OxKBC: Outcome Explanation for Factorization Based Knowledge Base Completion

Anonymous authors

Abstract

State-of-the-art models for Knowledge Base Completion (KBC) for large KBs (such as FB15k and YAGO) are based on tensor factorization (TF), e.g., DistMult, ComplEx. While they produce good results, they cannot expose any rationale behind their predictions, potentially reducing the trust of a user in the outcome of the model. Previous works have explored creating an inherently explainable model, e.g. Neural Theorem Proving (NTP), DeepPath, MINERVA, but explainability in them comes at the cost of performance. Others have tried to create an auxiliary explainable model having high fidelity with the underlying TF model, but unfortunately, they do not scale well to large KBs. In this work, we propose OxKBC – an Outcome eXplanation engine for KBC, which provides a post-hoc explanation for every triple inferred by a (uninterpretable) factorization based model. It first augments the underlying Knowledge Graph by introducing weighted edges between entities based on their similarity given by the underlying model. It then defines a notion of human-understandable explanation paths along with a language to generate them. Depending on the edges, the paths are aggregated into second–order templates for further selection. The best template with its grounding is then selected by a neural selection module that is trained with minimal supervision by a novel loss function. Experiments over Mechanical Turk demonstrate that users overwhelmingly find our explanations more trustworthy compared to rule mining.

1. Introduction

Knowledge bases (KBs) store facts in the form of triples $r(s, o)$, which indicate that the head entity $s$ is related to the tail entity $o$ via the relation $r$, e.g., $capitalOf(WashingtonDC, USA)$. These KBs are typically incomplete [Bollacker et al., 2008] and the task of KB Completion (KBC) aims to infer new facts from a given KB.

Many state-of-the-art models for KBC on large-scale KBs use tensor factorization ($TF$) [Sedghi and Sabharwal, 2018, Lacroix et al., 2018, Jain et al., 2018, Xue et al., 2018]. They learn embeddings for each entity and relation in a continuous vector space and define a scoring function to evaluate the validity of a given triple. At test time, given a query $r(s, ?)$ they output a ranked list of entities $o$ by sorting the model scores for $r(s, o)$ for all possible tail entities. Though $TF$ models achieve excellent task performance, they are inherently opaque – it is very difficult to uncover a rationale
for why the model scores an entity high for a query, because neither the dimensions nor the scoring function are human interpretable. Since users are often the final consumers of a KB (e.g., via a KB-based QA task, or an entity retrieval task [Xiao et al., 2016]), we believe providing a rationale behind a prediction to them is an important sub-task for building trust in these systems.

**Contributions:** We present OXKBC: a post-hoc Outcome eXplanation engine, which is faithful to a given TF-based KBC model \( M \). It works on the underlying Knowledge Graph, \( KG \), corresponding to the facts given in the Knowledge Base, \( KB \). It first augments the \( KG \) with weighted edges between entities based on their similarity as defined by the underlying model \( M \). It then posits that an explanation for a prediction \( o \) for query \( r(s, ?) \) is a path between \( s \) and \( o \) in this augmented graph.

Similar paths are grouped into second-order templates and OXKBC trains a neural template selection module for explaining a given prediction. We define novel unsupervised and semi-supervised loss functions, so that this module can be trained with no or minimal training data. OXKBC outputs the highest scoring path within the selected template as an explanation to the user. We evaluate OXKBC explanations via a user study over Amazon Mechanical Turk. We find that workers overwhelmingly prefer OXKBC explanations compared to rule-mining techniques proposed earlier.

2. Related Work

**KBC:** Other than \( TF \) based models, research has also explored path based approaches for KBC. They use random walk with restarts [Lao and Cohen, 2010] and more recently, reinforcement learning [Das et al., 2018, Xiong et al., 2017] to construct multi-hop paths in the knowledge graph and use those to make predictions. The paths may offer natural interpretability [Nickel et al., 2016], but the performance of these models is significantly lower than \( TF \), especially on large KBs like FB15K [Bordes et al., 2013], and YAGO [Suchanek et al., 2007]. [Rocktäschel and Riedel, 2017] inductively learn first order rules to generate paths by using a differentiable version of backward chaining. Unfortunately, it does not scale beyond 10K facts in KB.\(^1\) In contemporaneous work, [Minervini et al., 2020] speed up neural theorem proving for this task. Unfortunately, we could not get their method to complete even after a week of continuous computation on FB15K, suggesting that their method still is not scalable to large KBs. Earlier, research had also considered symbolic Horn-clause style inference [Lao and Cohen, 2010, Schoenmackers et al., 2010], though these are no longer competitive with modern neural methods.

\( TF \) methods, like DistMult [Yang et al., 2015] and ComplEx [Trouillon et al., 2016], calculate the score \( S^M(s, r, o) \), of a triple \( r(s, o) \) via a latent tensor factorization over entity embeddings \((\vec{s}, \vec{t})\) and relation embeddings \((\vec{r})\). Key idea in these methods is to learn embeddings such that the score \( S^M \) of the facts present in the \( KB \) is high. In this case, DistMult [Yang et al., 2015] aligns relation embeddings to the Hadamard product of the embeddings of head and tail entities. It uses \( S^M(s, r, o) = \sigma^T(\vec{s} \bullet \vec{r}) \sigma(\vec{t}_s^T \vec{s}) \sigma(\vec{t}_o^T \vec{o}) \), where \( \sigma \) is the sigmoid function. The subscript \( w \)

\(^1\) https://www.akbc.ws/2017/slides/sebastian-riedel-slides.pdf
\(^2\) Code taken from https://github.com/dair-iitd/KBI
denotes the world knowledge embedding, \( t \) denotes the type embedding of an entity and \( ts \) and \( to \) denote head and tail type embeddings of a relation, respectively.

**Explainability in KBC:** There are broadly two approaches for explainable AI. One is to design a transparent model, which can inherently reveal its functioning (e.g., linear models, decision trees, Horn-clause inference for KBC). Unfortunately, transparent models often result in compromised performance in general [Weld and Bansal, 2018], as well as specifically for KBC (as discussed earlier in this section). An alternative is to provide post-hoc explanations for the behaviour of a high-quality black box model [Wachter et al., 2017]. This approach is further divided into two categories: ‘Model Explanation’ (MX) and ‘Outcome Explanation’ (OX) [Costabello et al., 2019].

In MX, once the black-box model is learnt, a global explainable auxiliary model is built using the predictions of the black-box model such that the auxiliary model has high fidelity with the original black-box model. **Rule mining** [Yang et al., 2015] is an MX approach, which mines first order rules based on embeddings learnt by \( M \). It identifies paths (of length \( \leq 3 \)) using the notion of support and confidence borrowed from the frequent itemset mining literature [Pei et al., 2000]. Search space is pruned by considering only those relations which are close to each other in the embedding space. The extracted first order rules may be used to explain a prediction. We compare against this approach in our experiments. The other MX approach is known as the pedagogical approach – it trains a global auxiliary interpretable model (weighted Horn clause rules) from the output of the non-interpretable but accurate TF model [Gusmão et al., 2018]. Its experiments were on a toy dataset with only 13 relations, and unfortunately, we were unable to scale this to the size of our dataset (see Appendix).

OX, attempts to achieve a higher fidelity between explainable model and \( M \) by creating local model approximations around each prediction independently, e.g. [Ribeiro et al., 2016]. This approach often trains a new classifier for each prediction, which is slow if a large number of predictions need to be explained. In response, OxKBC takes a novel OX approach, which decomposes the explanation engine into two components: a global trained model that, for a given prediction, selects the most plausible second-order explanation template (which quantifies over both relations and entities), and an explanation generator that outputs the best instantiation of the template for this prediction. This approach scales better, since the trained model is global and easy to train, and no retraining is required per prediction.

**3. OxKBC: The Outcome Explanation Engine**

We are given a knowledge base of facts, \( KB \), with relations \( R \) and entities \( E \), and a model \( M \), trained on \( KB \). \( M \) scores each triple \( r(s,o) \) using the score \( S^M(s,r,o) \). The goal of OxKBC is to show an end-user a reason why \( M \) predicts a tail entity \( o \) for the query \( r(s,?) \). It does so by finding an explanation path between \( s \) and \( o \) which entails the given relation \( r \).

**3.1 Language for generating Explanation Paths**

A KB can be seen as a knowledge-graph \( KG \), where there is a directed edge from \( s \) to \( o \) labeled by \( r \) for every KB fact \( r(s,o) \). We now define the notion of an explanation path in this \( KG \). Once the model \( M \) is learnt, we augment \( KG \) with additional edges that correspond to similarity between any two entities \( e \) and \( e' \). We represent such an edge as \( \sim_M (e,e') \) and call them ‘entity similarity edges’. Here, the operator \( \sim_M \) represents that \( M \) considers these two entities similar. We call the original edges of \( KG \) as ‘relation edges’. An explanation path, \( P(s,o) \), is a path between two entities, \( s \) and \( o \), in the augmented \( KG \). Here we call \( s \) as the \( Head(P) \) and \( o \) as the \( Tail(P) \). A path is represented
as a sequence of edges concatenated by a ",". Following grammar generates an explanation path recursively:

1. **Relation Edges**: \( P \leftarrow r(s, o) \ \forall \ r(s, o) \in KB \)
2. **Pre-fix a similarity edge**: \( P \leftarrow \sim_M (s', s), P_1 \ s.t. \ Head(P_1) = s. \)
3. **Post-fix a similarity edge**: \( P \leftarrow P_1, \sim_M (o', o) \ s.t. \ Tail(P_1) = o. \)
4. **Concatenation of two paths**: \( P \leftarrow P_1, P_2 \ s.t. \ Tail(P_1) = Head(P_2). \)

\( P_1 \) and \( P_2 \) on the R.H.S. themselves denote explanation paths generated using the same rules recursively. An explanation path may contain both types of edges: ‘relation edges’ and ‘entity similarity edges’. For each entity similarity edge between two entities \( e \) and \( e' \), we may define its weight as \( \text{sim}_M(e, e') \) where \( \text{sim}_M \) function captures the degree of similarity between the entities as given by the underlying model \( M \). This is easy to obtain from any TF model, e.g., for TypedDistMult [Jain et al., 2018], we use \( \cos(\vec{r}_{w}, \vec{r}_{w}')\cos(\vec{r}_{ts}, \vec{r}_{ts}')\cos(\vec{r}_{to}, \vec{r}_{to}') \), where \( \cos \) is the cosine similarity.

For a relation edge, we consider its weight as \( 1 \) as we include relation edges for only those triples that exist in KB. Another alternative is to use the model score \( S_M(s, r, o) \) here as well, like similarity edges, and that would make all our edges probabilistic. But we leave that for future work. Using this notion of edge weights, we define \( \text{EdgeScore}(P) \) of a path \( P \) as product of all the edge weights in it. For \( P \) to be a good explanation, its \( \text{EdgeScore} \) should be high.

Next, we define an operator, \( \text{RelComposition}(P) \), which performs a composition of all the relations in the path \( P \). It returns a vector in the same space as the relation embeddings learnt by the model. We follow previous work to use a Hadamard product of the embeddings of all relations in the path [Guu et al., 2015]. For example, for \( P(s, o) = r_1(s, u_1), r_2(u_1, o) \), \( \text{RelComposition}(P) = r_1 \odot r_2 \).

Now, let the model \( M \) predict an entity \( o \) for the query \( r(s, ?) \). To quantify the plausibility of an explanation path \( P(s, o) \) for the prediction \( r(s, o) \), we define a plausibility score as:

\[
\text{Plausibility}(P(s, o), r(s, o)) = \text{sim}_M(\text{RelComposition}(P(s, o)), r) \cdot \text{EdgeScore}(P) \tag{1}
\]

In the above equation, \( \text{EdgeScore}(P) \) captures all the entity similarities in the path \( P \) and \( \text{sim}_M(\text{RelComposition}(P), r) \) captures whether composition of relations in the path entail the given relation \( r \) or not. \( \text{sim}_M \) function is as defined earlier.

Task of OXKBC is to select the most plausible explanation path. However, one can easily observe that Plausibility score might not be comparable for two paths of unequal lengths. Even for two equal length paths, it could be difficult to compare due to different edge types. For example, consider two paths \( P_1 \) and \( P_2 \) with length two but different edge types: \( P_1(s, o) = \sim_M (s, u_1), r_1(u_1, o) \) and \( P_2(s, o) = r_1(s, v_1), r_2(v_1, o) \). \( P_1 \) has a similarity edge followed by a relation edge and \( P_2 \) has two relation edges. Clearly Plausibility\( (P_1, r(s, o)) = \text{sim}_M(s, u_1) \cdot \text{sim}_M(r_1, r) \) and Plausibility\( (P_2, r(s, o)) = \text{sim}_M(\text{RelComposition}(r_1, r_2), r) \) are not comparable as former involves product of two similarities whereas the latter has only one term. To resolve such discrepancies, we define the notion of a second–order template which aggregates comparable explanation paths. Once we have defined a template, then OXKBC would be left with the job of selecting the most appropriate template and ground it to generate an explanation.

### 3.2 Templates: Aggregation of similar Explanation Paths

Depending on the sequence of edge types in it, we classify an explanation path into a second–order template in which we quantify over both relations and entities. Each template corresponds to a fixed
sequence of edge types. This aggregation is required because the entire space of explanation paths is huge and it is not feasible to compare all of them. Hence, to operationalize the selection process, we aggregate them into templates. Some of the templates used to explain the prediction $o$ for a query $r(s, ?)$ are enumerated below:

| T1. Relation Similarity: $r'(s, o) \in KB$ and $r \sim_M r'$ |
| T2. Entity Similarity: $r(s', o) \in KB$ and $s \sim_M s'$ |
| T3. Entity & Relation Similarity: $s \sim_M s'$ and $r'(s', o) \in KB$ and $r \sim_M r'$, $r \neq r'$ |
| T4. Two Length Relation Similarity: $r_1(s, u_1) \in KB$ and $r_2(u_1, o) \in KB$ and $r \sim_M$ RelComposition$(r_1, r_2)$ |

We note that while other works have considered rules based on relation similarity earlier, e.g. [Yang et al., 2015] mine first order one length and two length rules (T1 and T4 respectively), we are the first one to use the notion of entity similarity for generating explanations. Intuitively, entity similarity captures an aggregation of many 2 length paths: a high similarity score between $e$ and $e'$ implies that they can be used interchangeably, i.e. $\exists r \in R$ and $u \in E$ s.t. $r(e, u), r(e', u) \in KB$, i.e. $P(e, e') = r(e, u), s(u, e')$ is a path in KB if $s = r^{-1}$ is also in the set of relations $R$, and this holds for many choices of $r$ and $u$, implying existence of many such paths. While rule mining may discover just one path between $e$ and $e'$, entity similarity may hop directly using an entity similarity edge, e.g. for second fact in Table 3.

We also note that since our templates are second order, they have the representation power to subsume first order rules. For example, T1, which infers that $r(s, o)$ because $r'(s, o) \in KB$ and "$r$ is similar to $r'$" subsumes one length rule "$r' \Rightarrow r$" as learnt in many other rule mining systems.

While annotating the data, we observed that in few cases $o$ is a prediction for a given query $r(s, ?)$ because in the KB it is the most frequent entity for the relation $r$. For example, most of the websites $(s)$ in the KB are in ‘Language’ $(r)$ ‘English’ $(o)$, and hence model learns to predicts ‘English’ whenever asked for ‘Language’ of any website. To account for such explanations, we consider a frequency based template as well:

| T5. Frequent for Relation: o is a highly frequent tail entity for the relation r in KB |

These templates may not be able to explain every prediction, hence, we also introduce a default template (T0) that corresponds to ‘no explanation’. Table 3 lists sample explanations (generated by OXKBC) on FB15K test data along with template IDs.

**Template Scoring:** The Compatibility function between a path $P$ and a prediction $r(s, o)$ as defined in equation 1 offers a natural way of scoring an explanation path $P$ for the given prediction. Since a given similarity based template represents comparable paths with same sequence of edge types in it, we can quantify it with the explanation path having maximum score. Accordingly, we define template score, $T_i$, for a similarity based template $T_i$ as:

$$T_i(s, r, o) = \max_{P \in Ti} Compatibility(P, r(s, o)) \quad \forall i \in \{1, \ldots, 4\}$$

(2)

Inspired by the notion of selecting the best amongst all paths in a given similarity based template, we define score of frequency based templates $T_5$ as the normalized frequency of the corresponding relation, tail entity pair $(r, o)$ in $KB$:

$$T_5(s, r, o) = \frac{|\{s' : r(s', o) \in KB\}|}{|\{(s', o') : s', r(s', o') \in KB\}|}$$

(3)
These goodness scores act as important features in the template selection module (section 3.3). Once a template is selected, an explanation in English is produced as shown in Table 3 in the appendix. For templates T1-T4, we ground them with the entities/relations in the explanation path corresponding to the argmax of the Compatibility score.

3.3 Selection Module

OxKBC’s selection module (SM) decides which template to select for explaining a given prediction. It uses a 2 layer MLP for this task. For each template $i$, it takes an input feature vector and outputs a score $S_{SM}^{i}$, representing SM’s belief that this template is a good explanation for the prediction. The scores $S_{SM}^{i}$ are converted into probabilities $P_{SM}^{i}$ through a softmax layer. OxKBC chooses the template with the highest probability.

**Input Features**: For a given query $r(s, ?)$ and a template $i$, we compute template goodness scores $T_{i}(s, r, u) \forall u \in E$. This defines a distribution of goodness scores over tail entities for a given input query $r(s, ?)$. To explain a prediction $r(s, o)$, we construct a feature vector such that it captures the relative score $T_{i}(s, r, o)$ of $o$ w.r.t the distribution $T_{i}(s, r, u)$ over $u \in E$. We claim that this feature vector has all the information to decide if this template is good for explaining the given prediction or not. It maintains the distribution level global features w.r.t. the query $r(s, ?)$: max, mean, std deviation of the distribution, and prediction specific features: score, rank and other statistics of the specific prediction $o$. See Table 6 in Appendix for exact definition of features. Since we do not have goodness scores for T0, SM considers its features as trainable parameters.

**Training**: Because manual labeling of explanations is a time-consuming task, we primarily train SM in unsupervised and semi-supervised settings. For completeness, we also report on a fully supervised setting on our small annotated dataset. For the unsupervised setting, the key insight is that for most triples known to be true, we may expect some explanation, but we do not know which template to select. However, for a randomly selected negative sample, the selection module must always select T0 as we do not expect any explanation for such a random sample. Therefore in the absence of template level annotation, we rely on distant supervision from this weak signal. SM formulates it in a multi-instance learning (MIL) framework [Dietterich et al., 1997] (see Appendix for details on architecture), where feature vector for each template is an instance, and supervision is at the bag level (positive or negative labels on a triple).

**Loss Function**: We operationalize this through an unsupervised loss function. When classifying a positive triple, SM gains reward ($R_{pos}$) proportional to the highest template probability amongst templates other than T0. To account for the case that sometimes no template might be available even for a correct triple, SM gets a lower reward even for selecting T0 (eq. (4)). For negative samples, the module is penalized ($R_{neg}$) if it selects any template other than T0 – it gets a negative reward proportional to total probability of all the explanation templates other than T0, and positive reward proportional to probability of T0 (eq. (5)).

\[ R_{pos} = \rho_{pos}P_{0}^{SM} + \max_{i \neq 0} P_{i}^{SM} \quad (4) \]
\[ R_{neg} = P_{0}^{SM} - \rho_{neg} \sum_{i \neq 0} P_{i}^{SM} \quad (5) \]
\[ L_{\text{unsup}} = -R_{pos} - R_{neg} \quad (6) \]
\[ L_{\text{semi}} = L_{\text{unsup}} + \lambda_{1} L_{\text{sup}} + \lambda_{2} L_{\text{PR}} \quad (7) \]

3. We use a very small set of 20 labeled samples as dev set for hyperparameter tuning.
In semi-supervised setting, we assume access to a small train set of labeled explanations, i.e., the correct template for a fact is annotated. We add a supervised loss term ($L_{sup}$), which is the standard cross-entropy loss over the set of labeled examples. Additionally, we project the template label distribution from labeled set into unlabeled data through posterior regularization [Ganchev et al., 2010]. We add a KL divergence loss ($L_{PR}$) between labeled probability distribution over templates in train set and learned distribution. All hyperparameters $\rho$s and $\lambda$s are tuned on a small devset.

4. Experiments

The goal of our experiments is to answer the following research questions: (1) What is the benefit of adding a small amount of supervision on the overall performance of our model? (2) How good are our explanations compared to existing approaches? To answer the first question, we perform an internal evaluation using the annotations made by the authors. To answer the later, we perform a human evaluation using workers on Mechanical Turk. Next, we describe our experiments in detail.

Datasets: Our experiments use the standard KBC datasets, FB15K [Bordes et al., 2013] and YAGO3-10 [Dettmers et al., 2018]. FB15K is a subset of the original Freebase [Bollacker et al., 2008] and has 592,213 facts with 14,951 entities and 1,345 relations. A large fraction of this knowledge base describes facts about actors, awards, movies and sports. YAGO3-10 is a subset of YAGO [Suchanek et al., 2007] and has 1,089,040 facts with 123,128 entities and 37 relations. Most of the facts describe attributes like citizenship, gender, profession, etc. for various people.

Models: We use TypedDistMult [Jain et al., 2018] as our underlying tensor factorization model $M$, since its code is freely available and it has near state-of-the-art performance on these datasets. To train it, we retain the exact train, dev and test folds used in previous work.

Training Data: For training SM, we randomly select 100 samples from the original dev set, where $M$ makes correct prediction at the top. Two authors annotate each sample with the template producing the best explanation with inter-annotator agreement (Cohen’s Kappa) of 0.87, and resolve conflicts through discussion. This labeling took about three hours. Out of 100 training samples, a small set of 20 samples is used as dev set for hyper parameter tuning. Remaining 80 samples are used as train set in semi-supervised and supervised setting to compute supervised loss ($L_{sup}$). The entire train data of the original model $M$ is used for computing the unsupervised loss ($L_{unsup}$) in semi-supervised and unsupervised setting.

Templates: We evaluate only those templates via $SM$ that are chosen at least once while manually annotating the 100 training samples. For FB15K, it includes T1, T2, T4 and T5 and for YAGO, only T1, T2 and T5 are used.

4.1 Internal Evaluation

Similar to labelled training data, we annotate around 135 randomly sampled facts from the original test data with the template producing the best explanation. We note that in our internal evaluation, we use only those test queries where the true fact was scored at the top by our model since it is not quite clear how to evaluate the explanation of an incorrect tuple. We report micro-F1 for these templates over this unseen labelled test data as our evaluation metric. We repeat each experiment three times by changing random seeds and report the mean and standard deviation (See Appendix for details on hyperparameters, training time and computing infrastructure).
Table 1 reports the results from this evaluation. We observe that unsupervised setting has a huge variance, especially for the YAGO dataset, which is not surprising since it is training from a very weak supervision. Adding 80 labels in training makes a huge improvement, both in performance metric as well as variance of the results. We note that results on YAGO are particularly high, because there are very few relations for which \( M \) makes correct predictions at the top, making the role of OxKBC easy.

**Table 1: Micro-F1 score of OxKBC on two datasets (mean ± std deviation)**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Unsupervised</th>
<th>Supervised</th>
<th>Semi-supervised</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB15K</td>
<td>0.537 ± 0.012</td>
<td>0.421 ± 0.089</td>
<td>0.64 ± 0.011</td>
</tr>
<tr>
<td>YAGO</td>
<td>0.699 ± 0.384</td>
<td>0.965 ± 0.010</td>
<td>0.968 ± 0.004</td>
</tr>
</tbody>
</table>

4.2 User Evaluation

Since explanations are ultimately for end users, we compare the quality of our explanations over other proposed approaches, by a user study over Amazon Mechanical Turk\(^4\) on FB15K dataset. **Baseline:** For implementing rule mining baseline, we adapt the procedure as defined in [Yang et al., 2015] to mine closed-path logical rules upto 3 length and use embeddings trained by \( M \) to prune the exponential search space. The rules are ordered by confidence, defined as the ratio of number of correct predictions using the rule and the total number of predictions possible using body of the rule. Refer to section 5 in [Yang et al., 2015] for more details. In the first version of our experiments, we noticed that for many predictions, rules did not provide any explanation, which led to workers preferring OxKBC almost always over rule mining. To fix this, we reduced the thresholds, but that resulted in selection of longer rules much more frequently because of their rarity (making their confidence high). Experiments with this baseline resulted in workers preferring OxKBC over rules 84% of the time. But for a given prediction, we notice that a rule of length 1 is almost always more convincing than a rule of length 2 and so on. Hence we modify the rule selection procedure to prefer a lower length rule whenever present. Within same length, rule with highest confidence in selected. We report our results using this strong version of the baseline.

4.2.1 Quality of Explanations

We ask workers to compare explanations generated by OxKBC with explanations derived from the baseline. Both the systems use the same underlying base model \( M \) to generate their explanations. We randomly select 450 different facts from the test data and generate explanations from the two explanation engines. Out of these, 97 have the exact same explanation from both engines, and 21 have no explanation from both engines. For the remaining, we ask five different workers to select the better explanation. They are also given an option to reject or accept both explanations. In cases, where one engine offers no explanation, we ask workers whether the other explanation is better or worse than not having an explanation. For explaining entity similarity, we also generate a context for why two entities are similar and the user can hover to see the facts where the entities are interchangeable to see the rationale for why \( M \) considers two entities similar. Finally, we use majority voting to decide which explanation is better for a fact.

We present the results in Table 2 and split the analysis into two parts: one when model’s top prediction matches the gold annotation in the data, denoted as GT@1 (Gold Truth@1) in Table 2a, and the other when it doesn’t match the gold: N-GT@1 (Not–Gold Truth@1) in Table 2b. We see that not only are the explanations from OxKBC preferred over rule mining for GT@1, but even for the predictions

\[ \text{GT@1} \]

\[ \text{N-GT@1} \]

\[ \text{Selecting Workers: See Appendix B for details on how we select quality workers for our experiments.} \]
Table 2: OXKBC vs. rule mining under different scenarios: Overall: when either of the two systems give an explanation; Both: When both give an explanation; Only OXKBC: When only OXKBC gives an explanation (explained above); Similarly for ‘Only Rules’. Note that when only one of the systems produces an explanation (last two rows), we ask workers whether it is better or worse than having no explanation. If it is selected as being worse, we say that the other system, producing ‘no explanation’, does better in comparison.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>OXKBC Better</th>
<th>Rules Better</th>
<th>Tie</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall</td>
<td>145</td>
<td>49</td>
<td>18</td>
<td>212</td>
</tr>
<tr>
<td>Both</td>
<td>118</td>
<td>40</td>
<td>18</td>
<td>176</td>
</tr>
<tr>
<td>Only OXKBC</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Only Rules</td>
<td>25</td>
<td>7</td>
<td>0</td>
<td>32</td>
</tr>
</tbody>
</table>

(a) Comparison between explanations for GT@1

<table>
<thead>
<tr>
<th>Scenario</th>
<th>OXKBC Better</th>
<th>Rules Better</th>
<th>Tie</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall</td>
<td>94</td>
<td>16</td>
<td>10</td>
<td>120</td>
</tr>
<tr>
<td>Both</td>
<td>76</td>
<td>10</td>
<td>10</td>
<td>96</td>
</tr>
<tr>
<td>Only OXKBC</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Only Rules</td>
<td>18</td>
<td>3</td>
<td>0</td>
<td>21</td>
</tr>
</tbody>
</table>

(b) Comparison of explanations for N-GT@1

for samples in N-GT@1, OXKBC gives a more probable explanation than rule mining. Note that the workers are never told if the given fact is true or not. Table 3 lists some facts, where OXKBC outperformed the explanations generated by rule mining and vice-versa.

<table>
<thead>
<tr>
<th>Fact</th>
<th>Explanation from OXKBC</th>
<th>Explanation from Rule Mining</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Academy Award for Best Sound Mixing, has nomination for, WarGames)</td>
<td>[T4] (Academy Award for Best Sound Mixing, has nominee, Willie D. Burton) and (Willie D. Burton, was an award nominee for, WarGames)</td>
<td>(Academy Award for Best Sound Mixing, has nomination for, On Golden Pond) and (On Golden Pond, was nominated for, Academy Award for Best Cinematography) and (Academy Award for Best Cinematography, has nomination for, WarGames)</td>
</tr>
<tr>
<td>(The Last King of Scotland, has genre, Drama)</td>
<td>[T2] (The Lives of Others, has genre, Drama) and &quot;The Last King of Scotland&quot; is similar to &quot;The Lives of Others&quot;</td>
<td>(The Last King of Scotland, has actor, Forest Whitaker) and (Forest Whitaker, won an award for, Bird) and (Bird, has genre, Drama)</td>
</tr>
<tr>
<td>(47th Annual Grammy Awards, had an award category, Grammy Awards for Song of the Year)</td>
<td>[T2] (50th Annual Grammy Awards, had an award category, Grammy Awards for Song of the Year) and &quot;47th Annual Grammy Awards&quot; is similar to &quot;50th Annual Grammy Awards&quot;</td>
<td>(47th Annual Grammy Awards, is an instance of repeating event; Grammy Awards) and (Grammy Awards, category, Grammy Awards for Song of the Year)</td>
</tr>
<tr>
<td>(Actor, is the profession of, John Lithgow)</td>
<td>[T2] (Musician, is the profession of, John Lithgow) and &quot;Actor&quot; is similar to &quot;Musician&quot;</td>
<td>(Actor, is the profession of, Henry Winkler) and (Henry Winkler, has profession, Writer) and (Writer, is the profession of, John Lithgow)</td>
</tr>
</tbody>
</table>

Table 3: OXKBC was preferred to Rule Mining for first two facts, rule mining won in third fact, and none of the explanations were accepted in the last example.

4.2.2 Qualitative analysis of explanations for incorrect predictions

From Table 2b, we observe that workers prefer OXKBC’s explanations even when the prediction of the model $M$ is not marked as correct (N-GT@1). We randomly select 100 samples from N-GT@1 and manually analyze the explanations generated by OXKBC. For 77 out of 100 samples,
Table 4: Explanations generated by OxKBC for samples from N-GT@1 when the top prediction is indeed correct in real world.

OxKBC generated convincing explanations. On further investigation, we discovered that 52 out of 77 predictions were indeed correct in real-world but the model was penalized because the fact was not present in the KB or dev/test sets. This is an issue of the incompleteness of KB and demonstrates that good explanation can help in augmenting the KB with more facts (examples in Table 4). In the second case, we looked at remaining 25 odd examples where top model prediction was actually incorrect (in real-world). For 15 out of 25 samples, explanations looked reasonable even though the facts were not true. Table 5 lists few examples. While statistically a good prediction, in this particular case, the facts were untrue. We consider that these errors should not be used for model improvement, since the explanations are reasonably convincing even if the facts are untrue. The remaining 10 examples were the cases where the model made genuine errors (e.g., type related errors) and we did not get any proper explanation. These are the cases where we can potentially improve the model, e.g. by reranking until we get a plausible explanation. Doing this experimentation further as part of a controlled user study is a direction for future work.

Table 5: Explanations generated by OxKBC for samples from N-GT@1 when the predictions are actually incorrect.

5. Conclusions

We present OxKBC, a novel approach for providing post-hoc explanations for tensor factorization based KBC models. We introduce the idea of entity similarity edges in the knowledge graph and use paths in the augmented knowledge graph as explanations. We quantify over both relations and entities to define second-order templates for aggregating explanation paths and train a neural selection module to pick the best template for a given prediction. The neural model can be trained in a minimally supervised fashion due to a novel loss function. Based on a user study done on mechanical turk, we find that explanations generated by our system are significantly better than those generated from rule mining. In the future, we hope to extend the explanation engine for debugging the KBC model and improving its performance.
References


Appendix A.

Template Features:

<table>
<thead>
<tr>
<th>Global Features for a query $r(s, ?)$</th>
<th>Prediction specific features for $r(s, o)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>max score $\max_{u \in E} T_i(s, r, u)$</td>
<td>score $T_i(s, r, o)$</td>
</tr>
<tr>
<td>distr. mean $\text{mean}_{u \in E} T_i(s, r, u)$</td>
<td>similarity $\text{sim}<em>M(o, \arg \max</em>{u \in E} T_i(s, r, u))$</td>
</tr>
<tr>
<td>distr. std $\text{std}_{u \in E} T_i(s, r, u)$</td>
<td>rank $\text{rank}_o T_i(s, r, u)$</td>
</tr>
</tbody>
</table>

Table 6: Input Features for a prediction $r(s, o)$ from $i^{th}$ template $T_i$

Architecture: We use a two layer MLP with 90 and 40 neurons in two hidden layers as our model for the selection module. We imitate the Multi-Instance Learning setting, where the same model is used to output scores for each template $T_i$. Finally, we take a softmax over all these scores, to return a probability distribution over the space of templates, and select the one with maximum probability as our best explanation template. Figure 1 shows the overall architecture.

Hyperparameters: We use SGD optimizer, with a learning rate of 0.001, momentum of 0.9 and a batch size of 2048 for all our experiments. Table 7 states the hyper parameters used for each setting.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Setting</th>
<th>$\rho_{neg}$</th>
<th>$\rho_{pos}$</th>
<th>$\lambda_2$</th>
<th>$\lambda_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB15K</td>
<td>Un-supervised</td>
<td>1</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Semi supervised</td>
<td>1</td>
<td>0.125</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>YAGO</td>
<td>Un-supervised</td>
<td>0.5</td>
<td>0.2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Semi supervised</td>
<td>0.25</td>
<td>0.25</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 7: Hyper-parameters used for various settings
Figure 1: Architecture of our Selection Module. The underlying two-layer MLP is same when predicting score for every template.

Computing Infrastructure and Time Analysis: For generating feature vectors corresponding to each template, we used a machine with 8 Intel(R) Xeon(R) CPU @ 2.50GHz cores and 128 GB of RAM. No GPU memory was required. It required about 5 hours to preprocess and generate the feature vectors for the train, dev and test sets. For training the two layer MLP in SM, we used a machine with 4 cores, 64 GB of RAM and 1 GPU with 12 GB memory, though the GPU memory required is less than 1 GB (for a batch size of 2048). The module gets trained in about 20 minutes.

Pedagogical Approach: The approach as mentioned in [Gusmão et al., 2018] does not scale to the size of FB15K dataset. They perform their experiments on a toy FB13 dataset where they use only 13 relations of the entire FB15K dataset. We tried their methodology on FB15K but memory and storage requirements were intractable. According to our calculations, it would take 25 TB of storage and 20 days of compute, using the available code on an available system with 256 GB of RAM and 40 Intel(R) Xeon(R) CPU @ 2.80GHz cores.

Appendix B. Mechanical Turk Experiment Details

Selecting Workers: To make sure that the workers really understand the task, we create an interactive tutorial of 6 questions in which users can select different options and get feedback about each of them. Figure 2 shows a screenshot from the tutorial. To make our tutorial unbiased, we select questions which expose the good and bad sides of both the systems, and at no point are the workers told which explanation comes from which system. On completing the tutorial, they perform the task of comparing the two explanations on a small set of 10 gold questions. For each question, they are also asked to give a subjective reason for their choice. We analyze their answers quantitatively and their reasons qualitatively and manually cherry-pick the diligent and reasonable workers. All subsequent experiments are done with these quality workers.
**Human Intelligence Task:** Figure 2 shows a screenshot of the task given to workers. We ask them to chose which out of the two explanations are better. They can select or reject both the explanations as well. In case only one system gives an explanation, workers choose if it is better than no explanation or not. In order to explain the task better, we create an interactive tutorial where workers can see why some explanation is inferior compared to its counterpart. For explaining entity similarity or high frequency of entity, we also generate a context of why two entities are similar or why is some entity frequent and the user can hover over bordered boxes to see the additional facts.
(a) Instructions provided to workers

**OBJECTIVE:** Which of the two explanations is better for explaining the given prediction?

**DETAILED INSTRUCTIONS:**
Our Artificial Intelligence (AI) based Question Answering (QA) system predicts an answer to questions of type:

*Entity e1 is related via relation r to an entity _____?*

Or simply (e1, r, ?)

For example, **Kate Moss has profession _____?**

**Answer:** Model

This would be followed by two plausible explanations, which may not always imply the prediction, but they may increase our confidence about it. For example, two explanations for the above prediction can be:

1. Kate Moss has profession Actor and Actor is profession of Brigitte Nielsen and Brigitte Nielsen has profession Model
2. Keira Knightley has profession Model and "Kate Moss" is similar to "Keira Knightley"

Your job is to select which of the two explanations is better.

In case only one explanation is generated, you need to choose whether it is more helpful than having no explanation or not.

Please note the following:

1. Hyperlinks on the objects would take you to the corresponding wiki page if it exists, otherwise, you will land on Wikipedia homepage.
2. Whenever an explanation has some bolded text like, `Jiminy`, you can hover over it to see additional supporting facts.
3. Sometimes you might see an empty list on hovering, in which case system has no supporting facts to specify.
4. There might be some relations in our Knowledge Base that have exactly the same meaning. For example has genre and netflix title has genre

---

**Question:** Desperate Housewives was nominated for _____?

**Answer:** Primetime Emmy Award for Outstanding Comedy Series

**Explanation A:** Desperate Housewives is an award nominating work of Marc Cherry and Marc Cherry was nominated for Primetime Emmy Award for Outstanding Comedy Series

**Explanation B:** Desperate Housewives has regular appearance of Lyndsy Fonseca and Lyndsy Fonseca has regular appearance in How I Met Your Mother and How I Met Your Mother was nominated for Primetime Emmy Award for Outstanding Comedy Series

- A is better than B
- B is better than A
- Both A and B are equally good
- Both A and B are bad

Wrong.
Just because another show was nominated for an award and both have regular appearance of a common actor (Lyndsy Fonseca), it doesn’t indicate strongly that ‘Desperate Housewives’ should also be nominated for the same award. Moreover, Explanation A not only gives an explanation, it also tells the writer / producer of Desperate Housewives (Marc Cherry) who was nominated.

---

**Question:** Turun Palloseura football team has a play position _____?

**Answer:** Midfielder

**Explanation A:** Turun Palloseura football team has a play position Defender and Defender is a position in the football team Lierse S.K. and Lierse S.K. has a position of Midfielder

**Explanation B:** In our Knowledge Base, many football teams (598 of 745) have players at Midfielder

(Australia national soccer team, Olympique de Marseille and 596 more...) football team has a play position Midfielder

- A is better than B
- B is better than A
- Both A and B are equally good
- Both A and B are bad

---

(b) The tutorial has 6 of such questions which we use to tell workers why some explanations are better than others.

---

(c) Example query with frequent entity context

**Figure 2:** Human Intelligence Task on Amazon Mechanical Turk