DAIKIRI
Erklärbare Diagnostische KI für industrielle Daten

Project Number: 01IS19085B **Start Date of Project:** 01/01/2020 **Duration:** 24 months

Deliverable 2.2
Introduction of Constraints & Scalable Implementation

Dissemination Level	Public
Due Date of Deliverable	31/12/2021
Actual Submission Date	31/12/2021
Work Package	WP 2: Embeddings
Task	T2.3, T2.4
Type	Report
Approval Status	Final
Version	1.0
Number of Pages	16

GEFÖRDERT VOM

Bundesministerium
für Bildung
und Forschung

This project has received funding from the Federal Ministry of Education and Research (BMBF) within the project DAIKIRI under the grant no 01IS19085B.

History

Version	Date	Reason	Revised by
1.0	31/12/2021	Final version created	Caglar Demir

Author List

Organization	Name	Contact Information
UPB	Caglar Demir	caglar.demir@upb.de

Executive Summary

Knowledge Graph Embedding (KGE) models learn continuous vector representations for Knowledge Graphs (KGs). These representations have been successfully applied in a large number of applications. In the previous deliverable (D2.1), we presented our six KGE models that were developed within the DAIKIRI project. As a result of our investigations in KGEs, we published three research papers in top-tier conferences [Demir et al., 2021a, Demir and Ngomo, 2021, Demir et al., 2021b]. All our models can scale well on large KGs as they retain a linear space complexity in the size of KGs. In this work, we investigate (1) novel techniques allowing to introduce constraints in KGEs and (2) developing a scalable implementation of models. Findings of our investigation in constraining KGEs indicate that leveraging the domain and range information of relations in predicting of missing links improve generalization performances of all models on all benchmark datasets. This is an important finding as our technique can be readily applied in any pretrained model, i.e., **no extra computation is required**. Our results also indicate that state-of-the-art KG models do not fully capture information pertaining to domains and ranges of relations encoded in benchmark datasets.

Our investigation on publicly available implementations of KGE models suggest that many framework do not facilitate parallelism, let alone distributed computing. To best of our knowledge, publicly available KGE frameworks do not utilize multi-CPUs in preprocessing of input KGs. This indicates that loading a KG into a memory is often carried out by using a single CPU. This design decision may stem from the fact that benchmark KG datasets for link prediction are relatively small compared to KGs used in business related applications. Hence, many KGE frameworks may not be suitable for applications outside of research domains. Motivated by this shortcoming, we developed the DAIKIRI-Embedding framework[1]. Our framework contains scalable implementations of many KGE models. DAIKIRI-Embedding **leverages multi-CPUs, GPUs, and even TPUs** during the training and testing phases via Pytorch-Lightning [Falcon and Cho, 2020]. Moreover, DAIKIRI-Embedding **utilizes multi-CPUs to load/parse large KGs** via the DASK framework [Rocklin, 2015]. Hence, the DAIKIRI-Embedding framework can be applied on large KGs. To evaluate the scalability of our framework, we performed numerous experiments. We have successfully used the DAIKIRI-Embedding framework to learn embeddings of DBpedia KG that contain more than hundreds of millions of triples. We made scripts and pre-trained embeddings publicly available on the Hobbit platform.

[1] `https://github.com/dice-group/DAIKIRI-Embedding`

Contents

1 Introduction

Over the last decade, KGs have become indispensable in a large number of data-driven applications [Hogan et al., 2021]. For instance, many companies followed the example of the Google Knowledge Graph, including LinkedIn, Microsoft, eBay, Amazon, AirBnB, and Uber [Singhal, 2012, He et al., 2016, Pittman, 2017, Krishnan, 2018, Chang, 2018, Hamad et al., 2018]. The wealth of knowledge available in KGs also serves as background data for an increasing number of intelligent applications [Wang et al., 2017], including web search, cancer research, and even entertainment [Eder, 2012, Saleem et al., 2014, Malyshev et al., 2018]. However, most KGs on the Web are far from being complete [Nickel et al., 2015]. For instance, the birth places of 71% of the people in Freebase and 66% of the people in DBpedia are not found in the respective KGs. In addition, more than 58% of the scientists in DBpedia are not linked to the predicate that describes what they are known for [Krompaß et al., 2015]. KGE models have been particularly successful at tackling many problems such as relation prediction, link prediction, entity resolution, question answering, product recommendation [Nickel et al., 2015, Ji et al., 2020]. In this work, we focus on KGE approaches that are trained to tackle the link prediction task. Link prediction on KGs refers to identifying such missing information [Dettmers et al., 2018]. KGE models have been particularly successful at tackling the link prediction task [Nickel et al., 2015].

Recent studies highlight the ever-increasing predictive ability of KGE models. Although, KGE models can accurately predict missing links in the input KG by means of learned vector representations of entities and relations, they often lack of explainability in their predictions. Motivated by this inability, we investigated constraints in KGEs. Most KGEs models do not facilitate the domain expert knowledge in the learning process. This often stems from the fact that incorporating domain knowledge in the learning process is not trivial. Moreover, incorporating the domain expert knowledge into the learning process may induce bias in the predictions. In this work, we investigate introducing constraints in the learning phase, as well as in the testing phase. Our results indicate that utilizing the domain and range information of relations in the testing phase improve generalization performances of all models on all benchmark datasets. In turn, constraining embeddings during the training phase based on a similarity function as previously done by Demir and Ngomo [2019] led to inferiors performances in the link prediction task.

The structure of this work is as follows: Section 2 briefly introduces preliminaries. In Section 3, we elucidate our constraint techniques for KGEs. Section 4 explains our KGE framework developed within the DAIKIRI project. Next, Section 5 provides details of our experimental setup used in experiments. In Section 6, we report results of our experiments. Finally, we conclude this work with Section 7.

2 Background

In this section, we briefly introduce necessary background knowledge for this work. In Section 2.1, we provide the definition of knowledge graph and link prediction used throught our work. Section 2.2 introduce our six knowledge graph embedding model. We refer [Demir et al., 2021a, Demir and Ngomo, 2021, Demir et al., 2021b] for more details pertaining to our models.

2.1 Knowledge Graph & Link Prediction

Let $\mathcal{E}$ and $\mathcal{R}$ represent the sets of entities and relations. Then, a KG can be formalised as a set of triples $\mathcal{G} = \{(\mathtt{h}, \mathtt{r}, \mathtt{t})\} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ where each triple contains two entities $\mathtt{h}, \mathtt{t} \in \mathcal{E}$ and a relation

$\mathtt{r} \in \mathcal{R}$. We defined the domain of a relation as follows

$$domain(r) = \{h \mid \forall\ (\mathtt{h}, \mathtt{r}, \mathtt{t}) \in \mathcal{G}\}. \tag{1}$$

Similarly, the range of a relation is defined as

$$range(r) = \{t \mid \forall\ (\mathtt{h}, \mathtt{r}, \mathtt{t}) \in \mathcal{G}\}. \tag{2}$$

The domain rand range of a relation is utilized at constraining KGE models. The link prediction problem is formalised by learning a scoring function $\psi : \mathcal{E} \times \mathcal{R} \times \mathcal{E} \rightarrow \mathbb{R}$ ideally characterized by $\psi(\mathtt{h}, \mathtt{r}, \mathtt{t}) > \psi(\mathtt{x}, \mathtt{y}, \mathtt{z})$ if $(\mathtt{h}, \mathtt{r}, \mathtt{t})$ is true and $(\mathtt{x}, \mathtt{y}, \mathtt{z})$ is not [Dettmers et al., 2018, Demir et al., 2021b].

2.2 Knowledge Graph Embeddings

2.2.1 Shallom

Link prediction problem refers to predicting missing triples (see Section 2.1). Most approaches achieve this goal by predicting entities, given an entity and a relation. Shallom tackles the link prediction problem by predicting missing relations. The learning problem is formalized as a multi-label classification problem. Shallom is analogous to C-BOW as both approaches predict a central token ($\mathtt{p}$) given surrounding tokens (($\mathtt{s}$, $\mathtt{o}$)). We defined Shallom as

$$\psi(s, o) = \sigma\Big(\mathbf{W} \cdot \mathrm{ReLU}\big(\mathbf{H} \cdot \Psi(s, o) + \mathbf{b}_1\big) + \mathbf{b}_2\Big), \tag{3}$$

where $\Psi(s, o) \in \mathbb{R}^{2d}$, $\mathbf{H} \in \mathbb{R}^{k \times 2d}$, $\mathbf{W} \in \mathbb{R}^{|\mathcal{R}| \times k}$, $\mathbf{b}_1 \in \mathbb{R}^k$, and $\mathbf{b}_2 \in \mathbb{R}^{|\mathcal{R}|}$. $\sigma(\cdot)$, $\mathrm{ReLU}(\cdot)$ and $\Psi(\cdot, \cdot)$ denote the sigmoid, the rectified linear unit and the vector concatenation functions, respectively. Given ($\mathtt{s}$, $\mathtt{o}$), $\Psi(s, o)$ returns concatenated embeddings of $(\mathtt{s}, \mathtt{o})$. Thereafter, we perform two affine transformations with the ReLU and the sigmoid function to obtain predicted probabilities for relation ($\hat{\mathbf{y}} \in \mathbb{R}^{|\mathcal{R}|}$). Finally, the incurred loss is computed by the binary cross-entropy function:

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{i}^{|\mathcal{R}|} \Big((\mathbf{y}_i \cdot \log(\hat{\mathbf{y}}_i)) + (1 - \mathbf{y}_i) \cdot \log(1 - \hat{\mathbf{y}}_i) \Big) \tag{4}$$

where $\hat{\mathbf{y}}$ is the vector of predicted probabilities and $\mathbf{y}$ is a binary vector of indicating multi labels. Equation (3) shows that the space complexity of Shallom is linear in the number of entities of the input KG.

The architecture of Shallom is visualized in Figure 1. To obtain a composite representation of ($\mathtt{s}$, $\mathtt{o}$), we concatenate embeddings of entities as opposed to averaging them, since averaging embeddings loses the order of the input (as in the standard bag-of-words representation [Le and Mikolov, 2014]). Retaining order of embeddings avoids possible loss of information. As concatenation does not consider any interaction between the latent features, the first affine transformation is applied with the ReLU activation function. Thereafter, the second affine transformation is applied with the sigmoid function to generate probabilities for relations.

2.2.2 ConEx

Inspired by the previous works ComplEx [Trouillon et al., 2016] and ConvE [Dettmers et al., 2018], we dub our approach ConEx (<u>con</u>volutional compl<u>ex</u> knowledge graph embeddings).

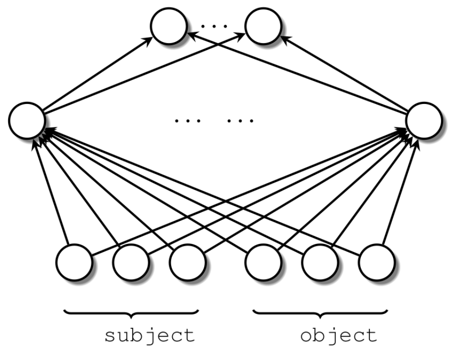

Figure 1: Visualization of SHALLOM.

Sun et al. [2019] suggested that ComplEx is not able to model triples with transitive relations since ComplEx does not perform well on datasets containing many transitive relations (see Table 5 and Section 4.6 in [Sun et al., 2019]). Motivated by this consideration, we propose CONEX, which applies the Hadamard product to compose a 2D convolution followed by an affine transformation with a Hermitian inner product in $\mathbb{C}$. By virtue of the proposed architecture (see Equation (5)), CONEX is endowed with the capability of

1. leveraging a 2D convolution and
2. degenerating to ComplEx if such degeneration is necessary to further minimize the incurred training loss.

CONEX benefits from the *parameter sharing* and *equivariant representation* properties of convolutions [Goodfellow et al., 2016]. The parameter sharing property of the convolution operation allows CONEX to achieve parameter efficiency, while the equivariant representation allows CONEX to effectively integrate interactions captured in the stacked complex-valued embeddings of entities and relations into computation of scores. This implies that small interactions in the embeddings have small impacts on the predicted scores[2]. The rationale behind this architecture is to increase the expressiveness of our model without increasing the number of its parameters. As previously stated in [Trouillon et al., 2016], this nontrivial endeavour is the keystone of embedding models. Ergo, we aim to overcome the shortcomings of ComplEx in modelling triples containing transitive relations through combining it with a 2D convolutions followed by an affine transformation on $\mathbb{C}$.

Given a triple $(\mathtt{h}, \mathtt{r}, \mathtt{t})$, $\textsc{ConEx}: \mathbb{C}^{3d} \mapsto \mathbb{R}$ computes its score as

$$\textsc{ConEx}(h, r, t) = \mathrm{Re}(\langle \mathrm{conv}(\mathbf{e}_h, \mathbf{e}_r), \mathbf{e}_h, \mathbf{e}_r, \overline{\mathbf{e}_t} \rangle), \tag{5}$$

where $\mathrm{conv}(\cdot, \cdot): \mathbb{C}^{2d} \mapsto \mathbb{C}^d$ is defined as

$$\mathrm{conv}(\mathbf{e}_h, \mathbf{e}_r) = f\big(\mathrm{vec}(f([\mathbf{e}_h, \mathbf{e}_r] * \omega)) \cdot \mathbf{W} + \mathbf{b}\big), \tag{6}$$

where $f(\cdot)$ denotes the rectified linear unit function (ReLU), $\mathrm{vec}(\cdot)$ stands for a flattening operation, $*$ is the convolution operation, ω stands for kernels/filters in the convolution, and $(\mathbf{W}, \mathbf{b})$ characterize an affine transformation.

By virtue of its novel structure, CONEX is enriched with the capability of controlling the impact of a 2D convolution and Hermitian inner product in the predicted scores. Ergo, CONEX is less prone

[2] We refer to Section 2 and Goodfellow et al. [2016] for further details of properties of convolutions.

to the vanishing gradient problem as the gradients of losses (see Equation (9)) w.r.t. $(\mathbf{e}_h, \mathbf{e}_r, \mathbf{e}_t)$ are allowed to backpropagate through $\text{conv}(\mathbf{e}_h, \mathbf{e}_r)$ or $\text{Re}(\langle \mathbf{e}_h, \mathbf{e}_r, \overline{\mathbf{e}_t} \rangle)$. Equation (5) can be equivalently expressed by expanding its real and imaginary parts:

$$
\begin{aligned}
\textsc{ConEx}(h, r, t) &= \text{Re}\Big(\sum_{k=1}^{d} (\gamma)_k (\mathbf{e}_h)_k (\mathbf{e}_r)_k (\overline{\mathbf{e}_t})_k \Big) && (7) \\
&= \langle \text{Re}(\gamma), \text{Re}(\mathbf{e}_h), \text{Re}(\mathbf{e}_r), \text{Re}(\mathbf{e}_t) \rangle \\
&\quad + \langle \text{Re}(\gamma), \text{Re}(\mathbf{e}_h), \text{Im}(\mathbf{e}_r), \text{Im}(\mathbf{e}_t) \rangle \\
&\quad + \langle \text{Im}(\gamma), \text{Im}(\mathbf{e}_h), \text{Re}(\mathbf{e}_r), \text{Im}(\mathbf{e}_t) \rangle \\
&\quad - \langle \text{Im}(\gamma), \text{Im}(\mathbf{e}_h), \text{Im}(\mathbf{e}_r), \text{Re}(\mathbf{e}_t) \rangle && (8)
\end{aligned}
$$

where $\overline{\mathbf{e}_t}$ is the conjugate of $\mathbf{e}_t$ and γ denotes the output of $\text{conv}(\mathbf{e}_h, \mathbf{e}_r)$ for the brevity. Such multiplicative inclusion of $\text{conv}(\cdot, \cdot)$ equips ConEx with two more degrees of freedom due the $\text{Re}(\gamma)$ and $\text{Im}(\gamma)$ parts. We train our approach by following a standard setting [Dettmers et al., 2018, Balažević et al., 2019b]. Similarly, we applied the standard data augmentation technique, the KvsAll training procedure[3]. After the data augmentation technique, for a given pair (`h`, `r`), we compute scores for all $x \in \mathcal{E}$ with $\psi(\mathtt{h}, \mathtt{r}, \mathtt{x})$. We then apply the logistic sigmoid function $\sigma(\psi(h, r, t))$ to obtain predicted probabilities of entities. ConEx is trained to minimize the binary cross entropy loss function L that determines the incurred loss on a given pair (`h`, `r`) as defined in the following:

$$
L = -\frac{1}{|\mathcal{E}|} \sum_{i=1}^{|\mathcal{E}|} (\mathbf{y}^{(i)} \log(\hat{\mathbf{y}}^{(i)}) + (1 - \mathbf{y}^{(i)}) \log(1 - \hat{\mathbf{y}}^{(i)})), \tag{9}
$$

where $\hat{\mathbf{y}} \in \mathbb{R}^{|\mathcal{E}|}$ is the vector of predicted probabilities and $\mathbf{y} \in [0, 1]^{|\mathcal{E}|}$ is the binary label vector.

In Figure Figure 2, we visualized a 2D PCA projection of relation embeddings that are obtained after ConEx is trained on the FB15K-237 benchmark dataset. Figure 2 shows that inverse relations cluster in distant regions. Note that we applied the standard data augmentation technique (see section 4.1 in [Balažević et al., 2019b]). Such relations are renamed by adding suffix of *inverse* as done in [Balažević et al., 2019b]. Figure Figure 2 also show that embeddings of `person` related relations are learned to be close to each other. Hence, this information can be utilized to predict the birth place of the people in Freebase as the birth place of 71% of the people in Freebase is missing [Krompaß et al., 2015]. Based on the visualisation, one may infer that the cluster located on the left upper part of the figure where `person/nationality`, `person/gender` and `person/gender`, may consist on 1-1 type relations as many entities as head entities on FB15K-237 do not occur with such relations multiple times. However, this interpretation requires further investigation.

2.2.3 Convolutional Hypercomplex Embeddings

Motivated by findings of Demir and Ngomo [2021] in the composition of a 2D convolution with a Hermitian inner product on complex-valued embeddings. We extend ConEx into quaternions and octonions. To this end, we first propose QMult and OMult that are multiplicative models. Next, we build ConvQ and ConvO upon QMult and OMult, respectively. Inspired by the early works DistMult [Yang et al., 2015] and ConvE [Dettmers et al., 2018], we dub our approaches QMult, OMult, ConvQ, and ConvO where “Q” represents the quaternion variant and “O” the octonion variant.

[3] Note that the KvsAll strategy is called 1-N scoring in [Dettmers et al., 2018]. Here, we follow the terminology of [Ruffinelli et al., 2019].

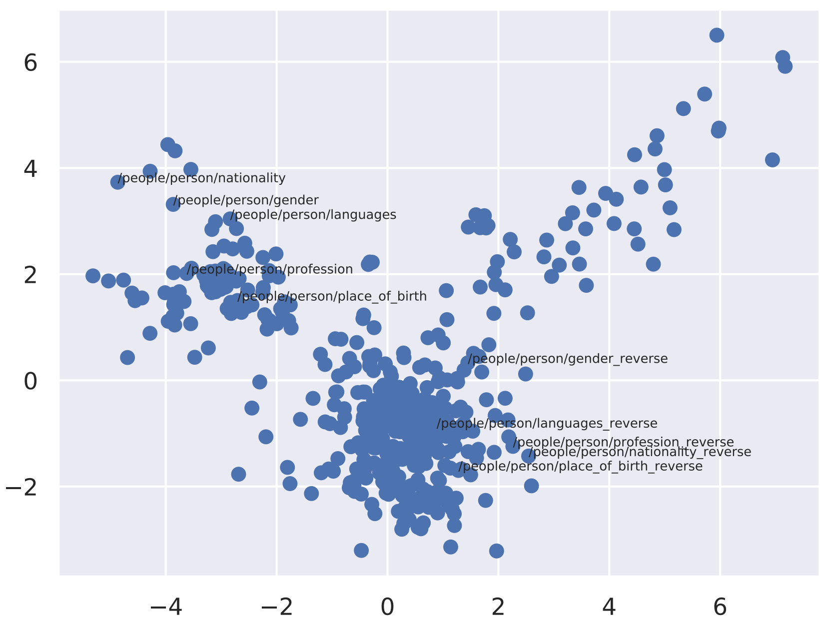

Figure 2: A 2D PCA projection of relation embeddings.

Given a triple $(\mathtt{h}, \mathtt{r}, \mathtt{t})$, QMULT : $\mathbb{H}^{3d} \mapsto \mathbb{R}$ computes a triple score through the quaternion multiplication of head entity embeddings $\mathbf{e}_h$ and relation embeddings $\mathbf{e}_r$ followed by the inner product with tail entity embeddings $\mathbf{e}_t$ as

$$\textsc{QMult}(h, r, t) = \mathbf{e}_h \otimes \mathbf{e}_r \cdot \mathbf{e}_t, \tag{10}$$

where $\mathbf{e}_h, \mathbf{e}_r, \mathbf{e}_t \in \mathbb{H}^d$. Similarly, OMULT : $\mathbb{O}^{3d} \mapsto \mathbb{R}$ performs the octonion multiplication followed by the inner product as

$$\textsc{OMult}(h, r, t) = \mathbf{e}_h \bigstar \mathbf{e}_r \cdot \mathbf{e}_t, \tag{11}$$

where $\mathbf{e}_h, \mathbf{e}_r, \mathbf{e}_t \in \mathbb{O}^d$. Computing scores of triples in this setting can be illustrated in two consecutive steps: (1) rotating $\mathbf{e}_h$ through $\mathbf{e}_r$ by applying quaternion/octonion multiplication and (2) measuring the angle between $(\mathbf{e}_h \otimes \mathbf{e}_r)$ and $\mathbf{e}_t$ as expressed by the inner product. During training, this angle is maximized for triples $(\mathtt{h}, \mathtt{r}, \mathtt{t}) \in \mathcal{G}$.

Motivated by the findings of [Demir and Ngomo, 2021], we combine convolution operations with QMULT and OMULT as defined in Equation (12) and Equation (13):

$$\textsc{ConvQ}(h, r, t) = \mathrm{conv}(\mathbf{e}_h, \mathbf{e}_r) \circ (\mathbf{e}_h \otimes \mathbf{e}_r) \cdot \mathbf{e}_t, \tag{12}$$

$$\textsc{ConvO}(h, r, t) = \mathrm{conv}(\mathbf{e}_h, \mathbf{e}_r) \circ (\mathbf{e}_h \bigstar \mathbf{e}_r) \cdot \mathbf{e}_t, \tag{13}$$

where $\mathrm{conv}(\cdot, \cdot) : \mathbb{H}^{2d} \mapsto \mathbb{R}^{4d}$ (respectively $: \mathbb{O}^{2d} \mapsto \mathbb{R}^{8d}$) is defined as

$$\mathrm{conv}(\mathbf{e}_h, \mathbf{e}_r) = f\Big(\mathrm{vec}(f([\mathbf{e}_h, \mathbf{e}_r] * \omega)) \cdot \mathbf{W}\Big), \tag{14}$$

where $f(\cdot), \mathrm{vec}(\cdot), *, \omega$, and $\mathbf{W}$ denote the rectified linear unit function, a flattening operation, convolution operation, kernel in the convolution and a projection matrix, respectively. During training, we follow a 1-N scoring regime (with $N = |\mathcal{E}|$) for efficient training [Dettmers et al., 2018]. In the 1-N scoring regime, a KGE model takes ($\mathtt{s}$, $\mathtt{p}$) as an input and generates $|\mathcal{E}|$ scores for each RDF triple

$(\mathtt{s}, \mathtt{p}, \mathtt{x})$ with $x \in \mathcal{E}$. Training with 1-N scoring regime has two advantages: (1) the regime has an effect akin to batch normalization, and (2) faster convergence [Dettmers et al., 2018]. We also employ the Glorot initialization technique for parameters of ConvQ, as using the logistic sigmoid activation often drives the top hidden layer into saturation provided that parameters are randomly initialized [Glorot and Bengio, 2010].

3 Constraining Knowledge Graph Embeddings

Most state-of-the-art knowledge graph embedding approaches do not explicitly involve any form of constraints. To improve generalization performances, approaches often rely on constraining magnitude of embeddings by mean of applying L-2 or N3 regularization, while few approaches normalized embeddings via the batch normalization or the group normalization techniques. However, most approaches do not incorporate the domain expert knowledge as constraints in KGEs. In this section, we introduce our constraining techniques.

In Section 3.1, we elucidate the particular incorporation of the domain expert knowledge by means of the domain and range information of relations in KGEs. Our proposed technique

- incorporates the domain knowledge with no extra computation and
- can be readily applied for any knowledge graph embedding model.

In Section 3.2, we introduce our second constraint technique that is motivated by the Elastic Net Regularization and PYKE KGE model [Demir and Ngomo, 2019]. More specifically, we propose a similarly function that harmonically combines the domain and range similarity between relations. Based on this similarity function, we designed a regularisation technique that the output of the radial basis kernel function of two embeddings of relations to be similar to the aforementioned similarity function.

3.1 Constraining Prediction To Recover Semantic Errors

Motivation. Our attempt of constraining knowledge graph embeddings is motivated by the standard training technique (the KvsAll scoring) for knowledge graph embedding models. Figure 3 illustrates the KvsAll scoring technique. Most knowledge graph embedding models including our models are trained with the KvsAll technique. For each triple $(\mathtt{h}, \mathtt{r}, \mathtt{t})$, an approach (ConvE in Figure 3) takes an embedding of head entity denoted by purple row and an embedding of relation denoted by green row. After performing sequence of computations, unnormalized log probabilities (logits) are generated. Thereafter, logits are normalized via the sigmoid loss function to obtain predictions $\hat{\mathbf{y}} \in (0,1)^{|\mathcal{E}|}$. Thereupon, the incurred loss on a given pair $(\mathtt{h}, \mathtt{r})$ is computed via the binary cross entropy loss function (see Equations (4) and (9)). Within this setting, learning embeddings of knowledge graphs can be seen as tackling multi-label classification problem, where an input $\mathbf{x} \in \mathbb{R}^{2d}$ and an label $\mathbf{y} \in 0,1^{|\mathcal{E}|}$ that is a binary vector indicating multi-labels. For instance, given (`BarackObama`, `type`), indexes of `Person` and `Politician` in $\mathbf{y}$ is set to be 1, wheres indexes of `SportCar`, `Country`, or `FootballPlayer` are set to be 0.

State-of-the-art models are often trained with the KvsAll scoring technique [Ruffinelli et al., 2019]. However, applying the KvsAll scoring technique becomes computationally infeasible as the size of the unique entities increases ($|\mathcal{E}|$). To update embeddings of a single input w.r.t. the incurred loss, we need to store (1) embeddings of all entities and relations, (2) derivative of loss w.r.t. parameters

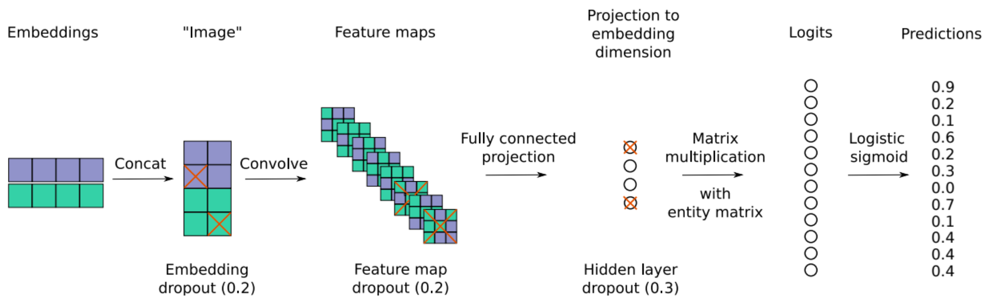

Figure 3: Illustration of the KvsAll scoring technique via ConvE approach obtained from [Dettmers et al., 2018]

in intermediary computations (the dropout, feature map, fully connected projection, second dropout in Figure 3), and importantly (3) derivatives of loss w.r.t. embeddings of all entities obtained at generation predictions. As the size of the embedding dimension and the size of the unique entities increases, (3) becomes computationally infeasible, hence requires performant hardware.

The KvsAll scoring technique implicitly leads knowledge graph embedding models to consider losses incurred due to all non-existing triples equally important. For instance, given (`BarackObama`, `type`), losses incurred on (`BarackObama`, `type`, `Country`), (`BarackObama`, `type`, `SportCar`), and (`BarackObama`, `type`, `FootballPlayer`) are equally important as far as the optimization process is concerned. We propose to make a distinction between losses incurred on non-existing triples $(\mathtt{x}, \mathtt{y}, \mathtt{z}) \notin \mathcal{G}$. To this end, we are interested in utilizing the domain and range information of relations that are readily available in the input KG. Within this setting, a loss occurred on (`BarackObama`, `type`, `FootballPlayer`) is considered as less important than a loss occurred on (`BarackObama`, `type`, `Country`) and and (`BarackObama`, `type`, `SportCar`).Such constraint can be applied during the training phase by making labels less sparse, i.e., setting low scores for those entities that are subsumed by the range of relations, or during the testing phase. Due to the space constraint, here, we focus on introducing the domain and range constraints in prediction.

Constraining predictions during testing. We firstly obtain the domain and range information of relations on $\mathcal{G}^{Train}$ via Equation (1) and Equation (2). Next, given a test triple (`h`, `r`, `t`), we obtain $|\mathcal{E}|$ number of predicted scores based on (`h`, `r`) by using a pre-trained model. Thereupon, we filter scores of $\{\mathtt{t} \in \mathcal{E} | \mathtt{t} \notin range(\mathtt{r})\}$. Hence, we constrain predictions by ignoring entities that are not containing in the range of relations. A domain expert intuitively may know that `BarackObama` is not likely to have a type of `Country` or `SportCar`, whereas `BarackObama` is likely to have a type of `FootballPlayer` as the following is expected to hold for consistent knowledge graphs: $\neg \exists x \in \mathcal{E} : (\mathtt{x}, \mathtt{type}, \mathtt{Person}) \wedge (\mathtt{x}, \mathtt{type}, \mathtt{SportCar})$.

The utility of this constraint can be easily validated by comparing the link prediction results of models with and without this constraint. If model capture the range information of relation, applying this constraint is expected to not increase the link prediction results.

3.2 Elastic Constraint Regularization

During our investigation of constraining KGE models, we explored several different techniques. However, they often lead to inferior performance in the link prediction task. Here, we introduce `ElasticReg` constraint that constraints the distances between relations during the training phase based on a semantic similarity measure between relations.

We define this similarity function $\varphi : \mathcal{R} \times \mathcal{R} \mapsto [0, 1]$ as follows

$$\varphi(r_i, r_j) = \alpha \frac{|domain(r_i) \cap domain(r_j)|}{\beta(|domain(r_i)| + |domain(r_j)|)|} + (1-\alpha) \frac{|range(r_i) \cap range(r_j)|}{\beta(|range(r_i)| + |range(r_j)|)}, \tag{15}$$

where $\alpha \in [0, 1]$, $\beta = .5$. In Equation (15), the importance of domain and range information can be controlled via α as similarly done in L-1 and L-2 regularization trade off in the Elastic Net. The similarity matrix A between relations is constructed via φ. A is a symmetric and semi positive definite matrix, i.e,. the diagonal values are 1.0 and the eigenvalues are strictly positive values. Our goal is to constraint embeddings of relations during training through A. Hence, embeddings of relations having similar domains and/or ranges are constrained to be similar. To quantify the degree of similarity in the vector space, we rely on the radial basis kernel function:

$$K(\mathbf{e}_{r_j}, \mathbf{e}_{r_i}) = exp\Big(- \frac{||\mathbf{e}_{r_i}, \mathbf{e}_{r_j}||^2}{2\sigma^2}\Big). \tag{16}$$

During the training phase, embeddings of relations are updated to minimize the binary cross entropy function and the constraint error $\big(\varphi(r_i, r_j) - K(\mathbf{e}_{r_j}, \mathbf{e}_{r_i})\big)^2$.

4 DAIKIRI-Embedding and Scalability

In this section, we elucidate the three components of the DAIKIRI-Embedding framework.

Preprocessing. During preprocessing a knowledge graph in the form of n-triples, we rely on the DASK framework [Rocklin, 2015]. The DASK framework allows to utilize multi-CPUs at preprocessing/loading large data in the form of CSV, Parquet, and HDF among many others. We load the n-triple formatted knowledge graph by using the white-space as a column separator. This permits using all CPUs during loading large datasets. Given that head entity/subject and tail entity/predicate must not contain any white space according to RDF[4], we alleviate any error due to inconsistent formatting.

Training. To implement a scalable training module in the DAIKIRI-Embedding framework, we rely on PyTorch Lightning [5]. PyTorch Lightning facilitates using parallelism, hence, enhance scalability. PyTorch Lightning is one of the mostly used deep learning framework along with PyTorch and Tensorflow [Falcon and Cho, 2020].

Evaluation. In many industrial application, creating a training, validation, and test split of the data is not trivial. With this consideration, we implement two different evaluation scenario in the DAIKIRI-Embedding framework:

- K-fold cross entropy with different initialization on the train split and
- the standard testing on the test split.

5 Experimental Setup

5.1 Datasets

We used three most commonly used link benchmark datasets to evaluate the impact of our constraint techniques. To evaluate the scalability of the DAIKIRI-Embedding framework, we used the most recent

[4] `https://www.w3.org/TR/n-triples/`
[5] https://www.pytorchlightning.ai/

DBpedia dataset after we removed all triples containing literals. In Table 1, we provide a brief overview of benchmark datasets.

Table 1: Overview of datasets in terms of number of entities, number of relations, and node degrees in the train split along with the number of triples in each split of the dataset.

Dataset	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{R}\rvert$	Degr. (M±SD)	$\lvert\mathcal{G}^{\text{Train}}\rvert$	$\lvert\mathcal{G}^{\text{Validation}}\rvert$	$\lvert\mathcal{G}^{\text{Test}}\rvert$
DBpedia	114,747,965	13,906	-	375,900,462	-	-
YAGO3-10	123,182	37	9.6±8.7	1,079,040	5,000	5,000
FB15K	14,951	1,345	32.46±69.46	483,142	50,000	59,071
WN18	40,943	18	3.49±7.74	141,442	5,000	5,000
FB15K-237	14,541	237	19.7±30	272,115	17,535	20,466
WN18RR	40,943	11	2.2±3.6	86,835	3,034	3,134

5.2 Evaluation

We used publicly available pretrained ConEx, QMult, OMult, ConvQ, and ConvO. We employ the standard metrics *filtered* Mean Reciprocal Rank (MRR) and hits at N (H@N) for link prediction [Dettmers et al., 2018, Balažević et al., 2019a]. For each test triple $(\mathtt{h}, \mathtt{r}, \mathtt{t})$, we construct its reciprocal $(\mathtt{t}, \mathtt{r}^{-1}, \mathtt{h})$ and add it into $\mathcal{G}^{\text{test}}$ which is a common technique to decrease the computational cost during testing [Dettmers et al., 2018]. Then, for each test triple $(\mathtt{h}, \mathtt{r}, \mathtt{t})$, we compute the score of $(\mathtt{h}, \mathtt{r}, \mathtt{x})$ triples for all $x \in \mathcal{E}$ and calculate the filtered ranking $rank_t$ of the triple having t. Then we compute the MRR: $\frac{1}{|\mathcal{G}^{\text{test}}|} \sum_{(\mathtt{h},\mathtt{r},\mathtt{t}) \in \mathcal{G}^{\text{test}}} \frac{1}{rank_t}$. Consequently, given a $(\mathtt{h}, \mathtt{r}, \mathtt{t}) \in \mathcal{G}^{\text{test}}$, we compute ranks of missing entities based on *the rank of head and tail entities* as similarly done in Balažević et al. [2019a,b], Dettmers et al. [2018].

5.3 Reproducibility

We used pretrained models provided in Demir et al. [2021a]. We refer to the project page to reproduce our link prediction experiments[6].

6 Results

6.1 Link Prediction

Table 2 reports link prediction performances on all three benchmark datasets. Results indicate that constraining predictions via the range information of relations increase generalization performances of all approaches on all datasets. This is an important finding as results suggest that

- KGE approaches do not fully capture the range information of relations, and
- generalization performances can be increased by only a look-up operation.

[6] `https://github.com/dice-group/Convolutional-Hypercomplex-Embeddings-for-Link-Prediction`

Table 2: Link prediction results on WN18RR, F15K-237 and YAGO3-10. Results are obtained from corresponding papers. Bold entries denote best results. The dash(-) denotes values missing in the papers. † represents applying the range constraint at prediction time

	WN18RR				**FB15K-237**				**YAGO3-10**			
	MRR	@1	@3	@10	MRR	@1	@3	@10	MRR	@1	@3	@10
TransE [Ruffinelli et al., 2019]	.228	.053	.368	.520	.313	.221	.347	.497	-	-	-	-
ConvE [Ruffinelli et al., 2019]	.442	.411	.451	.504	.339	.248	.359	.521	-	-	-	-
TuckER [Balažević et al., 2019b]	.470	.443	.482	.526	.358	.266	.394	.544	-	-	-	-
A2N [Bansal et al., 2019]	.450	.420	.460	.510	.317	.232	.348	.486	-	-	-	-
QuatE [Zhang et al., 2019]	.482	.436	.499	**.572**	.311	.221	.342	.495	-	-	-	-
HypER [Balažević et al., 2019a]	.465	.436	.477	.522	.341	.252	.376	.520	.533	.455	.580	.678
DistMult [Dettmers et al., 2018]	.430	.390	.440	.490	.240	.160	.260	.420	.340	.240	.380	.540
ConvE [Dettmers et al., 2018]	.430	.400	.440	.520	.335	.237	.356	.501	.440	.350	.490	.620
ComplEx [Dettmers et al., 2018]	.440	.410	.460	.510	.247	.158	.275	.428	.360	.260	.400	.550
REFE [Chami et al., 2020]	.455	.419	.470	.521	.302	.216	.330	.474	.370	.289	.403	.527
ROTE [Chami et al., 2020]	.463	.426	.477	.529	.307	.220	.337	.482	.381	.295	.417	.548
ATTE [Chami et al., 2020]	.456	.419	.471	.526	.311	.223	.339	.488	.374	.290	.410	.538
ComplEx-N3 [Chami et al., 2020]	.420	.390	.420	.460	.294	.211	.322	.463	.336	.259	.367	.484
MuRE [Chami et al., 2020]	.458	.421	.471	.525	.313	.226	.340	.489	.283	.187	.317	.478
RotatE [Sun et al., 2019]	.476	.428	.492	.571	.338	.241	.375	.533	.495	.402	.550	.670
QMult [Demir et al., 2021a]	.438	.393	.449	.537	.346	.252	.383	.535	.555	.475	.602	.698
OMult [Demir et al., 2021a]	.449	.406	.467	.539	.347	.253	.383	.534	.543	.461	.592	.692
ConvQ [Demir et al., 2021a]	.457	.424	.470	.525	.343	.251	.376	.528	.539	.459	.587	.687
ConvO [Demir et al., 2021a]	.458	.427	.473	.521	.366	.271	.403	.543	.489	.395	.546	.664
ConEx [Demir and Ngomo, 2021]	.481	.448	.493	.550	.366	.271	.403	.555	.552	.474	.601	.696
QMult †	.473	.427	.491	.566	.382	.285	.421	.576	**.576**	**.490**	**.631**	**.728**
OMult †	**.484**	**.444**	**.504**	.563	.381	.284	.418	.576	.563	.480	.612	.716
ConvQ †	.474	.442	.486	.535	.375	.280	.409	.568	.497	.403	.553	.672
ConvO †	.471	.441	.484	.529	**.398**	**.301**	**.437**	.592	.545	.463	.594	.694
ConEx †	**.491**	**.457**	**.504**	.561	**.398**	**.301**	**.437**	**.595**	.565	.484	.613	.709

Discussion. In Table 2, we did not report link prediction results with *ElasticReg* as our initial experiments show that *ElasticReg* often detriment link prediction results of approaches. The inferior performance may stem from the fact that embeddings learned via *ElasticReg* may not be tailored to tackle the link prediction. Another possible explanation may be that we did not optimize hyperparameters of models for *ElasticReg*. In our future work, we plan to incorporate the domain and range information of relations in the cross entropy loss function.

6.2 Scalability

We have successfully used the DAIKIRI-Embedding framework to learn DBpedia embeddings for over 100 million entities (see Table 1). We made the pre-trained ConEx embedding of DBpedia publicly available on the HOBBIT platform [7]. During this computation, we used all 32 CPUs in the training phase as well as in the preprocessing phase. Training 1.1 billions of parameters took only few days. Currently, we are working using these embeddings in many different applications including fact-checking.

7 Conclusion

In this work, we introduced a technique that constrains predictions of knowledge graph embedding models by means of incorporating the domain and range information of relations. Our results indicate that generalization performances of many knowledge graph embedding models are increased on all benchmark datasets without requiring any extra computation. Hence, the proposed technique can be readily applied on pre-trained models. Moreover, we show that DAIKIRI-Embedding can be easily used to scale on large knowledge graphs. In future we will work on investigating introducing constraints in the loss function of knowledge graph embedding models.

References

Ivana Balažević, Carl Allen, and Timothy M Hospedales. Hypernetwork knowledge graph embeddings. In *International Conference on Artificial Neural Networks*, pages 553–565. Springer, 2019a.

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

Trapit Bansal, Da-Cheng Juan, Sujith Ravi, and Andrew McCallum. A2n: attending to neighbors for knowledge graph inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4387–4392, 2019.

Ines Chami, Adva Wolf, Da-Cheng Juan, Frederic Sala, Sujith Ravi, and Christopher Ré. Low-dimensional hyperbolic knowledge graph embeddings. *arXiv preprint arXiv:2005.00545*, 2020.

Spencer Chang. Scaling knowledge access and retrieval at airbnb. airbnb medium blog, 2018.

Caglar Demir and Axel-Cyrille Ngonga Ngomo. A physical embedding model for knowledge graphs. In *Joint International Semantic Technology Conference*, pages 192–209. Springer, 2019.

[7] Conex embedding of DBpedia in the HOBBIT platform `https://bit.ly/3jywxxl`.

Caglar Demir and Axel-Cyrille Ngonga Ngomo. Convolutional complex knowledge graph embeddings. In *Eighteenth Extended Semantic Web Conference - Research Track*, 2021. URL `https://openreview.net/forum?id=6T45-4TFqaX`.

Caglar Demir, Diego Moussallem, Stefan Heindorf, and Axel-Cyrille Ngonga Ngomo. Convolutional hypercomplex embeddings for link prediction. *arXiv preprint arXiv:2106.15230*, 2021a.

Caglar Demir, Diego Moussallem, and Axel-Cyrille Ngonga Ngomo. A shallow neural model for relation prediction. In *2021 IEEE 15th International Conference on Semantic Computing (ICSC)*, pages 179–182. IEEE, 2021b.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Jeffrey Scott Eder. Knowledge graph based search system, June 21 2012. US Patent App. 13/404,109.

William Falcon and Kyunghyun Cho. A framework for contrastive self-supervised learning and designing a new approach. *arXiv preprint arXiv:2009.00104*, 2020.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

F Hamad, Isaac Liu, and X Zhang. Food discovery with uber eats: Building a query understanding engine. *Uber Engineering*, 2018.

Q He, B Chen, and D Agarwal. Building the linkedin knowledge graph. *Engineering. linkedin. com*, 2016.

Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, José Emilio Labra Gayo, Sabrina Kirrane, Sebastian Neumaier, Axel Polleres, Roberto Navigli, Axel-Cyrille Ngonga Ngomo, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and Antoine Zimmermann. Knowledge graphs, 2021.

Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S Yu. A survey on knowledge graphs: Representation, acquisition and applications. *arXiv preprint arXiv:2002.00388*, 2020.

Arun Krishnan. Making search easier: How amazon's product graph is helping customers find products more easily. amazon blog, 2018.

Denis Krompaß, Stephan Baier, and Volker Tresp. Type-constrained representation learning in knowledge graphs. In *International semantic web conference*, pages 640–655. Springer, 2015.

Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196, 2014.

Stanislav Malyshev, Markus Krötzsch, Larry González, Julius Gonsior, and Adrian Bielefeldt. Getting the most out of wikidata: semantic technology usage in wikipedia's knowledge graph. In *International Semantic Web Conference*, pages 376–394. Springer, 2018.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, 2015.

R Pittman. Cracking the code on conversational commerce. *eBay Blog*, 2017.

Matthew Rocklin. Dask: Parallel computation with blocked algorithms and task scheduling. In Kathryn Huff and James Bergstra, editors, *Proceedings of the 14th Python in Science Conference*, pages 130 – 136, 2015.

Daniel Ruffinelli, Samuel Broscheit, and Rainer Gemulla. You can teach an old dog new tricks! on training knowledge graph embeddings. In *International Conference on Learning Representations*, 2019.

Muhammad Saleem, Maulik R Kamdar, Aftab Iqbal, Shanmukha Sampath, Helena F Deus, and Axel-Cyrille Ngonga Ngomo. Big linked cancer data: Integrating linked tcga and pubmed. *Journal of web semantics*, 27:34–41, 2014.

Amit Singhal. Introducing the knowledge graph: things, not strings. *Official google blog*, 5, 2012.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*, 2019.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, pages 2071–2080, 2016.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, 2017.

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

Shuai Zhang, Yi Tay, Lina Yao, and Qi Liu. Quaternion knowledge graph embeddings. In *Advances in Neural Information Processing Systems*, pages 2731–2741, 2019.