Kelpie: an Explainability Framework for Embedding-based Link Prediction Models

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

The latest generations of Link Prediction (LP) models rely on embeddings to tackle incompleteness in Knowledge Graphs, achieving great performance at the cost of interpretability. Their opaqueness limits the trust that users can place in them, hindering their adoption in real-world applications. We have recently introduced Kelpie, an explainability framework tailored specifically for embedding-based LP models. Kelpie can be applied to any embedding-based LP model, and supports two explanation scenarios that we have called *necessary* and *sufficient*. In this demonstration we showcase Kelpie's capability to explain the predictions of models based on vastly different architectures on the 5 major datasets in literature.

PVLDB Reference Format:
Andrea Rossi, Donatella Firmani, Paolo Merialdo, and Tommaso Teofili. Kelpie: an Explainability Framework for Embedding-based Link Prediction Models. PVLDB, 15(12): 3566 - 3569, 2022.
doi:10.14778/3554821.3554845

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/AndRossi/Kelpie-demo.

1 INTRODUCTION

Knowledge Graphs (KGs) are structured stores of real-world information. In a KG nodes represent *entities* and edges are labeled with semantic *relations*; two entities h and t linked by an edge with relation r form a *fact* $\langle h, r, t \rangle$, where h is the *head* and t is the *tail*. Nowadays, web-scale KGs are widely used both in academia (e.g., WikiData, YAGO) and in industry (e.g., Google, Facebook); furthermore, KGs are strikingly good for representing historical data. Even the largest KGs, though, suffer from *incompleteness*, as they only encompass a fraction of the knowledge they should contain.

Researchers have designed several approaches to KG Completion; among them, Link Prediction (LP) leverages the known facts to infer the missing ones. For example, knowing $\langle$*Dante Alighieri*, *born in*, *Florence*$\rangle$ and $\langle$*Dante Alighieri*, *Prior of*, *Florence*$\rangle$, we may infer $\langle$*Dante Alighieri*, *citizenship*, *Florence*$\rangle$ (if it was previously missing).

With the rise of Machine Learning (ML), the dominant approach to LP has become to map the graph elements to vectorized representations called *KG embeddings*. Since the seminal TransE model [3] an impressive variety of embedding-learning LP systems have been created, often paralleling or even surpassing traditional methods [10]. Furthermore, KG embeddings can be applied to downstream tasks such as fact checking [6] and KG Alignment [14].

A major shortcoming of embedding-based LP models lies in their opaqueness: as it often happens with ML methodologies, these systems do not provide any insights on the reasons behind their outcomes. This undermines the trust that users can place in these systems, and it hinders their adoption in fields where explainability may be an inherently requirement. For example, while LP models have shown great potential in biomedical tasks such as Drug Discovery and Repurposing, human users need to trust and understand these models in order to make such contributions possible [2].

To overcome these issues we have developed Kelpie [12], an explainability framework tailored specifically for embedding-based LP models. Kelpie explains any prediction by identifying the minimum subset of training facts that have led the model to yield it. Kelpie explanations can be precious when assessing the semantic coherence of a model: for instance, if the explanation to the prediction $\langle$*Dante Alighieri*, *citizenship*, *Florence*$\rangle$ does not match human intuition, e.g., $\langle$*Dante Alighieri*, *profession*, *poet*$\rangle$, this can indicate that the model leverages spurious correlations, or that the training data are biased. Our approach takes actively into account the inner mechanisms of LP models, and it can be theoretically applied to *any* embedding-based architecture: this can be invaluable in a sparkling field like LP, where dozens of novel methods are proposed each year. The key contributions in this demonstration are the following:

- Section 2 describes the technical background of Kelpie, along with the main challenges of explaining LP systems;
- Section 3 presents Kelpie with its explanation scenarios, its methodologies and its implementation, highlighting its novelty;
- Section 4 describes our demonstration for Kelpie: our presentation will cover several settings, allowing our audience to choose among multiple scenarios, datasets and models.

We finally provide conclusive remarks in Section 5.

Proceedings of the VLDB Endowment, Vol. 15, No. 12 ISSN 2150-8097.
doi:10.14778/3554821.3554845

2 TECHNICAL BACKGROUND

In this section we fist define the core ideas behind embedding-based LP models; we then provide key concepts on their explainability.

2.1 Embedding-Based Link Prediction

A Knowledge Graph (KG) is a labeled directed graph $KG = (\mathcal{E}, \mathcal{R}, \mathcal{G})$ where $\mathcal{E}$ is the set of entities, $\mathcal{R}$ is the set of relations and $\mathcal{G} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ is the set of *facts*. LP datasets are usually sampled from real-world KGs, and their sets of facts $\mathcal{G}$ are further split into a training set $\mathcal{G}_{train}$, a validation set $\mathcal{G}_{valid}$, and a test set $\mathcal{G}_{test}$.

Embedding-based LP models usually work by defining a *scoring function* ϕ that, for any fact $\langle h, r, t \rangle$, uses the embeddings of h, r and t to estimate the plausibility of that fact. In our formulations we assume that high ϕ values convey high plausibility; analogous formulations can be derived for the opposite case. The embeddings of all the KG elements are initialized randomly and then trained to optimize the scores of the training facts, deemed true *a priori*. LP models may also include *shared parameters* not referring to any entity or relation (e.g., the weights of neural layers); shared parameters are generally trained jointly with the KG embeddings.

When the training is over we expect the model to yield high plausibility values for unseen true facts as well; in practice, new plausible facts are identified via either *tail prediction* or *head prediction*. In *tail prediction*, given a head entity h and a relation r, we find the entity that, if used as a tail for $\langle h,\ r,\ ? \rangle$, results in the best ϕ score; *head predictions* are defined analogously. In our formulations, for the sake of simplicity we mostly refer to tail predictions.

In evaluation, head and tail predictions are run on each test fact ranking the *target entity*, i.e., the entity actually featured in the fact, against all the others in $\mathcal{E}$. Ideally, the target entity should obtain rank 1, conveying a correct prediction: e.g, when predicting the tail for head *Dante Alighieri* and relation *citizenship*, we expect the target entity *Florence* to obtain the best score and, thus, tail rank 1. In literature, the ranks obtained across all test facts are combined into global metrics, e.g., Hits@K or Mean Reciprocal Rank. Such metrics convey the overall performance of models without providing any insights on their specific strengths and weaknesses. Finer-grained interpretability methods are thus sorely needed in LP research [10].

2.2 Explainability of Link Prediction Models

Embedding-based LP models tend to be opaque, as they do not keep memory of the effect of the various training facts, and they do not provide any insights on the predictions they yield; even worse, these models tend to defy all the major explainability approaches.

Saliency-based frameworks such as LIME, Anchor, or SHAP [1], have recently gained popularity due to their effectiveness and versatility. These systems explain predictions by identifying which features of the input samples have affected the outcomes the most. This idea implicitly requires the input features to be human-interpretable. In our LP models, though, the input samples are just triplets of embeddings, i.e., non-human-interpretable vectors: saliency-based systems would just identify their most relevant elements, which would not be truly informative for human users.

A different approach interprets predictions by identifying which training samples have influenced them the most: this is ideal for the LP task, where the training samples provide the logical backbone for all the embeddings we learn. The framework by Koh and Liang [7] follows this direction; unfortunately, their computational times have been proven unfeasible in the LP field [9]. A few works propose other ways to estimate the effect of fact additions or removals, mostly in the scope of assessing the robustness of KG embeddings, but they are limited to single-fact perturbations [9, 16]. All in all, the challenge of creating a truly expressive LP explainability framework has remained mostly unaddressed so far.

3 KELPIE OVERVIEW

In this section we discuss the Kelpie framework. We first define its explanation scenarios, and then describe its main methodologies.

3.1 Kelpie Explanations

Kelpie is a local post-hoc framework [5], so it explains specific predictions by solely analyzing the model samples. Furthermore, Kelpie can work in either a *necessary* or a *sufficient* setting: given a tail prediction $\langle h, r, t \rangle$, we define:

- *Necessary Explanation*: it is the smallest set of training facts mentioning h that, if removed, would disable the prediction, i.e. would switch the top ranking tail from t to another entity.
- *Sufficient Explanation*: given a set C of random entities c for which the model does not predict $\langle c,\ r,\ t \rangle$, it is the smallest set of training facts mentioning h that, if added to any $c \in C$, enable the model to predict $\langle c,\ r,\ t \rangle$.

In other words, a necessary explanation consists in all the pieces of evidence mentioning h that allowed us to infer the target tail t. For instance, in the example of the tail prediction $\langle$*Dante Alighieri*, *citizenship*, *Florence*$\rangle$, this would encompass all the *Dante Alighieri* facts that allow us to infer his citizenship: e.g., $\langle$*Dante Alighieri*, *born in*, *Florence*$\rangle$, and $\langle$*Dante Alighieri*, *Prior of*, *Florence*$\rangle$.

Conversely, a sufficient explanation features the h training facts that, if added to any entity c (replacing h with c), switch the top-ranking predicted tail to for c to the same t that is predicted for h. We call such a switch in the predicted tail a *conversion*. In the $\langle$*Dante Alighieri*, *citizenship*, *Florence*$\rangle$ example we may find that transferring the $\langle$*Dante Alighieri*, *born in*, *Florence*$\rangle$ fact to any non-Florentine entities is sufficient to have the model convert their predicted citizenship to *Florence*.

Necessary and sufficient explanations are complementary to each other, so users can choose which one suits their goals best: on the one hand, necessary explanations encompass all the reasons why a specific entity has been predicted in a certain way; on the other hand, sufficient explanations provide general rules that describe which parts of a specific entity "lock" the prediction to explain, and could even extend it to other entities.

3.2 Kelpie Framework

The Kelpie framework supports by design any LP models based on embeddings, and it can be used to extract either necessary or sufficient explanations. As already pointed out Kelpie explanations can have length greater than 1; while adding to their expressiveness, this makes the space of candidate explanations unfeasible to visit in a brute force approach. Hence, as depicted in Figure 1, Kelpie relies on a three-module architecture:

- a *Pre-Filter* identifies the most promising training facts featuring h, and discards the others;
- an *Explanation Builder* visits the space of the resulting fact combinations, enacting early termination policies;
- a *Relevance Engine* actually computes how relevant any selected combination of facts is to the prediction to explain.

We describe these modules in the following sections, with particular focus on the Relevance Engine and its *Post-Training* technique. We report in our online repository extensive experiments for all the main tweakable parameters of our framework.

3.3 Pre-Filter

Given any $\langle h, r, t \rangle$ tail prediction, the role of the Pre-Filter is to discard the presumably least relevant training facts mentioning h in order to prevent combinatorial explosion when their number is too large. The Pre-Filter does this by computing, for each of them, a *promisingness* value conveying its likelihood to be relevant.

Our implementation of promisingness relies on the basic graph topology. Intuitively, the closer two entities are in the graph, the stronger their semantic affinity. Therefore, for any $\langle h,\ s,\ q \rangle$ or $\langle q,\ s,\ h \rangle$ training fact featuring h, we find the shortest non oriented path to t: the shorter the path, the better the fact promisingness. We describe in our online repository experiments with different implementations of promisingness.

The Pre-Filter outputs the set $\mathcal{F}^h_{train}$ of the top-k most promising facts; the k parameter can be tweaked to balance the search space size with the certainty of not discarding any relevant facts.

3.4 Explanation Builder

Given any $\langle h, r, t \rangle$ tail prediction to explain, the Explanation Builder has the role of combining the pre-filtered facts $\mathcal{F}^h_{train}$ into a space of candidate explanations, and of guiding the search in it. In the following we refer to the generic candidate explanation as X, and to the explanation to find, i.e., the minimal effective one, as X^*.

The Explanation Builder first visits all the single-fact candidate explanations, having the Relevance Engine assess their relevance ξ. Then, it progressively increases the size of the visited candidates until one matches our acceptance criteria by exceeding thresholds on ξ; in this case we return it as X^*. In the necessary scenario, this ensures that removing the facts of X^* from the G_{train} would disable the prediction to explain; in the sufficient scenario, it ensures that adding the facts of X^* to any $c \in C$ would convert them.

We make the search for X^* more efficient by enacting early termination policies. Among same-sized candidates, we prioritize those that feature the individually most relevant facts; if we gather evidence that no current-sized candidates are viable explanations, we move to larger sizes. The progressive increase in candidate size ensures the minimality of the X^* we eventually find. If the exploration ends with no X being identified as X^*, we return the most relevant X met so far, adopting a best-effort approach.

3.5 Relevance Engine

Given any $\langle h, r, t \rangle$ tail prediction to explain, the Relevance Engine has the role of measuring the *relevance* for the candidates visited by the Explanation Builder. This corresponds in the necessary scenario

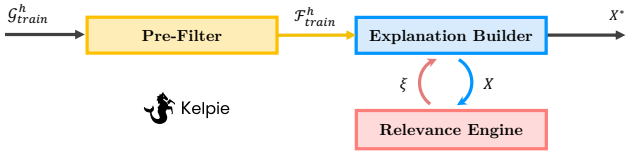

Figure 1: Kelpie framework architecture.

to the effect of removing their facts from h, and in the sufficient scenario to the effect of adding those facts to any $c \in C$.

Since retraining the whole model for each visited candidate explanation X would be unfeasible, we employ a novel ML technique that we call *Post-Training*. To assess the relevance of any X we do not operate directly on the entity h (or the entities $c \in C$); rather, we create a *mimic*, an alternate versions that we initialize by cloning the training facts of h (or c), and that we perturb by removing (or adding) the facts in X. We then compute an embedding for our mimic: (*i*) we initialize it randomly, as for any other entity; (*ii*) we freeze all the other elements in the model (KG embeddings and shared parameters); and (*iii*) we train the mimic embedding alone to optimize the plausibility its training facts. The resulting embedding will behave similarly to how the original entity would, had injected perturbations been present since the beginning. The Post-Training process is remarkably lightweight: it only optimizes one embedding, instead of those of all entities and relations, and only on relatively few facts, instead of the whole training set. In greater detail, given a candidate explanation X we proceed as follows:

- In the *necessary scenario* we post-train a mimic of h without the facts in X. The necessary relevance is formulated in terms of how using the mimic instead of the original h worsens the the tail rank of the $\langle h, r, t \rangle$ tail prediction.
- In the *sufficient scenario* for each entity to convert $c \in C$ we post-train a mimic adding the facts in X to c. The sufficient relevance is formulated in terms of how using the mimics instead of the original c improves the tail ranks of the $\langle c, r, t \rangle$ tail predictions.

4 DEMONSTRATION SCENARIOS

For our demonstration we use the intuitive UI in Figure 2, which allows us to select predictions by different models on different datasets, and to visualize the corresponding Kelpie explanations.

4.1 Demonstration Settings

In our demonstration we provide examples of both *necessary* and *sufficient explanations*. We explain the predictions of 3 models, representing each of the 3 main LP families in literature [10]:

- *ComplEx* [15] is a matrix factorization model that combines the embeddings via bilinear products in the complex space. We use the state-of-the-art implementation by Lacroix *et al.* [8], that relies on a Multiclass Negative Log-Likelihood Loss.
- *ConvE* [4] is a Deep Learning model that combines KG embeddings with a convolutional architecture. We adhere to the original implementation, that relies on a Binary Cross-Entropy Loss.
- *TransE* [3] is a pioneering geometric model that interprets relations as translations in the embedding space. We follow the original model implementation, using a Pairwise Ranking Loss.

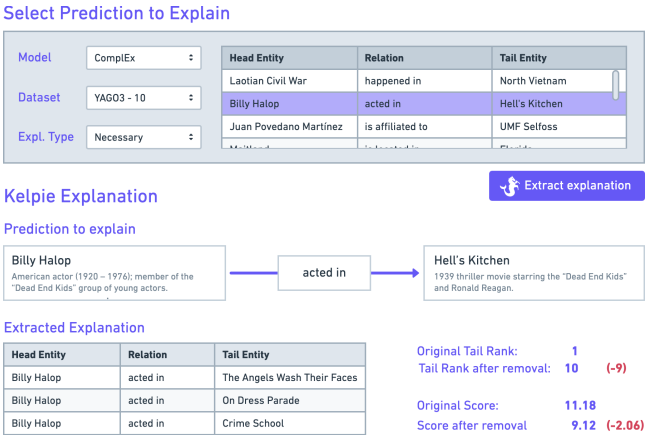

Figure 2: Kelpie demonstration UI.

For each model, we explain predictions performed on all the 5 major datasets in LP literature: *FB15k*, *WN18*, *FB15k-237*, *WN18RR* and *YAGO3-10*. FB15k and WN18 have been extracted from by Bordes *et al.* [3] selecting the facts featuring the richest entities in Freebase and WordNet respectively. It has later been observed that both datasets suffer from test leakage due to the presence of inverse relations; therefore, Toutanova and Chen [13] and Dettmers *et al.* [4] have proceeded to further subsample them, creating the more challenging *FB15k-237* and *WN18RR*. The poor predictive performance displayed by all models on these datasets suggest that only a fraction of their test facts may actually be predictable, given the information provided in training [11]. Finally, *YAGO3-10* has been sampled from the YAGO3 KG by Dettmers *et al.* [4] selecting entities that appear in facts with at least 10 different relations.

In our demonstration, we will allow participants to play around with our interactive UI, letting them choose which predictions yielded by which models, and on which datasets, they would like to interpret. We will actively discuss the returned explanations in the context of the model capabilities and the dataset characteristics.

In most cases our explanations match human intuition: for example the ComplEx YAGO3-10 historical prediction ⟨*Republic of Genoa*, *participated in*, *Battle of Crécy*⟩ has a sufficient explanation in the training fact ⟨*Battle of Crécy*, *happened in*, *Republic of Genoa*⟩. In other cases explanations can be less intuitive: this often reveals the presence of bias in our datasets. For example, in YAGO3-10, the prediction of a person working at a University is generally explained with that person having graduated at the same University. While in the real world graduating at a University does not imply ending up working there, in YAGO3-10 this happens for a large number of entities: our models incorporate such a fictitious pattern, making the resulting predictions questionable, albeit correct.

4.2 Demonstration Goals

Our demonstration has four main purposes:

- showcasing the ability of Kelpie to identify explanations in diverse settings. In our demonstration, we will achieve this by asking the audience to select themselves the models, datasets and predictions to show the explanations for.
- displaying the effectiveness of the extracted explanations: as a way to demonstrate explanation effectiveness, we will report how removing or adding the obtained explanation facts and re-training the whole model would affect the original prediction.
- illustrating how explanations can unveil the presence of bias or of obscure correlations in the original datasets. This can provide useful insights on how to intervene on our datasets to re-balance them and make them adhere to real-world semantics.
- showing how broad, dataset-level correlations are matched from the obtained explanations: for instance we will show how explanations for FB15k and WN18 predictions reflect the prominent presence of inverse relations in these datasets, incentivizing models to just leverage trivial information for their predictions.

5 CONCLUSIONS

We demonstrate the Kelpie explainability framework for embedding-based Link Prediction models. Kelpie explains predictions by identifying which combinations of training facts have enabled them; it supports two complementary scenarios based on the concepts of necessity and sufficiency. In our demonstration we show how our approach can be applied to *any* embedding-learning architecture for LP - a sorely needed quality in such a dynamic research field. Kelpie can highlight which pieces of information our models actually leverage, unveiling the inner reasonings, strengths and weaknesses of current state-of-the-art models.

ACKNOWLEDGMENTS

Our work was partly supported by POR FESR LAZIO 2014–2020 "Gruppi di ricerca 2020" (Project code: POR A0375E0096 - CUP F85F21001540009) and by SEED PNR 2021 grant “FLOWER”.

REFERENCES

[1] F. Bodria, F. Giannotti, R. Guidotti, F. Naretto, D. Pedreschi, and S. Rinzivillo. Benchmarking and survey of explanation methods for black box models. *arXiv preprint arXiv:2102.13076*, 2021.
[2] 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. *Artificial Intelligence in the Life Sciences*, 2022.
[3] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *NIPS*, 2013.
[4] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel. Convolutional 2d knowledge graph embeddings. In *AAAI*, 2018.
[5] 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.
[6] V. Huynh and P. Papotti. A Benchmark for Fact Checking Algorithms Built on Knowledge Bases. In *CIKM*, 2019.
[7] P. W. Koh and P. Liang. Understanding black-box predictions via influence functions. In *ICML*, 2017.
[8] T. Lacroix, N. Usunier, and G. Obozinski. Canonical Tensor Decomposition for Knowledge Base Completion. In *ICML*, 2018.
[9] P. Pezeshkpour, Y. Tian, and S. Singh. Investigating robustness and interpretability of link prediction via adversarial modifications. In *NAACL-HLT*, 2019.
[10] A. Rossi, D. Barbosa, D. Firmani, A. Matinata, and P. Merialdo. Knowledge graph embedding for link prediction: A comparative analysis. *TKDD*, 2021.
[11] 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.
[12] A. Rossi, D. Firmani, P. Merialdo, and T. Teofili. Explaining link prediction systems based on knowledge graph embeddings. In *SIGMOD*, 2022.
[13] K. Toutanova and D. Chen. Observed versus latent features for knowledge base and text inference. In *CVSC*, 2015.
[14] 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.
[15] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard. Complex Embeddings for Simple Link Prediction. In *ICML*, 2016.
[16] 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.