-0.360	-0.211	-0.860	-0.771	-0.380	-0.253	-0.790	-0.714	-0.640	-0.466
	Kelpie	**-0.490**	**-0.321**	**-0.920**	-0.857	**-0.540**	**-0.356**	**-0.920**	**-0.904**	**-0.740**	**-0.580**
	DP	-0.380	-0.258	-0.900	**-0.859**	-0.460	-0.303	-0.770	-0.701	-0.670	-0.533
ComplEx	K1	-0.580	-0.466	-0.680	-0.530	-0.440	-0.244	-0.700	-0.538	-0.870	-0.718
	Kelpie	**-0.850**	**-0.695**	**-0.910**	**-0.827**	**-0.590**	**-0.413**	**-0.980**	**-0.913**	**-0.960**	**-0.858**
	DP	-0.540	-0.458	-0.800	-0.742	-0.340	-0.185	-0.750	-0.650	-0.810	-0.714
	Criage	-0.030	-0.020	-0.050	-0.045	-0.090	-0.050	-0.180	-0.150	-0.05	-0.030
ConvE	K1	-0.360	-0.228	-0.700	-0.581	-0.290	-0.191	-0.780	-0.622	-0.860	-0.735
	Kelpie	**-0.710**	**-0.516**	**-0.930**	**-0.860**	**-0.430**	**-0.284**	**-0.980**	**-0.914**	**-0.980**	**-0.884**
	DP	-0.350	-0.232	-0.790	-0.752	-0.290	-0.195	-0.850	-0.750	-0.880	-0.799
	Criage	-0.080	-0.056	-0.160	-0.146	-0.290	-0.196	-0.170	-0.156	-0.070	-0.042

Table 3: End to end effectiveness of necessary explanations. More negative values correspond to higher effectiveness.

		FB15k		WN18		FB15k-237		WN18RR		YAGO3-10	
		ΔH@1	ΔMRR	ΔH@1	ΔMRR	ΔH@1	ΔMRR	ΔH@1	ΔMRR	ΔH@1	ΔMRR
TransE	K1	+0.249	+0.445	+0.205	+0.370	+0.057	+0.088	+0.075	+0.114	+0.027	+0.082
	Kelpie	**+0.319**	**+0.516**	**+0.259**	**+0.434**	**+0.128**	**+0.218**	**+0.117**	**+0.169**	**+0.048**	+0.109
	DP	+0.273	+0.461	+0.183	+0.350	+0.051	+0.080	+0.082	+0.116	+0.036	**+0.117**
ComplEx	K1	+0.943	**+0.921**	**+0.953**	**+0.962**	+0.320	+0.408	+0.834	+0.877	+0.847	+0.885
	Kelpie	**+0.945**	+0.920	**+0.953**	**+0.962**	**+0.377**	**+0.490**	+0.834	+0.877	**+0.858**	**+0.892**
	DP	+0.910	+0.888	+0.931	+0.940	+0.245	+0.334	**+0.836**	**+0.878**	+0.835	+0.878
	Criage	+0.068	+0.069	+0.105	+0.137	+0.035	+0.038	+0.110	+0.147	+0.000	+0.000
ConvE	K1	+0.662	+0.632	+0.893	+0.891	+0.161	+0.138	**+0.857**	**+0.882**	**+0.807**	+0.847
	Kelpie	**+0.677**	**+0.649**	**+0.903**	**+0.900**	**+0.225**	**+0.203**	+0.827	+0.856	+0.799	**+0.848**
	DP	+0.234	+0.199	+0.396	+0.412	+0.202	+0.169	+0.373	+0.419	+0.366	+0.391
	Criage	+0.106	+0.065	+0.164	+0.166	+0.132	+0.094	+0.150	+0.164	+0.019	+0.020

Table 4: End to end effectiveness of sufficient explanations. More positive values correspond to higher effectiveness.

It is interesting to take a closer look at the predictions that the baselines fail to explain and that Kelpie tackles successfully. For instance, let us consider the YAGO3-10 tail prediction $\langle$*Penarth*, *located_in*, *Wales*$\rangle$. Our baselines identify, as a necessary explanation, the fact that *Penarth* is in *South_Wales*; this explanation is ineffective, as removing this fact and retraining the model does not affect the prediction. The reason is that the prediction is enabled by multiple *Penarth* facts, and no one among them can, by itself, fully explain it. Kelpie, in addition to *Penarth* being *located_in, South_Wales*, also identifies that it is in *Glamorgan* (a Welsh region) and in *Vale_of_Glamorgan* (a Welsh county). We have verified that the Kelpie explanation is indeed effective: removing its facts, the original prediction is not inferred anymore. Analogously, when extracting a sufficient explanation for the prediction that actor *Utpal_Dutt* has *Indian nationality*, our baselines just yield the fact that he is married; this explanation is not effective, because adding this fact to other entities does not convert their *nationality* to *Indian*. On the contrary, Kelpie extracts an effective explanation observing that

Dutt has lived in the Indian city of Kolkata, believes in Hinduism, and has Indian and Bengali ethnicity.

$K1$ results get closer to "full" Kelpie results (and, in one instance, they even surpass them) on datasets where predictions are known to mostly depend on the presence of one specific training fact. This phenomenon is particularly prominent in WN18, that is known to feature inverse relations [13], and in WN18RR, where test facts seem to be only predictable if their symmetric version is present in the training set [43]. This is also reflected in the size of the extracted explanations, that we discuss in Section 5.4.

Interestingly, on all models, all frameworks appear less effective on FB15k-237 than on the other datasets. We explain this by considering that many test predictions in FB15k-237 have been observed to be affected by forms of bias that make entities artificially easier to predict independently of their training facts [44]. This makes those predictions harder to interpret for local post-hoc frameworks, whose explanations are formulated in terms of those facts.

We also observe that the TransE sufficient explanations of both Kelpie and DP seem quite ineffective on WN18RR and YAGO3-10. This most likely depends on TransE itself having extreme difficulties on these datasets (see Table 2), as if it could not generalize any patterns to infer predictions: trying to identify patterns to convert entities c is bound to be unsuccessful as well. Among our baselines, DP obtains good results on ComplEx and TransE experiments, whereas its performance significantly degrades on ConvE. We explain this by considering that ComplEx and TransE have very simple scoring functions where $\frac{\partial \phi(h,\,r,\,t)}{\partial \boldsymbol{h}}$ does not depend on the value of $\boldsymbol{h}$.[6] This embodies the best-case scenario for DP, which estimates relevances by translating $\boldsymbol{h}$ by a constant ϵ in the direction of that derivative. ConvE, on the other hand, relies on a more recent architecture based on multi-layered deep learning, so the derivative of its scoring function is far less stable: in this setting the DP approach of applying an arbitrary constant ϵ perturbation can be significantly less effective.

We finally observe that the baseline Criage seems to perform rather poorly, only achieving results comparable to the other systems with the ConvE model on FB15k-237.

To confirm the reliability of our results, we have repeated a subset of our end-to-end experiments on 10 different samples of 100 tail predictions each, obtaining similar values to those in this section. We report this procedure in detail in our online repository.

5.4 Explanation Lengths and Minimality

In this section we analyze the lengths of the explanations obtained in our end-to-end experiments and verify that they are indeed the smallest sets of facts explaining the original predictions.

		FB15k		WN18		FB15k-237		WN18RR		YAGO3-10	
		AVG	STD	AVG	STD	AVG	STD	AVG	STD	AVG	STD
Necessary	TransE	2.78	1.13	2.17	1.24	2.50	1.15	1.67	0.96	1.99	1.20
	ComplEx	3.40	1.10	3.39	1.00	3.83	0.57	3.16	1.08	2.27	1.31
	ConvE	3.36	0.89	2.91	1.20	2.26	1.24	2.66	1.21	1.73	0.97
Sufficient	TransE	1.89	1.03	1.02	0.14	3.43	0.83	1.67	0.79	1.45	0.73
	ComplEx	1.40	0.75	1.00	0.00	2.51	1.20	1.00	0.00	1.04	0.31
	ConvE	1.83	1.23	1.01	0.10	3.09	1.10	1.04	0.24	1.18	0.48

Table 5: Lengths of the extracted explanations.

		FB15k		WN18		FB15k-237		WN18RR		YAGO3-10	
		H@1	MRR	H@1	MRR	H@1	MRR	H@1	MRR	H@1	MRR
Necessary	TransE	-51.0%	-55.6%	-65.2%	-73.1%	-51.9%	-57.5%	-76.1%	-80.0%	-67.6%	-70.4%
	ComplEx	-55.3%	-63.2%	-46.2%	-54.9%	-39.0%	-49.1%	-44.9%	-52.9%	-61.5%	-71.9%
	ConvE	-45.1%	-51.1%	-59.1%	-64.8%	-44.2%	-48.6%	-46.9%	-54.5%	-72.5%	-77.0%
Sufficient	TransE	-61.8%	-60.6%	-96.9%	-96.4%	-35.4%	-35.8%	-76.1%	-75.0%	-68.8%	-76.3%
	ComplEx	-79.2%	-75.0%	-100.0%	-99.9%	-42.8%	-40.1%	-100.0%	-99.7%	-99.3%	-98.2%
	ConvE	-67.9%	-64.8%	-99.5%	-98.2%	-38.6%	-19.4%	-98.7%	-98.0%	-89.9%	-89.6%

Table 6: Loss in effectiveness when sub-sampling our end-to-end necessary and sufficient explanations.

We display in Table 5 the average explanation length (AVG) and corresponding Standard Deviation (STD) for each model and dataset both in the necessary and in the sufficient scenario. Under the same model and dataset, necessary explanations are always longer than sufficient ones. This is reasonable: for instance, as already mentioned in Section 4, the necessary explanation for tail prediction ⟨*Barack_Obama, nationality, USA*⟩ would probably feature multiple facts, e.g., ⟨*Barack_Obama, president_of, USA*⟩ and ⟨*Barack_Obama, part_of,* 109^{th}*_US_Congress*⟩; on the contrary, to convert any other entity c to have *nationality USA* it is probably enough to just add ⟨*c, president_of, USA*⟩. We also observe that in many cases the average explanation length is lesser than 2: this suggests that Kelpie does indeed identify the *smallest* sets of facts constituting an explanation.

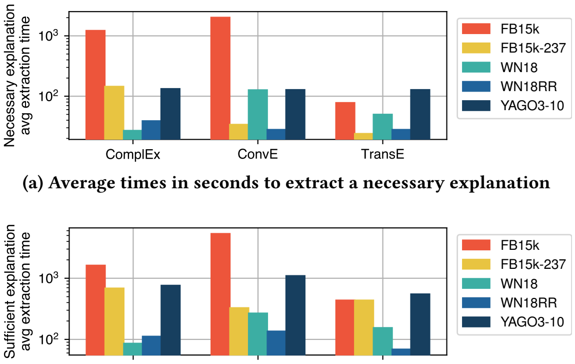
We conduct in this regard a thorough study on explanation minimality, and we verify that, removing subsets from the explanations extracted in our end-to-end experiments, they lose their effectiveness. We remove from each explanation X^* a random subset whose size is extracted from a uniform distribution $[1, len(X^*))$[7]; we then re-train the model applying the sub-sampled explanations instead of the complete ones, and measure the consequent loss of effectiveness. In both the necessary and the sufficient scenario, the more negative the value, and the greater the loss in effectiveness:

- In **necessary** explanations, the effectiveness is the decrease they cause in the H@1 and MRR of the predictions to explain. For example, if the original explanations caused a -0.90 H@1 drop and the sub-sampled ones just a -0.30 H@1 drop, the loss in H@1 effectiveness is $(-0.30 - (-0.90))/(-0.90) = -66.7\%$.
- In **sufficient** explanations, the effectiveness is the increase they cause in the H@1 and MRR of the predictions to convert. For example, if the complete explanations caused a +0.80 H@1 increase and the sub-sampled ones just cause a +0.20 H@1 increase, the loss in H@1 effectiveness is $(0.20 - 0.80)/0.80 = -75.0\%$.

We report our results in Table 6. In both scenarios, across all models and datasets we observe quite high percentages, proving that Kelpie does indeed identify in most cases minimal explanations. As an extreme example, ComplEx sufficient explanations on datasets WN18 and WN18RR always amount to *a single fact*: this shows that Kelpie does stop the explanation extraction as soon as

[6] In ComplEx $\frac{\partial \phi(h,\,r,\,t)}{\partial \boldsymbol{h}}$ is a constant function; in TransE it is a step function, so it can be considered constant with respect to the small perturbations applied by DP.

[7] Explanations of length 1 are minimal by definition: when in this experiment we subsample them, they result in null explanations that do not affect prediction ranks.



(a) Average times in seconds to extract a necessary explanation

(b) Average times in seconds to extract a sufficient explanation

Figure 5: Extraction times across all models and datasets.

the combination of identified facts is satisfying, rather than just piling up as many facts as possible. We notice that in FB15k-237, compared to the other datasets, Kelpie is more prone to finding slightly longer explanations than required in both the necessary and the sufficient scenario. This is consistent to our previous observations about FB15k-237: the presence of the aforementioned types of bias, in addition to affecting the capability of all frameworks to identify viable explanations, also makes it harder for Kelpie to correctly assess the length of the explanation to extract.

5.5 Execution Times

We provide here insights on the computational cost of running Kelpie, and on how the Pre-Filter module affects our efficiency.

End-to-end average times. In Figures 5a, 5b we report the average explanation extraction times across our end-to-end experiments. Across all datasets and models, Kelpie generally manages to extract successful explanations in seconds or minutes. These times mostly depend on: (*i*) the average number of training facts per entity, which determines the space of candidate explanations to explore (e.g., in FB15k entities tend to be featured in more training facts than in the other datasets, leading to longer computations); and (*ii*) the model hyperparameters, which directly affect the post-training duration. In general, extracting sufficient explanations tends to require longer times than necessary explanations: this is understandable, as in the sufficient scenario all the analyzed candidate explanations are applied to additional random entities to estimate their relevance.

Efficiency of the Pre-Filtering step. As described in Section 4.1 our Pre-Filter directly affects our execution times by reducing the space of candidate explanations. When explaining a tail prediction $\langle h,\ r,\ t\rangle$ we start our exploration assessing with post-training the individual relevances of the k most promising facts in $\mathcal{G}^h_{train}$; in absence of Pre-Filtering, we would have to cover all of them, which might affect execution times for high-degree entities. Even worse, without Pre-Filtering the space of l-long combinations would become $\binom{|\mathcal{G}^h_{train}|}{l}$ instead of $\binom{k}{l}$; even with the early termination policies enacted by Explanation Builder, the presence of many facts with medium-low relevance can make the preliminary relevance distribution much larger and noisier, slowing the exploration down.

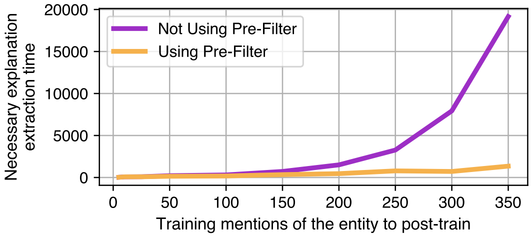


Figure 6: Extraction times with and without the Pre-Filter.

We report in Figure 6 the average times required to extract, both with and without Pre-Filtering, explanations for ComplEx FB15k-237 tail predictions in which the head occurs between 5 and 350 times in training. Our results are averaged across 10 different sets of tail predictions to obtain more reliable results. Without Pre-Filtering, we observe an almost exponential trend with respect to the degree of the head entity: this is in line with the binomial computational complexity mentioned above. On the contrary, using Pre-Filtering the extraction times remain remarkably stable, and almost unaffected by the degree of the head entity.

5.6 Kelpie in action

We show in this section a few qualitative examples of the useful insights that Kelpie explanations can provide.

Example 1. We use Kelpie to extract necessary explanations for three YAGO3-10 tail predictions with structure ⟨*actor, acted_in, movie*⟩, and obtained respectively with TransE, ConvE and ComplEx; we report the predictions and their explanations in Table 7. In each of the three cases our explanation consists in a set of other movies in which the same actor performed. We find that, in all three predictions, the known actor (i.e., the head entity) is part of a recurring acting group: the *Dead End Kids* for Billy Halop; *Our Gang* for Mickey Daniels; the *Three Stooges* for Moe Howard. The obtained explanations reveal that our models could predict the correct movie because, in training, they had seen the same actor working together with the other cast members of that movie in multiple occasions (i.e., the films mentioned in the explanations). Intriguingly, in our dataset there is no explicit mention of any of these acting groups: the models were able to identify the correlation among their members by just observing that all of them worked together several times. Observations like this are key to understanding the inner mechanisms of models, and they are only possible in presence of explainability tools like Kelpie.

Example 2. As mentioned in Section 1, Kelpie can also unveil biases in the training data. We show this by taking into account some correctly predicted YAGO3-10 test facts and the corresponding Kelpie sufficient explanations, reported in Table 8. All predictions follow the pattern ⟨*person, born_in, city*⟩; surprisingly, they are always explained with the person either *playing for* or *being affiliated to* a football team from that city or nation. Despite the loose semantic correlation between the football team of players and their their place of birth, these explanations are correct, i.e., they do convert other entities. These explanations reveal that our models, instead of

Prediction to explain	Sufficient Explanation
Billy_Halop, acted_in, Hell's_Kitchen	Billy_Halop, acted_in, The_Angels_Wash_Their_Faces Billy_Halop, acted_in, On_Dress_Parade Billy_Halop, acted_in, Crime_School
Mickey_Daniels, acted_in, The_Big_Show	Mickey_Daniels, acted_in, A_Quiet_Street Mickey_Daniels, acted_in, July_Days Mickey_Daniels, acted_in, Stage_Fright Mickey_Daniels, acted_in, The_Champeen
Moe_Howard, acted_in, Income_Tax_Sappy	Moe_howard, acted_in, Higher_Than_a_Kite Moe_howard, acted_in, Hello_Pop! Moe_howard, acted_in, Booby_Dupes Moe_howard, acted_in, Bedlam_in_Paradise

Table 7: An example of Kelpie necessary explanations.

Prediction to explain	Sufficient Explanation
Benedict_Vilakazi, born_in, Soweto	Benedict_Vilakazi, plays_for, Mpumalanga_Black_Aces
Nikola_Jerkan, born_in, Split	Nikola_Jerkan, affiliated_to, Croatia_Football_Team
Gabriel_Gómez, born_in, Panama_City	Gabriel_Gómez, plays_for, Tauro_f.c.
Pablo_Menéndez, born_in, Montevideo	Pablo_Menéndez, plays_for, Club_Nacional_Football
Ljubiša_Spajić, born_in, Belgrade	Ljubiša_Spajić, plays_for, Budućnost_Podgorica
Otto_Hemele, born_in, Prague	Otto_Hemele, affiliated_to, Slavia_Prague
İskender_Alın, born_in, Istanbul	İskender_Alın, affiliated_to, Bakırköyspor

Table 8: An example of dataset bias unveiled by Kelpie.

leveraging reasonable correlations, are being affected by *data bias*. In YAGO3-10 facts rarely convey personal data, so predicting the place of birth of a person is very challenging; in our case the best pattern that models can leverage seems to be the slight preference that football players may have towards teams from their birthplace; this phenomenon is further fueled by YAGO3-10 being strongly focused on the football domain. Kelpie has thus highlighted that the dataset is imbalanced, and does not accurately represent real-world semantics; this can allow researchers to correct it, e.g., enriching it to make it more akin to production KGs.

5.7 End-user Study

We report here an end-user study on the usability of Kelpie explanations and on how they affect on trust in LP models. We take into account 36 ComplEx and TransE correct test predictions on YAGO3-10, extract the corresponding Kelpie explanations and formulate, for each prediction-explanation pair, three questions:

1. "How clear is the notion of necessary/sufficient explanation from 1 to 10 in the current case?". This question measures the *conceptual* comprehension of necessity and sufficiency.
2. "What would happen if we removed/added the facts of this explanation to the training set?" (i.e., applying the definitions in Section 2.2). The possible answers are (*i*) the actual effect of the explanation; (*ii*) "Nothing would change"; (*iii*) "I don't know"; and (*iv*) a nonsensical wrong answer. This question measures the *practical* comprehension of the effect of necessary/sufficient explanations.
3. "Based on the current explanation, how reliable is the model from 1 to 10?". This question measures how revealing the reasons behind the predictions affects the *trust* in the model.

We submit the overall 108 questions to 44 participants, of which 22 female and 22 male, aged between 23 and 79 years old, and with education ranging from High School to University. The vast majority of our participants are not Computer Science practitioners.

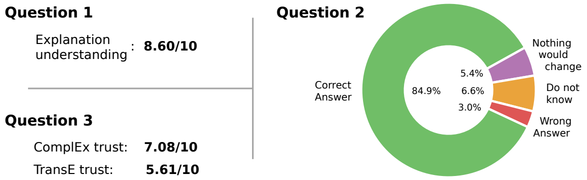


Figure 7: End user study results.

The 1^{st} and 2^{nd} question focus on the usability of Kelpie explanations. In the first question we usually obtain very high values, with an average of 8.6 out of 10. Similarly, in the 2^{nd} question we obtain correct answers in the vast majority of cases: we report the distribution of answers in across the 40 predictions in Figure 7. These observations confirm that the explanations extracted by Kelpie are generally easy to understand, proving their strong usability.

The 3^{rd} question focuses on how the perceived reliability of the LP model changes when its behaviour is explained: as mentioned in Section 1, explanations allow users to assess whether a model can be trusted or not. We find that, on average, the participants rate their trust in ComplEx and TransE as 7.1 and 5.6 out of 10 respectively. These findings are intriguing, as they imply that ComplEx is more prone than TransE to leveraging patterns that match human intuition. As reported in Table 2, in general ComplEx achieves much better predictive performance than TransE: this suggests that the capability to leverage human-like correlations may be key to yielding more accurate link predictions.

6 CONCLUSIONS

Motivated by fast-growing body of literature regarding Link Prediction (LP) on Knowledge Graphs we have presented Kelpie, a full-fledged explainability framework for embedding-based LP models. We have formally defined the complementary concepts of necessary and sufficient explanations in the LP domain; we have then proposed a novel framework capable of extracting both types of explanations. We have described in detail the framework components, discussing their implementations. We have conducted extensive experiments applying Kelpie on models representative for the three main families of LP approaches, and on the five best-established datasets in literature. We have compared Kelpie to baselines performing data poisoning via adversarial modifications, that we have adapted to our scenario when needed; we have shown that Kelpie significantly outperforms them in the vast majority of conditions.

Future Works. As pointed out in Section 5.1, while in our paper and implementation we focus on fact-based models, the Kelpie methodology can be applied to *any* embedding-based LP models, including those that leverage contextual information. For these systems, it would be intriguing to extend Kelpie to also extract context-based explanations. For example, given a path-based model, post-training may allow us to identify entire paths enabling its predictions in addition to the most relevant head (or tail) training facts. We plan investigate this research direction in the future.

ACKNOWLEDGMENTS

We thank proff. Alessandro Micarelli, Fabio Gasparetti and Alberto Paoluzzi for providing the computational resources for our work.

REFERENCES

[1] K. Aas, M. Jullum, and A. Løland. Explaining individual predictions when features are dependent: More accurate approximations to shapley values. *Artif. Intell.*, 2021.

[2] F. Akrami, M. S. Saeef, Q. Zhang, W. Hu, and C. Li. Realistic Re-evaluation of Knowledge Graph Completion Methods: An Experimental Study. In *SIGMOD*, 2020.

[3] V. Arya, R. K. E. Bellamy, P. Chen, A. Dhurandhar, M. Hind, S. C. Hoffman, S. Houde, Q. V. Liao, R. Luss, A. Mojsilovic, S. Mourad, P. Pedemonte, R. Raghavendra, J. T. Richards, P. Sattigeri, K. Shanmugam, M. Singh, K. R. Varshney, D. Wei, and Y. Zhang. One explanation does not fit all: A toolkit and taxonomy of AI explainability techniques. *CoRR*, abs/1909.03012, 2019.

[4] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. Dbpedia: A nucleus for a web of open data. In *The semantic web*. Springer, 2007.

[5] A. Baraldi, F. D. Buono, M. Paganelli, and F. Guerra. Using landmarks for explaining entity matching models. In *EDBT*, 2021.

[6] R. Bhowmik and G. de Melo. Explainable link prediction for emerging entities in knowledge graphs. In *ISWC*, 2020.

[7] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, 2008.

[8] S. Bonner, I. P. Barrett, C. Ye, R. Swiers, O. Engkvist, and W. L. Hamilton. Understanding the performance of knowledge graph embeddings in drug discovery. *arXiv preprint arXiv:2105.10488*, 2021.

[9] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *NIPS*, 2013.

[10] G. L. Ciampaglia, P. Shiralkar, L. M. Rocha, J. Bollen, F. Menczer, and A. Flammini. Computational fact checking from knowledge networks. *PloS one*, 2015.

[11] V. D. Cicco, D. Firmani, N. Koudas, P. Merialdo, and D. Srivastava. Interpreting deep learning models for entity resolution: an experience report using LIME. In *aiDM@SIGMOD*, 2019.

[12] G. V. den Broeck, A. Lykov, M. Schleich, and D. Suciu. On the tractability of SHAP explanations. In *AAAI*, 2021.

[13] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel. Convolutional 2d knowledge graph embeddings. In *AAAI*, 2018.

[14] M. Ebraheem, S. Thirumuruganathan, S. R. Joty, M. Ouzzani, and N. Tang. Deeper - deep entity resolution. *CoRR*, abs/1710.00597, 2017.

[15] J. Feng, M. Huang, M. Wang, M. Zhou, Y. Hao, and X. Zhu. Knowledge graph embedding by flexible translation. In *KR*, 2016.

[16] T. Gaudelet, B. Day, A. R. Jamasb, J. Soman, C. Regep, G. Liu, J. B. Hayter, R. Vickers, C. Roberts, J. Tang, et al. Utilizing graph machine learning within drug discovery and development. *Briefings in Bioinformatics*, 2021.

[17] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi. A survey of methods for explaining black box models. *ACM Comput. Surv.*, 2018.

[18] L. Guo, Z. Sun, and W. Hu. Learning to Exploit Long-term Relational Dependencies in Knowledge Graphs. In *ICML*, 2019.

[19] F. R. Hampel, E. M. Ronchetti, P. J. Rousseeuw, and W. A. Stahel. *Robust statistics: the approach based on influence functions*. John Wiley & Sons, 2011.

[20] A. Holzinger, P. Kieseberg, E. R. Weippl, and A. M. Tjoa. Current advances, trends and challenges of machine learning and knowledge extraction: From machine learning to explainable AI. In *CD-MAKE*, 2018.

[21] V. Huynh and P. Papotti. A Benchmark for Fact Checking Algorithms Built on Knowledge Bases. In *CIKM*, 2019.

[22] L. Ingber. Very fast simulated re-annealing. *Mathematical and computer modelling*, 1989.

[23] R. Kadlec, O. Bajgar, and J. Kleindienst. Knowledge Base Completion: Baselines Strike Back. In *Rep4NLP@ACL*, 2017.

[24] E. Kaufmann and S. Kalyanakrishnan. Information complexity in bandit subset selection. In S. Shalev-Shwartz and I. Steinwart, editors, *COLT*, 2013.

[25] P. W. Koh and P. Liang. Understanding black-box predictions via influence functions. In D. Precup and Y. W. Teh, editors, *ICML*, 2017.

[26] N. Kolitsas, O. Ganea, and T. Hofmann. End-to-end neural entity linking. In *CoNLL*, 2018.

[27] T. Lacroix, N. Usunier, and G. Obozinski. Canonical Tensor Decomposition for Knowledge Base Completion. In *ICML*, 2018.

[28] Y. Li, J. Li, Y. Suhara, A. Doan, and W. Tan. Deep entity matching with pre-trained language models. *Proc. VLDB Endow.*, 2020.

[29] Y. Liu, W. Hua, K. Xin, and X. Zhou. Context-aware temporal knowledge graph embedding. In *WISE*, 2019.

[30] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In *NIPS*, 2017.

[31] F. Mahdisoltani, J. Biega, and F. M. Suchanek. YAGO3: A knowledge base from multilingual wikipedias. In *CIDR*, 2015.

[32] T. B. Malas, W. J. Vlietstra, R. Kudrin, S. Starikov, M. Charrout, M. Roos, D. J. Peters, J. A. Kors, R. Vos, P. AC't Hoen, et al. Drug prioritization using the semantic properties of a knowledge graph. *Scientific reports*, 2019.

[33] G. A. Miller. Wordnet: a lexical database for english. *CACM*, 1995.

[34] D. Monroe. AI, explain yourself. *CACM*, 2018.

[35] J. G. Moreno, R. Besançon, R. Beaumont, E. D'hondt, A.-L. Ligozat, S. Rosset, X. Tannier, and B. Grau. Combining word and entity embeddings for entity linking. In *ESWC*. Springer, 2017.

[36] S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, G. Krishnan, R. Deep, E. Arcaute, and V. Raghavendra. Deep learning for entity matching: A design space exploration. In *SIGMOD*, 2018.

[37] D. Q. Nguyen. An overview of embedding models of entities and relationships for knowledge base completion. *CoRR*, abs/1703.08098, 2017.

[38] S. Paliwal, A. de Giorgio, D. Neil, J.-B. Michel, and A. M. Lacoste. Preclinical validation of therapeutic targets predicted by tensor factorization on heterogeneous graphs. *Scientific reports*, 2020.

[39] C. H. Papadimitriou and K. Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.

[40] P. Pezeshkpour, Y. Tian, and S. Singh. Investigating robustness and interpretability of link prediction via adversarial modifications. In *NAACL-HLT*, 2019.

[41] M. T. Ribeiro, S. Singh, and C. Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *SIGKDD*, 2016.

[42] M. T. Ribeiro, S. Singh, and C. Guestrin. Anchors: High-precision model-agnostic explanations. In *AAAI*, 2018.

[43] A. Rossi, D. Barbosa, D. Firmani, A. Matinata, and P. Merialdo. Knowledge graph embedding for link prediction: A comparative analysis. *TKDD*, 2021.

[44] A. Rossi, D. Firmani, and P. Merialdo. Knowledge graph embeddings or bias graph embeddings? a study of bias in link prediction models. In *DL4KG*, 2021.

[45] A. Rossi and A. Matinata. Knowledge Graph Embeddings: Are Relation-Learning Models Learning Relations? In *PIE*, 2020.

[46] L. S. Shapley. *A value for n-person games*. Princeton University Press, 2016.

[47] P. Shiralkar, A. Flammini, F. Menczer, and G. L. Ciampaglia. Finding streams in knowledge graphs to support fact checking. In *ICDM*, 2017.

[48] A. Singhal. Introducing the knowledge graph: things, not strings, 2012. Blogpost in the Official Google Blog.

[49] J. C. Spall. *Introduction to stochastic search and optimization: estimation, simulation, and control*. John Wiley & Sons, 2005.

[50] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *WWW*. ACM, 2007.

[51] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 1996.

[52] K. Toutanova and D. Chen. Observed versus latent features for knowledge base and text inference. In *CVSC*, 2015.

[53] R. Trivedi, B. Sisman, X. L. Dong, C. Faloutsos, J. Ma, and H. Zha. LinkNBed: Multi-Graph Representation Learning with Entity Linkage. In *ACL*, 2018.

[54] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard. Complex Embeddings for Simple Link Prediction. In *ICML*, 2016.

[55] H. Wang, H. Ren, and J. Leskovec. Relational message passing for knowledge graph completion. In *KDD*, 2021.

[56] Q. Wang, Z. Mao, B. Wang, and L. Guo. Knowledge graph embedding: A survey of approaches and applications. *TKDE*, 2017.

[57] D. S. Watson, L. Gultchin, A. Taly, and L. Floridi. Local explanations via necessity and sufficiency: Unifying theory and practice. In *UAI*, 2021.

[58] R. West, E. Gabrilovich, K. Murphy, S. Sun, R. Gupta, and D. Lin. Knowledge base completion via search-based question answering. In *WWW*, 2014.

[59] R. Xie, Z. Liu, and M. Sun. Representation learning of knowledge graphs with hierarchical types. In *IJCAI*, 2016.

[60] Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec. Gnnexplainer: Generating explanations for graph neural networks. In *NIPS*, 2019.

[61] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W.-Y. Ma. Collaborative knowledge base embedding for recommender systems. In *SIGKDD*, 2016.

[62] H. Zhang, T. Zheng, J. Gao, C. Miao, L. Su, Y. Li, and K. Ren. Data poisoning attack against knowledge graph embedding. In *IJCAI*, 2019.

[63] W. Zhang, S. Deng, H. Wang, Q. Chen, W. Zhang, and H. Chen. Xtranse: Explainable knowledge graph embedding for link prediction with lifestyles in e-commerce. In *JIST*, 2019.

[64] W. Zhang, B. Paudel, W. Zhang, A. Bernstein, and H. Chen. Interaction embeddings for prediction and explanation in knowledge graphs. In *WSDM*, 2019.