KGEx: Explaining Knowledge Graph Embeddings via Subgraph Sampling and Knowledge Distillation

Vasileios Baltatzis [*]
King's College, London
Imperial College, London
vasileios.baltatzis@kcl.ac.uk

Luca Costabello
Accenture Labs, Dublin
luca.costabello@accenture.com

Abstract

Despite being the go-to choice for link prediction on knowledge graphs, research on interpretability of knowledge graph embeddings (KGE) has been relatively unexplored. We present KGEx, a novel post-hoc method that explains individual link predictions by drawing inspiration from surrogate models research. Given a target triple to predict, KGEx trains surrogate KGE models that we use to identify important training triples. To gauge the impact of a training triple, we sample random portions of the target triple neighborhood and we train multiple surrogate KGE models on each of them. To ensure faithfulness, each surrogate is trained by distilling knowledge from the original KGE model. We then assess how well surrogates predict the target triple being explained, the intuition being that those leading to faithful predictions have been trained on "impactful" neighborhood samples. Under this assumption, we then harvest triples that appear frequently across impactful neighborhoods. We conduct extensive experiments on two publicly available datasets, to demonstrate that KGEx is capable of providing explanations faithful to the black-box model.

1 Introduction

Knowledge graphs are knowledge bases whose facts are labeled, directed edges between entities. Research led to broad-scope graphs like DBpedia [2], WordNet, and YAGO [22]. Countless domain-specific knowledge graphs have also been published on the web, from bioinformatics to retail [10].

Knowledge graph embeddings (KGE) are a family of graph representation learning methods that learn vector representations of nodes and edges of a knowledge graph. They are widely used in graph completion, knowledge discovery, entity resolution, and link-based clustering [20].

Despite achieving excellent trade-off between predictive power and scalability, these neural architectures suffer from poor human interpretability, to the detriment of user trust, troubleshooting, and compliance.

Previous work in knowledge graph representation learning aims at designing natively interpretable KGE models or generating post-hoc explanations for existing knowledge graph embedding models. Nevertheless, the field is still in its infancy and recently proposed explanation methods do not scale beyond toy datasets or do not provide thorough empirical evidence of being faithful to the KGE model being explained.

In this work, we propose KGEx, a post-hoc, local explanation sub-system for KGE models (Figure 1). KGEx works with any existing KGE model proposed in literature: given a target triple predicted with a KGE model, we return an explanation in the form of a ranked list of relevant triples from the training set. We use a combination of subgraph sampling and knowledge distillation that we refine

[*] Work done as an intern at Accenture Labs

Vasileios Baltatzis, Luca Costabello, KGEx: Explaining Knowledge Graph Embeddings via Subgraph Sampling and Knowledge Distillation. *Proceedings of the Second Learning on Graphs Conference (LoG 2023)*, PMLR 231, Virtual Event, November 27–30, 2023.

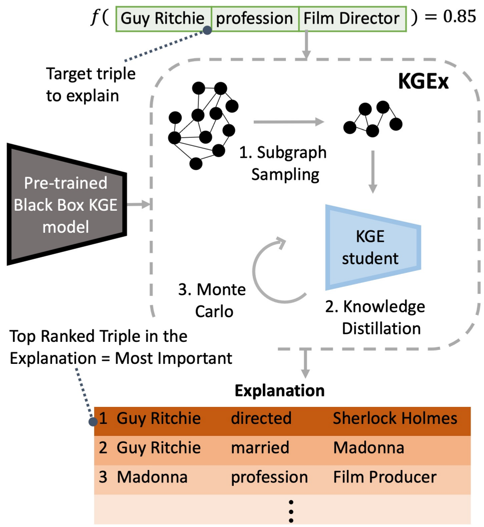
f(Guy Ritchie profession Film Director) = 0.85
Target triple to explain
KGEx
1. Subgraph Sampling
Pre-trained Black Box KGE model
KGE student
3. Monte Carlo
2. Knowledge Distillation
Top Ranked Triple in the Explanation = Most Important
Explanation
1 Guy Ritchie directed Sherlock Holmes
2 Guy Ritchie married Madonna
3 Madonna profession Film Producer
⋮

Figure 1: Overview of the proposed framework. Given a pre-trained KGE model and a target triple, KGEx outputs an explanation for the prediction in the format of a list of ranked triples.

with Monte Carlo sampling. Our experiments show that KGEx provides faithful explanations that can be used beyond toy knowledge graphs.

2 Related Work

Knowledge Graph Embeddings. Knowledge graph embedding models (KGE) are neural architectures designed to predict missing links between entities. TransE [5] is the forerunner of distance-based KGE models, and inspired a number of models commonly referred to as TransX. The symmetric bilinear-diagonal model DistMult [26] paved the way for its asymmetric evolutions in the complex space, ComplEx [25] and RotatE [23]. Some models such as RESCAL [19], TuckER [3], and SimplE [13] rely on different tensor decomposition techniques. Models such as ConvE [7] or ConvKB [18] leverage convolutional layers. Attention is used by [17]. The recent NodePiece uses an anchor-based approach to map entities and relations to a fixed-sized, memory-efficient vocabulary [8]. Recent surveys provide a good coverage of the landscape [4].

Explanations for KGEs. Recent studies address the problem of making KGE architectures more interpretable. MINERVA [6], NeuralLP [27], CTPs [15] integrate rule-based systems to deliver natively interpretable link prediction methods. Although promising, these works do not scale beyond toy datasets. Other works consist instead of *post-hoc* approaches (they operate on black-box, pre-trained KGE models) and are *local* methods, i.e. they explain the prediction of a single instance (i.e. a single missing link between two entities). Gradient Rollback [14] returns a ranked list of influential triples for a target prediction. Such list is computed by storing gradient updates during training, to the detriment of memory footprint and training time overhead. OXKBC creates post-hoc explanations by leveraging entity and path similarities [16]. Earlier work identifies training triples that, when removed, decrease the predicted probability score. ExplainE is grounded on counterfactual explanations and operates on toy knowledge graphs and low-dimensional embeddings [12]. [29] instead randomly perturbs the neighborhood of the target triple, but its design rationale is geared towards adversarial attacks rather than explainability. We consider methods explaining Graph Neural Networks (GNNs), such as [11, 28], to be outside the scope of our work as GNNs and KGEs are different families of architectures, designed to learn on different tasks and data structures. For example, GNNExplainer [28] is not designed for multi-relational graphs, and identifies influential portions of a GNN computation graph and node features - components absent in a KGE architecture.

3 Preliminaries

Knowledge Graph. A knowledge graph $\mathcal{G} = \{(s, p, o)\} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ is a set of triples $t = (s, p, o)$ each including a subject $s \in \mathcal{E}$, a predicate $p \in \mathcal{R}$, and an object $o \in \mathcal{E}$. $\mathcal{E}$ and $\mathcal{R}$ are the sets of all entities and relation types of $\mathcal{G}$.

Knowledge Graph Embedding Models. KGE encode both entities $\mathcal{E}$ and relations $\mathcal{R}$ into low-dimensional, continuous vectors $\in \mathbb{R}^k$ (i.e, the embeddings). Embeddings are learned by training a neural architecture over a training knowledge graph $\mathcal{G}$: an input layer feeds training triples, and a scoring layer $f(t)$ assigns plausibility scores to each triple. $f(t)$ is designed to assign high scores to positive triples and low scores to negative *corruptions*. Corruptions are synthetic negative triples generated by a corruption generation layer: we define a corruption of t as $t^- = (s, p, o')$ or $t^- = (s', p, o)$ where s', o' are respectively subject or object corruptions, i.e. other entities randomly selected from $\mathcal{E}$ [5]. Finally, a loss layer $\mathcal{L}_{KGE}$ optimizes the embeddings by learning optimal embeddings, such that at inference time the scoring function $f(t)$ assigns high scores to triples likely to be correct and low scores to triples unlikely to be true.

Link Prediction. The task of predicting unseen triples in knowledge graphs is formalized in literature as a learning to rank problem, where the objective is learning a scoring function $f(t = (s, p, o)) : \mathcal{E} \times \mathcal{R} \times \mathcal{E} \rightarrow \mathbb{R}$ that given an input triple $t = (s, p, o)$ assigns a score $f(t) \in \mathbb{R}$ proportional to the likelihood that the fact t is true. Such predictions are ranked against predictions from synthetic corruptions, to gauge how well the model tells positives from negatives.

Knowledge Distillation (KD). This method has been introduced to alleviate computational costs and allow knowledge to be transferred from large, complex models (teacher) to smaller, compact ones (student) [9]. Let $\mathcal{X}$ be the input data distribution, with $x_i \sim \mathcal{X}$ distinct samples drawn from that distribution. For brevity, we denote teacher and student representations as $\mathbf{g}_{T,i} = g_T(x_i)$ and $\mathbf{g}_{S,i} = g_S(x_i)$, respectively. Conventional KD can take up the the form of Eq. 1:

$$\mathcal{L}_{KD} = \sum_{x_i \sim \mathcal{X}} l(\mathbf{g}_{T,i}, \mathbf{g}_{S,i}), \tag{1}$$

where l is a loss function such as the Kullback-Leibler divergence, which tries to match the representations of teacher and student for an individual sample x_i.

Relational KD's (RKD) [21] purpose is to transfer the relationship between individual samples from the teacher to the student, as described in Eq. 2:

$$\mathcal{L}_{RKD} = \sum_{(x_i,...,x_n) \sim \mathcal{X}} l_\delta(\phi(\mathbf{g}_{T,i}, ..., \mathbf{g}_{T,n}), \phi(\mathbf{g}_{S,i}, ..., \mathbf{g}_{S,n})), \tag{2}$$

where ϕ is a relational potential function and l_δ is the Huber loss, which is defined in Eq. 3:

$$l_\delta(a, b) = \begin{cases} \frac{1}{2}(a - b)^2 & \text{for } \mid a - b \mid \leq 1, \\ \mid a - b \mid - \frac{1}{2}, & \text{otherwise.} \end{cases} \tag{3}$$

A particular case of relational potential function is the angle-wise relational potential ϕ_A, which can be applied on a triple of samples and is defined as in Eq. 4:

$$\phi_A(\mathbf{g}_{T,i}, \mathbf{g}_{T,j}, \mathbf{g}_{T,k}) = \langle \mathbf{d}_{ij}, \mathbf{d}_{jk} \rangle, \tag{4}$$

where $\mathbf{d}_{ij} = \frac{\mathbf{g}_{T,i} - \mathbf{g}_{T,j}}{\|\mathbf{g}_{T,i} - \mathbf{g}_{T,j}\|_2}$, $\mathbf{d}_{jk} = \frac{\mathbf{g}_{T,j} - \mathbf{g}_{T,k}}{\|\mathbf{g}_{T,j} - \mathbf{g}_{T,k}\|_2}$ and $\langle \cdot, \cdot \rangle$ is the dot product.

4 Methods

Given a pre-trained black-box KGE model and the prediction for an unseen target triple, we generate an explanation for such prediction in the form of a list of training triples ranked by their 'influence' on the prediction. Such explanation is generated by KGEx, our proposed explanation subsystem, which consists of three components. The first step samples a subgraph $\mathcal{H}$ from the original knowledge graph $\mathcal{G}$ in order to limit the search space for possible explanations (section 4.1). To increase the faithfulness of the surrogate model, in the second step we utilize KD to train a new KGE model on the subgraph, while the black-box KGE model we are explaining plays the role of the teacher (section 4.2). Finally, we repeat the second step through a Monte Carlo (MC) process. This is done to rank the triples in the subgraph according to their contribution to the prediction (section 4.3).

4.1 Subgraph sampling

Algorithm 1 Subgraph sampling w/ Predicate Neighborhood

```
Input: target triple $(s^*, p^*, o^*)$, number of predicate neighbors $n$
Output: Subgraph $\mathcal{H}$
$\mathcal{H} \leftarrow \emptyset$
$N_{\mathcal{G}}(s^*) = \{(s, p, o) \in \mathcal{G} | s = s^* \vee o = s^*\}$
$N_{\mathcal{G}}(o^*) = \{(s, p, o) \in \mathcal{G} | s = o^* \vee o = o^*\}$
$N_{\mathcal{G}}(s^*, o^*) = N_{\mathcal{G}}(s^*) \cup N_{\mathcal{G}}(o^*)$        ▷ 1-hop neighborhood of $s^*, o^*$
$\mathcal{H} = \mathcal{H} \cup N_{\mathcal{G}}(s^*, o^*)$
$P_{\mathcal{G}}(p^*) = \{(s, p, o) \in \mathcal{G} | p = p^*\}$        ▷ Triples involving $p^*$
for $i \leftarrow 0$ to $n - 1$ do
    Sample a triple $(\hat{s}, p^*, \hat{o}) \sim P_{\mathcal{G}}(p^*)$
    Get the 1-hop neighborhood $N_{\mathcal{G}}(\hat{s}, \hat{o})$
    $\mathcal{H} = \mathcal{H} \cup N_{\mathcal{G}}(\hat{s}, \hat{o})$
```

Our motivation stems from the explainable AI subfield of surrogate models [1]. Concretely, the idea behind surrogate models is to convert a "black-box" model g_T into a more interpretable "white-model" g_S. The main challenge when trying to design a surrogate model for a KGE model is that KGEs are transductive models and therefore have no inference capabilities, i.e. given a triple t that contains an - unseen during training - entity, there is no function g_T, so that we can infer an output $y = g_T(t)$. Given this limitation we cannot replace g_T with an interpretable g_S, directly. However, we can find a subgraph that will allow us to train such a surrogate model.

We formulate this task as finding the smallest subgraph $\mathcal{H} \subset \mathcal{G}$, which if used to train a KGE model g_S will give a latent space representation to the target triple that is as close as possible to the one assigned by the black-box KGE model g_T, which was trained on the whole knowledge graph $\mathcal{G}$. While searching for $\mathcal{H}$, we are facing a trade-off related to the subgraph size: a larger size promotes fidelity (i.e. faithfulness to the original model), while a smaller size reduces cognitive load and therefore favors interpretability.

The search for the subgraph $\mathcal{H}$ of particular target triple $t^* = (s^*, p^*, o^*)$ can be broken down to two parts. The first part involves retaining the 1-hop neighborhood $N_{\mathcal{G}}(s^*, o^*)$ of the subject s^* and the object o^* of t^*. Using the case depicted in Fig. 1, $N_{\mathcal{G}}$(*Guy Ritchie, Film Director*) would include all triples involving either *Guy Ritchie* or *Film Director*. These are the triples that are in the vicinity of the entities of the target triple and therefore will likely play an important role in the representation that the KGE will learn for these entities. As such, the fact that *(Guy Ritchie, director, Sherlock Holmes)* is important in explaining *(Guy Ritchie, profession, Film Director)*. If we were to retain only this 1-hop neighborhood, we would not be able to incorporate information on any long-range (in terms of hops in the graph) information that might be pivotal for the latent representation of the target triple. Additionally, the 1-hop neighborhood of the subject and the object does not take explicitly into account the predicate p^* of t^*, which could lead to not learning a meaningful representation for p^*. In the running example, the fact *(Madonna, profession, Film Producer)* might seem irrelevant at first sight and would not be part of $N_{\mathcal{G}}$(*Guy Ritchie, Film Director*). However, combined with the fact that *(Guy Ritchie, married, Madonna)* could again lead to the target triple being predicted as positive. In the second part, to alleviate the issues above, we propose two alternatives:

- **Random Walk (RW) Sampling** Apart from the 1-hop neighborhood, we also add a naive random walk of predefined size measured in numbers of steps, which starts from the target triple (see Algorithm 2 in Appendix A).
- **Predicate Neighborhood (PN) Sampling** To ensure that the predicate p^* of t^* is part of the subgraph we randomly sample from $\mathcal{G}$ a predefined number n of triples $t^* = (\hat{s}, p^*, \hat{o})$, which have the same predicate p^*, and include these along with their own 1-hop neighborhoods

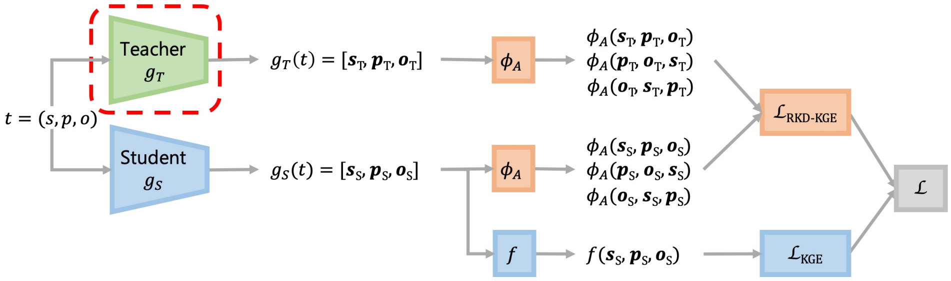
Teacher
g_T
Student
g_S
$t = (s, p, o)$
$g_T(t) = [\mathbf{s}_T, \mathbf{p}_T, \mathbf{o}_T]$
$g_S(t) = [\mathbf{s}_S, \mathbf{p}_S, \mathbf{o}_S]$
ϕ_A
ϕ_A
f
$\phi_A(\mathbf{s}_T, \mathbf{p}_T, \mathbf{o}_T)$
$\phi_A(\mathbf{p}_T, \mathbf{o}_T, \mathbf{s}_T)$
$\phi_A(\mathbf{o}_T, \mathbf{s}_T, \mathbf{p}_T)$
$\phi_A(\mathbf{s}_S, \mathbf{p}_S, \mathbf{o}_S)$
$\phi_A(\mathbf{p}_S, \mathbf{o}_S, \mathbf{s}_S)$
$\phi_A(\mathbf{o}_S, \mathbf{s}_S, \mathbf{p}_S)$
$f(\mathbf{s}_S, \mathbf{p}_S, \mathbf{o}_S)$
$\mathcal{L}_{\text{RKD-KGE}}$
$\mathcal{L}_{\text{KGE}}$
$\mathcal{L}$

Figure 2: A KGE model architecture incorporating the adapted Relational Knowledge Distillation (RKD-KGE).

$N_{\mathcal{G}}(\hat{s}, \hat{o})$ in the subgraph (Algorithm 1). In our example, that would entail sampling triples that involved the predicate *profession*.

4.2 Knowledge Distillation

After sampling the subgraph, we train a KGE model g_S on that. We need to ensure that this model is a faithful surrogate to the black-box model whose predictions we are trying to explain, through some sort of constraint. For this reason, we propose the use of KD as a way to allow the original model to drive the learning process of the surrogate model. Thus, in KD terms, the black-box model plays the role of the teacher, while the surrogate model that of the student, with functions g_T and g_S, respectively. The relational aspect of RKD makes it a natural fit for KGEs. Nevertheless, it is important to note that RKD is applied on individual samples. In mini-batch training, for instance, it will be applied on every possible combination of the samples that constitute the mini-batch. In KGEs, however, the relational aspect between the embeddings is inherent. Given that the training samples are in the form of triples $(s, p, o) \in \mathcal{G}$, the KGE loss is already affecting their embeddings $(\mathbf{s}, \mathbf{p}, \mathbf{o})$ in a relational manner. To accommodate the training of the KGE we adapt the RKD loss, so that it is applied only among the entities and the relation of a particular triple at a time, instead of randomly selected samples. We term this adaptation $\mathcal{L}_{RKD-KGE}$ and it takes the following form (Eq. 5):

$$\begin{aligned}\mathcal{L}_{RKD-KGE} = \sum_{(s,p,o)\in\mathcal{G}} & l_\delta(\phi_A(\mathbf{s}_T, \mathbf{p}_T, \mathbf{o}_T), \phi_A(\mathbf{s}_S, \mathbf{p}_S, \mathbf{o}_S)) \\ & + l_\delta(\phi_A(\mathbf{p}_T, \mathbf{o}_T, \mathbf{s}_T), \phi_A(\mathbf{p}_S, \mathbf{o}_S, \mathbf{s}_S)) \\ & + l_\delta(\phi_A(\mathbf{o}_T, \mathbf{s}_T, \mathbf{p}_T), \phi_A(\mathbf{o}_S, \mathbf{s}_S, \mathbf{p}_S))\end{aligned} \tag{5}$$

The overall loss of the architecture can then be defined as in Eq. 6, with the RKD-KGE part acting as a regularization on the exact embeddings that the KGE part is affecting:

$$\mathcal{L} = \mathcal{L}_{KGE} + \lambda\mathcal{L}_{RKD-KGE} \tag{6}$$

It is important to remind here that the teacher model is pre-trained and its weights are not updated. Instead the teacher representations are only utilized to aid the training of the student. An overview of the student's training procedure can be found in Fig. 2.

4.3 Monte Carlo process

The explanation produced by KGEx is a list of the triples included in the subgraph, ranked by their contribution to the prediction of the target triple. To generate this list, for each target triple that we want to explain and given a pre-trained KGE model (teacher), we train multiple KGE models (students) with the loss defined in Eq. 6. Each of the students is trained on a subset $\mathcal{H}_{mc} \subset \mathcal{H}$ and assigns a *rank* to the target triple. The contribution of each triple $t = (s, p, o) \sim \mathcal{H}$ is analogous to the average rank that was assigned to the target triple on the runs that t was part of $\mathcal{H}_{mc}$.

Table 1: Specifications of the datasets used in experiments

	FB15K-237	WN18RR
Training	272,115	86,835
Validation	17,535	3,034
$\mathcal{T}_1$: Test - Rank 1	100	100
$\mathcal{T}_{rand}$: Test - Random Rank	100	100
Entities	14,541	40,943
Relations	237	11

5 Experiments

We assess the faithfulness of the explanations returned by KGEx. Experiments show that KGEx surrogates are faithful to the original black-box models being explained.

Datasets. We experiment with the two standard link prediction benchmark datasets, WN18RR [7] (a subset of Wordnet) and FB15K-237 [24] (a subset of Freebase). We operate with reduced test sets that include 100 triples. This guarantees reasonable execution times of our experiments, most of which require to retrain a model multiple times as part of the Monte Carlo step, for each triple that we want to explain. We work with two separate test sets: first, to control for the black-box model predictive power, we define $\mathcal{T}_1$, which includes 100 test triples that have been ranked at first place by all the black-box models used in our experiments (see 'Evaluation Protocol' below). In some experiments we also use another test set, $\mathcal{T}_{rand}$, which includes 100 randomly-selected triples, regardless of the assigned rank by the black-box model. Table 1 shows the statistics of all the datasets used.

Evaluation protocol. We measure the faithfulness of explanations generated by KGEx in terms of predictive power discrepancy between black box predictions and predictions generated with KGEx surrogates. We adopt the standard evaluation protocol described by [5]. We predict whether each triple $t = (s, p, o) \in \mathcal{T}$ is true, where $\mathcal{T}$ is either $\mathcal{T}_1$ or $\mathcal{T}_{rand}$. We cast the problem as a learning-to-rank task: for each $t = (s, p, o) \in \mathcal{T}$, we generate synthetic negatives $t^- \in \mathcal{N}_t$ by corrupting one side of the triple at a time (i.e. either the subject or the object). In the standard evaluation protocol, synthetic negatives $\mathcal{N}_t$ are generated from all entities in $\mathcal{E}$. In our experiments, to guarantee a fair comparison between the black-box and the surrogate model, we limit to synthetic negatives created from entities included in the corresponding sampled subgraph $\mathcal{H}$. We predict a score for each t and all its negatives $t^- \in \mathcal{N}_t$. We then rank the only positive t against all the negatives $\mathcal{N}_t$. We report mean rank (MR), mean reciprocal rank (MRR), and Hits at n (where $n = 1, 10$) by filtering out spurious ground truth positives from the list of generated corruptions (i.e. "filtered" metrics).

Implementation Details and Baselines. The KGEx explanation subsystem and the black-box KGE models are implemented using TensorFlow 2.5.2 and Python 3.8[2]. KGE hyperparameter ranges and best combinations are reported in Appendix B. Regarding the KGEx specific hyperparameters, we use Predicate Neighborhood (PN) sampling with 5 neighbors for FB15K-237 and 3 neighbors for WN18RR (see section 5.2), KD coefficient $\lambda = 3$ (see section 5.3). Student models have embedding dimensionality $k = 50$, synthetic negatives ratio $\eta = 2$ and are trained using the Adam optimizer and a multiclass-NLL baseline loss with learning rate=0.1 for 200 epochs. Depending on the size of the subgraph, an explanation might require from 50 to 200 MC runs. A computational complexity analysis can be found in Appendix C. All experiments were run under Ubuntu 16.04 on an Intel Xeon E5-2630, 32 GB, equipped with a Titan XP 12GB.

5.1 Faithfulness: KGE Architectures

The first experiment tests how faithful the KGEx surrogates are to the pre-trained black-box models in terms of predictive performance. We conduct this experiment by leveraging both the $\mathcal{T}_1$ test set (which is produced from the ranks of each black-box that we are explaining) and the $\mathcal{T}_{rand}$ test set.

The results for $\mathcal{T}_1$ are shown on Table 2. Naturally, all black-box models have perfect metrics, by definition. By inspecting the performance of the surrogate models, we can see that they retain a respectable level in MRR of around 0.5 (depending on the black-box) for FB15K-237. It is quite interesting to note that the TransE surrogate is the one which manages to achieve the minimum drop

[2] https://github.com/Accenture/AmpliGraph

Table 2: Faithfulness on $\mathcal{T}_1$ (triples that were ranked 1 by the black-box): comparison between KGE architectures. Worst possible rank is 2712 for FB15K-237 and 370 for WN18RR. Filtered metrics. Best results in bold.

		FB15K-237 - Rank 1 ($\mathcal{T}_1$)				**WN18RR - Rank 1 ($\mathcal{T}_1$)**			
				Hits@				Hits@	
		MR	MRR	1	10	MR	MRR	1	10
TransE		1	1.0	1.0	1.0	1	1.0	1.0	1.0
DistMult		1	1.0	1.0	1.0	1	1.0	1.0	1.0
ComplEx		1	1.0	1.0	1.0	1	1.0	1.0	1.0
KGEx	TransE	**59**	**.55**	.44	**.75**	41	.24	.12	.52
	DistMult	72	.54	**.46**	.69	**5**	**.99**	**.98**	**.98**
	ComplEx	130	.50	.42	.65	8	.87	.84	.92

Table 3: Faithfulness on $\mathcal{T}_{rand}$ (randomly selected triples regardless of the black-box's assigned rank): comparison between KGE architectures. Worst possible rank is 2712 for FB15K-237 and 370 for WN18RR. Filtered metrics. Best results in bold.

		FB15K-237 - Random Rank ($\mathcal{T}_{rand}$)				**WN18RR - Random Rank ($\mathcal{T}_{rand}$)**			
				Hits@				Hits@	
		MR	MRR	1	10	MR	MRR	1	10
TransE		34	.36	.24	.61	**6**	.36	.10	**.85**
DistMult		36	.37	.24	**.64**	21	.55	.47	.70
ComplEx		**27**	**.41**	**.30**	.62	24	**.61**	**.49**	.78
KGEx	TransE	**118**	**.17**	**.10**	**.27**	39	.19	.00	.55
	DistMult	284	.13	.06	.25	**20**	**.43**	**.38**	**.57**
	ComplEx	282	.12	.05	.26	36	.39	.33	.47

in performance from its black-box, across all reported metrics. Turning to WN18RR, we can see even stronger results from the surrogates, and especially from DistMult and ComplEx. DistMult in particular replicates its black-box's almost perfect performance.

The same experiment is conducted on $\mathcal{T}_{rand}$ test set and the results are reported on Table 3. TransE again gets the best results for FB15K-237 with 53% drop in MRR from its black-box, but given the added difficulty of the task, DistMult and ComplEx still manage to get a comparable Hits@10. For WN18RR, similar to $\mathcal{T}_1$, DistMult and ComplEx retain quite high scores across all the metrics. While TransE is a bit lower compared to the other architectures, we can see that compared to its own black-box, it actually manages to stay on a very competitive level in terms of MRR and Hits@10. We have to mention here that the MRR of the TransE black-box in this case is 35-40% lower than DistMult or ComplEx, which is confirmed by similar numbers reported in recent literature. The reason for that is most likely related to the small amount of relations in WN18RR, which might not be properly captured by TransE's architecture. As a result, this affects the TransE surrogate both in $\mathcal{T}_1$ and $\mathcal{T}_{rand}$ and therefore is *not* an issue of the KGEx component but rather the black-box's.

5.2 Impact of Subgraph Sampling

We evaluate predicate neighborhood (PN) and random walk (RW) sampling (Table 4). Both methods contain the 1-hop neighborhood of the subject and the object of the target triple. For PN sampling, we conduct experiments with 3, 5 and 10 predicate neighbors for both datasets. For RW sampling, we use 10, 50 and 100 steps for FB15K-237 and 100, 200 and 500 steps for WN18RR.

Across both datasets PN sampling yields much better results than RW sampling. An interesting finding, is that regardless of method increasing the subgraph size either with more neighbors for PN sampling or with more steps in RW sampling, performance is actually decreasing. This shows that we do not need a very large subgraph to calculate effective embeddings for the target triple, and that including facts likely to be not relevant in the subgraph hurts performance.

Table 4: Subgraph Sampling approaches. KGE architecture used is ComplEx. Filtered metrics. Best results in bold.

	FB15K-237 - Rank 1 ($\mathcal{T}_1$)				
Subgraph Sampling	Subgraph			Hits@	
	Avg size	MR	MRR	1	10
3 Predicate Neighbors	750	113	.40	.32	.53
5 Predicate Neighbors	1783	130	**.50**	**.42**	.65
10 Predicate Neighbors	2742	171	.47	.38	**.66**
10 Random Walk Steps	521	**99**	.23	.09	.43
50 Random Walk Steps	555	124	.21	.11	.36
100 Random Walk Steps	606	150	.20	.10	.37

	WN18RR - Rank 1 ($\mathcal{T}_1$)				
Subgraph Sampling	Subgraph			Hits@	
	Avg size	MR	MRR	1	10
3 Predicate Neighbors	61	8	**.87**	**.84**	**.92**
5 Predicate Neighbors	74	**6**	.84	.81	.90
10 Predicate Neighbors	132	8	.81	.77	.89
100 Random Walk Steps	86	**6**	.68	.57	.87
200 Random Walk Steps	153	8	.63	.49	.84
500 Random Walk Steps	351	23	.55	.41	.79

Table 5: Impact of Knowledge Distillation on KGEx: comparison between KGE architectures. Filtered metrics. Best results in bold.

		FB15K-237 - Rank 1 ($\mathcal{T}_1$)				**WN18RR - Rank 1 ($\mathcal{T}_1$)**			
	Knowledge			Hits@				Hits@	
	Distillation	MR	MRR	1	10	MR	MRR	1	10
TransE		**37**	.54	**.46**	.69	27	.38	.27	.60
Distmult	No	157	.40	.31	.55	7	.98	**.98**	**.98**
ComplEx		189	.37	.28	.55	8	.81	.76	.90
TransE		59	**.55**	.44	**.75**	41	.24	.12	.52
DistMult	Yes	72	.54	**.46**	.69	**5**	**.99**	**.98**	**.98**
ComplEx		130	.50	.42	.65	8	.87	.84	.92

5.3 Knowledge Distillation Effect

We assess the effect of knowledge distillation on the KGEx pipeline. As defined in Eq. 6, the KD component acts as regularization during the student's training. Its contribution is regulated by the KD coefficient λ and, as we see in Fig. 3, performance across models remains fairly stable across different λ. We chose $\lambda = 3$ across experiments, as it gives marginally better results across all models.

Finally, we look at the enhancement in performance that KD offers in Table 5. It is evident that when KD is used, it improves results across all backbone models, compared to standalone students (i.e. models trained on the subgraph without KD). In detail, ComplEx and DistMult outperform their standalone counterparts across all metrics for both datasets with increase in MRR in the 6-13 % region. The outlier in our observations comes from TransE on WN18RR, which is the only case where the standalone model outperforms the KD-based one (MRR=0.38 against MRR=0.24 on $\mathcal{T}_1$). This is *not* a limitation of KGEx though, but rather something that highlights it is indeed working as intended. Given the characteristics of WN18RR (small number of relations), it looks like a smaller subgraph actually favors KGEs, as the standalone model, even though of smaller capacity, outperforms the model trained on the full KG. However, the KD student remains bounded by the knowledge that was distilled by its teacher and therefore remains faithful to the teacher, which is what is desirable. On the other hand, the standalone model leverages to a great extent the favorable topology of the subgraph,

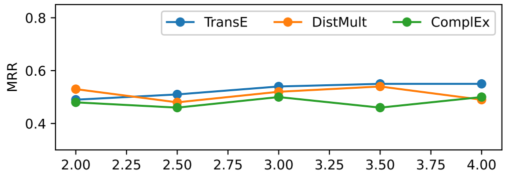

Figure 3: Impact of Knowledge Distillation loss coefficient λ on model performance for FB15K-237.

Table 6: Example explanations for FB15K-237 by KGEx. Each target triple is noted in bold. All triples are predicted as factually correct with a prediction score of 0.99. The top three explanation triples below each target triple are listed in order of importance. The black-box model is ComplEx with embedding dimentionality $k = 350$. The student models' embedding dimentionality is $k = 50$. The subgraph sampling approach is random walk with 10 steps. The number of Monte Carlo runs is 100 and in each run the subgraph of each target triple is partitioned in 10 subsets.

	Walk of fame	**inductee**	**Meryl Streep**
1	Meryl Streep	award	Tony award
2	Julianne Moore	film	Far from heaven
3	84th Academy awards	winner	Meryl Streep

	Ryanair	**headquarters**	**Dublin**
1	Ryanair	currency	Euro
2	Ryanair	phone service	Ireland
3	Dublin	transportation	Air travel

	Jaundice	**symptom of**	**Hepatitis**
1	Jaundice	symptom of	Pancreatic cancer
2	Abdominal pain	symptom of	Hepatitis
3	Jaundice	symptom of	Malaria

	Priyanka Chopra	**ethnicity**	**Punjabis**
1	Punjabis	location	Pakistan
2	Priyanka Chopra	lived	Jharkhand
3	Juhi Chawla	ethnicity	Punjabis

but because there is no connection to the teacher through the loss function, it fails completely to capture any of the teacher's representation abilities, which is what we would expect.

5.4 Example Explanations

We also provide some example explanations in Table 6. These were generated by KGEx using a ComplEx black-box model for target triples from FB15K-237. Explanations for DistMult and TransE can be found in Appendix D. Something that can be observed is KGEx's ability to include triples in the explanation that are beyond the 1-hop neighborhood of the subject and the object of the target triple. Such an example is the explanation triple *(Julianne Moore, film, Far from heaven)*, which explains the target triple *(Walk of fame, inductee, Meryl Streep)* because Meryl Streep and Julianne Moore have collaborated in movies. When explaining the target triple *(Priyanka Chopra, ethnicity, Punjabis)*, we see that the explanation contains relevant information and not facts about the subject's professional life as in the previous example. Finally, in other cases, when the context can be really specific, such as the symptoms example, we see that all the explanation triples are sourced based on the predicate. Even in that case though, although jaundice can be a symptom of various diseases, the ones chosen in the explanation (i.e. pancreatic cancer and malaria) are both correlated with hepatitis.

6 Conclusion

KGEx generates post-hoc, local explanations in the form of a ranked list of triples. We show that the interplay of graph sampling and knowledge distillation reduces the explanation search space while guaranteeing faithfulness to the black-box KGE model being explained. Deploying a Monte Carlo process to rank the explanation triples based on their influence on the prediction prioritizes important and relevant facts. A *limitation* of KGEx is, being a post-hoc method, it does not explain the internals of a KGE model, nor it guarantees a fully transparent-by-design predictive pipeline. However, it is a first step in the direction of designing more interpretable and trustworthy GraphML methods.

Moving forward, we will leverage the modular nature of the framework and propose replacement modules, such as a search-based approach instead of the MC process, to reduce computational burden. Future work will also focus on user studies to gauge the perceived quality of KGEx explanations, by measuring how much they assist human experts on the receiving side of KGE predictions.

References

[1] Ahmed M Alaa and Mihaela van der Schaar. Demystifying black-box models with symbolic metamodels. *NeurIPS*, 2019. 4

[2] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer, 2007. 1

[3] Ivana Balažević, Carl Allen, and Timothy M Hospedales. Tucker: Tensor factorization for knowledge graph completion. *arXiv preprint arXiv:1901.09590*, 2019. 2

[4] Federico Bianchi, Gaetano Rossiello, Luca Costabello, Matteo Palmonari, and Pasquale Minervini. Knowledge Graph Embeddings and Explainable AI. *arXiv preprint arXiv:2004.14843*, 2020. 2

[5] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *NIPS*, pages 2787–2795, 2013. 2, 3, 6

[6] Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In *ICLR*, 2018. 2

[7] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *AAAI*, 2018. 2, 6

[8] Mikhail Galkin, Jiapeng Wu, Etienne Denis, and William L Hamilton. Nodepiece: Compositional and parameter-efficient representations of large knowledge graphs. *arXiv preprint arXiv:2106.12144*, 2021. 2

[9] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. 3

[10] Aidan Hogan, Eva Blomqvist, Michael Cochez, et al. Knowledge graphs. *ACM Computing Surveys (CSUR)*, 2021. 1

[11] Qiang Huang, Makoto Yamada, Yuan Tian, Dinesh Singh, Dawei Yin, and Yi Chang. GraphLime: Local interpretable model explanations for graph neural networks. *arXiv preprint arXiv:2001.06216*, 2020. 2

[12] Bo Kang, Jefrey Lijffijt, and Tijl De Bie. ExplainE: An approach for explaining network embedding-based link predictions. *arXiv preprint arXiv:1904.12694*, 2019. 2

[13] Seyed Mehran Kazemi and David Poole. Simple embedding for link prediction in knowledge graphs. In *Procs. of NeurIPS*. 2018. 2

[14] Carolin Lawrence, Timo Sztyler, and Mathias Niepert. Explaining neural matrix factorization with gradient rollback. In *Procs of AAAI*, 2021. 2

[15] Pasquale Minervini, Sebastian Riedel, Pontus Stenetorp, Edward Grefenstette, and Tim Rocktäschel. Learning reasoning strategies in end-to-end differentiable proving. In *ICML*, 2020. 2

[16] Yatin Nandwani, Ankesh Gupta, Aman Agrawal, Mayank Singh Chauhan, Parag Singla, et al. Oxkbc: Outcome explanation for factorization based knowledge base completion. In *AKBC*, 2020. 2

[17] Deepak Nathani, Jatin Chauhan, Charu Sharma, and Manohar Kaul. Learning attention-based embeddings for relation prediction in knowledge graphs. *arXiv preprint arXiv:1906.01195*, 2019. 2

[18] Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. A novel embedding model for knowledge base completion based on convolutional neural network. In *NAACL*, pages 327–333, 2018. 2

[19] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *ICML*, 2011. 2

[20] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Procs of the IEEE*, 104(1):11–33, 2016. 1

[21] Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. Relational knowledge distillation. *CVPR*, pages 3962–3971, 2019. 3

[22] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Procs of WWW*, 2007. 1

[23] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. In *ICLR*, 2019. 2

[24] Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. Representing text for joint embedding of text and knowledge bases. In *Procs of EMNLP*, 2015. 6

[25] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *ICML*, pages 2071–2080, 2016. 2

[26] Bishan Yang, Scott Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In *ICLR*, 2015. 2

[27] Fan Yang, Zhilin Yang, and William W Cohen. Differentiable learning of logical rules for knowledge base reasoning. In *NIPS*, 2017. 2

[28] Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. GNNExplainer: Generating explanations for graph neural networks. *NeurIPS*, 32:9240, 2019. 2

[29] Wen Zhang, Bibek Paudel, Wei Zhang, Abraham Bernstein, and Huajun Chen. Interaction embeddings for prediction and explanation in knowledge graphs. In *Procs of the ICWSDM*, 2019. 2

KGEx: Explaining Knowledge Graph Embeddings via Subgraph Sampling and Knowledge Distillation: Supplementary material

A Subgraph sampling with random walk

Algorithm 2 Subgraph sampling w/ Random Walk

```
Input: target triple $(s^*, p^*, o^*)$, number of random walk steps $n$
Output: Subgraph $\mathcal{H}$
$\mathcal{H} \leftarrow \emptyset$
$N_\mathcal{G}(s^*) = \{(s,p,o) \in \mathcal{G} | s = s^* \vee o = s^*\}$
$N_\mathcal{G}(o^*) = \{(s,p,o) \in \mathcal{G} | s = o^* \vee o = o^*\}$
$N_\mathcal{G}(s^*, o^*) = N_\mathcal{G}(s^*) \cup N_\mathcal{G}(o^*)$          ▷ 1-hop neighborhood of $s^*, o^*$
$\mathcal{H} = \mathcal{H} \cup N_\mathcal{G}(s^*, o^*)$
$(s_o, p_o, o_o) = (s^*, p^*, o^*)$          ▷ Initialize random walk origin
for $i \leftarrow 0$ to $n - 1$ do
    Sample a triple $(\hat{s}, \hat{p}, \hat{o}) \sim N_\mathcal{G}(s_o, o_o)$
    $\mathcal{H} = \mathcal{H} \cup \{(\hat{s}, \hat{p}, \hat{o})\})$
    $(s_o, p_o, o_o) = (\hat{s}, \hat{p}, \hat{o})$          ▷ Update origin
```

B Hyperparameter search

We experiment with three popular KGE architectures: TransE, DistMult, ComplEx. For each of them we replicated SOTA results by carrying out extensive grid search, over the following ranges of hyperparameter values: embedding dimensionality $k = [200 - 500]$, with a step of 50; baseline losses={negative log-likelihood, multiclass-NLL, self-adversarial}; synthetic negatives ratio $\eta = \{20, 30, 40, 50\}$; learning rate= $\{1\text{e}{-4}, 5\text{e}{-5}, 1\text{e}{-5}\}$; epochs= $[500 - 2000]$, step of 500; L2 regularizer, with weight $\gamma = \{1\text{e}{-3}, 1\text{e}{-4}, 1\text{e}{-5}\}$. The best loss for all models was the multiclass-NLL and the best regularization weight $\gamma = 1\text{e}{-4}$. The best combinations for the rest of the hyperparameters are shown on Table 7

Table 7: Best hyperparameter combinations for baseline black-box models.

	FB15K-237				WN18RR			
	k	η	lr	epochs	k	η	lr	epochs
TransE	400	30	1e-4	1000	350	30	1e-4	2000
Distmult	300	50	5e-5	1000	350	30	1e-4	2000
ComplEx	350	30	5e-5	1000	200	20	5e-5	2000

C Computational complexity

The subgraph sampling step has a complexity $\mathcal{O}(|\mathcal{G}| + n)$, where $|\mathcal{G}|$ is the number of triples in the entire graph $\mathcal{G}$ (worst case we have to examine all triples in the graph for the initial 1-hop neighborhood calculation), and n is a hyper-parameter (number of random walk steps or triples from the predicate neighborhood - each loop step being constant in time).

The knowledge distillation step requires training a student KGE model for each subgraph sampled by the step above. The computational complexity cost to train a KGE model with our $\mathcal{L}_{RKD-KGE}$ loss is $O(E|\mathcal{H}|\eta k)$, where E are the number of epochs, $|\mathcal{H}|$ is the number of triples in the training graph $\mathcal{H}$, η is the number of synthetic negatives per positive, k is the embedding dimensionality (the KGE scoring function computation requires element-wise operations on the embeddings). This step also requires evaluating the rank of the target triple against a number of synthetic negatives (i.e. the KGE learning-to-rank evaluation protocol). This has a complexity of $\mathcal{O}(k\eta_{eval})$, where η_{eval} is the number of synthetic negatives used during the learning-to-rank evaluation (which is at most as large as the number of entities in the sampled subgraph).

Overall, considering the Monte Carlo process that KGEx involves, we have a complexity of $\mathcal{O}(|\mathcal{G}| + n + mE|\mathcal{H}|\eta k)$ as the dominant term, where m is the number of Monte Carlo iterations. KGEx is therefore linear with the number of triples in the knowledge graph $|\mathcal{G}|$.

It is worth noting that the sampled training sets for the students are always orders of magnitude smaller than the entire graph $\mathcal{G}$ and adopting appropriate values for η and E is important to guarantee appropriate response times.

The effect of k on computational complexity is addressed by training on GPU architectures, ideal for element-wise operations between vectors.

D Example Explanations for DistMult and TransE

The DistMult explanations are similar to the ones of ComplEx with explanations incorporating triples beyond the 1-hop neighborhood as in the case of the target triple *(Jaundice, symptom of, Hepatitis)*. The explanations for TransE show some interesting findings as well, since in two of the examples (*(Ryanair, headquarters, Dublin)* and *(Priyanka Chopra, ethnicity, Punjabis)*) the black-box model actually predicted the target triples as factually incorrect. If we take a closer look at the explanations generated for *(Priyanka Chopra, ethnicity, Punjabis)* we can see that the TransE black-box model in this case is focusing on the subject's professional life and therefore an incorrect prediction seems reasonable.

Table 8: Example explanations for FB15K-237 by KGEx. Each target triple is noted in bold. All triples are predicted as factually correct with a prediction score of 0.99. The top three explanation triples below each target triple are listed in order of importance. The black-box model is DistMult with embedding dimentionality $k = 300$. The student models' embedding dimentionality is $k = 50$. The subgraph sampling approach is random walk with 10 steps. The number of Monte Carlo runs is 100 and in each run the subgraph of each target triple is partitioned in 10 subsets.

	Walk of fame	**inductee**	**Meryl Streep**
1	Meryl Streep	film	Kramer gegen Kramer
2	Meryl Streep	award	Kramer gegen Kramer
3	Meryl Streep	nominated	BAFTA

	Ryanair	**headquarters**	**Dublin**
1	Dublin	country	Ireland
2	Ireland	contains	Dublin
3	Ryanair	phone service	Ireland

	Jaundice	**symptom of**	**Hepatitis**
1	Jaundice	symptom of	Pancreatic cancer
2	Anorexia	symptom of	Hepatitis
3	Anorexia	symptom of	Pancreatic cancer

	Priyanka Chopra	**ethnicity**	**Punjabis**
1	Hrithik Roshan	ethnicity	Punjabis
2	Priyanka Chopra	lived	Uttar Pradesh
3	Priyanka Chopra	lived	Cedar Rapids

Table 9: Example explanations for FB15K-237 by KGEx. Each target triple is noted in bold. The triples *(Walk of fame, inductee, Meryl Streep)* and *(Jaundice, symptom of, Hepatitis)* are predicted as factually correct with a prediction score of 0.99, while the triples *(Ryanair, headquarters, Dublin)* and *(Priyanka Chopra, ethnicity, Punjabis)* as factually incorrect with prediction scores 0.18 and 0.20 respectively. The top three explanation triples below each target triple are listed in order of importance. The black-box model is TransE with embedding dimentionality $k = 400$. The student models' embedding dimentionality is $k = 50$. The subgraph sampling approach is random walk with 10 steps. The number of Monte Carlo runs is 100 and in each run the subgraph of each target triple is partitioned in 10 subsets.

	Walk of fame	**inductee**	**Meryl Streep**
1	Meryl Streep	nominated	Angels in America
2	Meryl Streep	award	Tony award
3	Meryl Streep	film	Doubt

	Ryanair	**headquarters**	**Dublin**
1	Ryanair	phone service	Earth
2	Ryanair	phone service	UK
3	Dublin	country	Ireland

	Jaundice	**symptom of**	**Hepatitis**
1	Jaundice	symptom of	Cirrhosis
2	Anorexia	symptom of	Bladder cancer
3	Anorexia	symptom of	Hepatitis

	Priyanka Chopra	**ethnicity**	**Punjabis**
1	Priyanka Chopra	profession	Singer
2	Punjabis	location	Pakistan
3	Filmfare award	award	Priyanka Chopra