DAIKIRI
Erklärbare Diagnostische KI für industrielle Daten

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

Deliverable 2.1
Knowledge Graph Embeddings

Dissemination Level	Public
Due Date of Deliverable	Month 15, 31/03/2021
Actual Submission Date	Month 15, 31/03/2021
Work Package	WP2 — Embeddings
Task	T2.1, T2.2
Type	Report
Approval Status	Final
Version	1.0
Number of Pages	17

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	31/03/2021	Final version created	Caglar Demir

Author List

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

Executive Summary

Knowledge graph embedding methods learn continuous vector representations for knowledge graphs and have been used successfully in a large number of applications. In this work, we present our six knowledge graph embedding models that were developed within the DAIKIRI project. All our models can scale well on large knowledge graphs as they retain a linear space complexity in the number of entities in knowledge graphs. Our first model (SHALLOM) effectively infers missing relations given entities on knowledge graphs. Our experiments show that SHALLOM only requires a maximum training time of 8 minutes on benchmark datasets. Our second model (CONEX) learns complex-valued embeddings of entities and relations via combining a 2D convolution with a Hermitian inner product. By virtue of its novel architecture, CONEX reaches a new state-of-the-art performance on benchmark datasets for the link prediction problem. Motivated by these results, we extended CONEX into the quaternions and octonions. We first proposed QMULT and OMULT that apply quaternion and octonion multiplications to learn hypercomplex-valued embeddings of entities and relations. Next, we proposed combining 2D convolution operations with hypercomplex multiplications in a fashion akin to combining a 2D convolution with a Hermitian inner product. CONVQ and CONVO extends QMULT, OMULT by combining 2D convolutions with quaternion and octonion multiplication. Within the DAIKIRI project, we developed two open-source software libraries. The vectograph library allows to automatically create knowledge graph from tabular data[1]. The DICE Embeddings library contains scalable implementations of our models that can leverage multi CPUs, GPUs and even TPUs[2].

[1] `https://github.com/dice-group/vectograph`
[2] `https://github.com/dice-group/dice-embeddings`

Contents

1 Introduction

The number and size of Knowledge Graphs (KGs) available on the Web and in companies grows steadily.[3] For example, more than 150 billion facts describing more than 3 billion things are available in the more than 10,000 knowledge graphs published on the Web as Linked Data.[4] The wealth of knowledge available in KGs also serves as background data for an increasing number of intelligent applications [Wang et al., 2017]. Knowledge Graph Embedding (KGE) methods learn continuous vector representations for knowledge graphs and have been used successfully in many domains. Applications of KGEs include collective machine learning, type prediction, link prediction, entity resolution, knowledge graph completion, question answering, product recommendation [Nickel et al., 2015, Ji et al., 2020].

In this work package, we give an overview of our knowledge graph embedding models that are developed within the DAIKIRI project. To this end, we first provide a background knowledge in Section 2. Our six knowledge graph embedding models are elucidated in Section 3. Next, we briefly described our two open-source software libraries in Section 4. In Section 5, we report prediction performances of all our approaches on benchmark dataset. Finally, we conclude with Section 6.

2 Background

2.1 Link Prediction

Let $\mathcal{E}$ and $\mathcal{R}$ represent the sets of entities and relations. Then, a KG can be formalized as a set of triples $\mathcal{G} = \{(\mathtt{h}, \mathtt{r}, \mathtt{t}) \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}\}$ where each triple contains a head and tail entity $\mathtt{h}, \mathtt{t} \in \mathcal{E}$ and a relation $\mathtt{r} \in \mathcal{R}$. The link prediction task addresses the problem of predicting whether unseen triples (i.e., triples not found in $\mathcal{G}$) are true [Ji et al., 2020]. For a scoring function $\psi : \mathcal{E} \times \mathcal{R} \times \mathcal{E} \mapsto \mathbb{R}$, it should hold that $\psi(\mathtt{h}, \mathtt{r}, \mathtt{t}) > \psi(\mathtt{x}, \mathtt{y}, \mathtt{z})$ if and only if $(\mathtt{h}, \mathtt{r}, \mathtt{t})$ is true [Nickel et al., 2015].

2.2 Convolution

A convolution is an integral expressing the amount of overlap of one function f as it is shifted over another function g. Formally, the convolution operation over a finite range $[0, \tau]$ is given by

$$(f * g)(t) = \int_0^\tau f(\tau) g(t - \tau) d\tau \tag{1}$$

where $*$ denotes the convolution operation. f is often called the input while g is called the kernel (or filter). The output of the $f * g$ is referred as the feature map. In practice, the input often denotes a multidimensional vector of data while the kernel is a multidimensional array of parameters that are adapted by the learning algorithm. Suppose that f represents a 2-dimensional image and g denotes a 2-dimensional kernel. Then, 1 can be rewritten as

$$(f * g)(i, j) = \sum_m \sum_n f(m, n) g(i - m, j - n), \tag{2}$$

where i, j denotes the coordinate in 2-D input. We refer to the chapter 9 in [Goodfellow et al., 2016] for more details on the convolution operation.

[3] `https://lod-cloud.net/`

[4] `lodstats.aksw.org`

2.3 Hypercomplex Numbers

The quaternions are a 4-dimensional algebra [Hamilton, 1844]. A quaternion number $Q \in \mathbb{H}$ is defined as $Q = a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}$ where a, b, c, d are real numbers and $\mathbf{i}, \mathbf{j}, \mathbf{k}$ are imaginary units satisfying Hamilton's rule: $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$. $Q^{\triangleleft} = Q/|Q|$ denotes a unit normalized quaternion with $|Q| = \sqrt{a^2 + b^2 + c^2 + d^2}$. Let Q_1 and $Q_2 \in \mathbb{H}$ be two quaternions, the inner product $Q_1 \cdot Q_2 \in \mathbb{R}$ of two quaternions is defined as

$$Q_1 \cdot Q_2 = a_1 a_2 + b_1 b_2 + c_1 c_2 + d_1 d_2. \tag{3}$$

The quaternion multiplication of Q_1 and Q_2 is defined as

$$\begin{aligned} Q_1 \otimes Q_2 = &(a_1 a_2 - b_1 b_2 - c_1 c_2 - d_1 d_2) \\ &+ (a_1 b_2 + b_1 a_2 + c_1 d_2 - d_1 c_2)\ \mathbf{i} \\ &+ (a_1 c_2 - b_1 d_2 + c_1 a_2 + d_1 b_2)\ \mathbf{j} \\ &+ (a_1 d_2 + b_1 c_2 - c_1 b_2 + d_1 a_2)\ \mathbf{k}. \end{aligned} \tag{4}$$

Equation (3) and Equation (4) can be considered as scalar-valued functions that the former maps two quaternions into a real number, while the latter maps two quaternions into a quaternion. For a d-dimensional quaternion vector $\mathrm{a} + \mathrm{b}\ \mathbf{i} + \mathrm{c}\ \mathbf{j} + \mathrm{d}\ \mathbf{k}$ with $a, b, c, d \in \mathbb{R}^d$, the inner product and multiplication is defined accordingly.

The Octonions are an 8-dimensional algebra where an octonion number $O_1 \in \mathbb{O}$ is defined as $O_1 = x_0 + x_1\mathbf{e}_1 + x_2\mathbf{e}_2 + \ldots + x_7\mathbf{e}_7$, where $\mathbf{e}_1, \mathbf{e}_2 \ldots \mathbf{e}_7$ are imaginary units [Baez, 2002]. Their product ($\bigstar$), inner product ($\cdot$) and vector operations are defined analogously to quaternions. Let $O_1 = x_0 + x_1\mathbf{e}_1 + x_2\mathbf{e}_2 + x_3\mathbf{e}_3 + x_4\mathbf{e}_4 + x_5\mathbf{e}_5 + x_6\mathbf{e}_6 + x_7\mathbf{e}_7$ and $O_2 = y_0 + y_1\mathbf{e}_1 + y_2\mathbf{e}_2 + y_3\mathbf{e}_3 + y_4\mathbf{e}_4 + y_5\mathbf{e}_5 + y_6\mathbf{e}_6 + y_7\mathbf{e}_7$ be two octonions, then the inner product of $O_1 \cdot O_2 \in \mathbb{R}$ is obtained by taking the inner products between corresponding scalars and imaginary units and summing up the four inner products:

$$O_1 \cdot O_2 = x_0 y_0 + x_1 y_1 + x_2 y_2 + x_3 y_3 + x_4 y_4 + x_5 y_5 + x_6 y_6 + x_7 y_7 \tag{5}$$

The octonion multiplication $O_1 \bigstar O_2$ of O_1 and O_2 is defined as

$$\begin{aligned} &(x_0 y_0 - x_1 y_1 - x_2 y_2 - x_3 y_3 - x_4 y_4 - x_5 y_5 - x_6 y_6 - x_7 y_7) \\ &+(x_0 y_1 + x_1 y_0 + x_2 y_3 - x_3 y_2 + x_4 y_5 - x_5 y_4 - x_6 y_7 + x_7 y_6)\ \mathbf{e}_1 \\ &+(x_0 y_2 - x_1 y_3 + x_2 y_0 + x_3 y_1 + x_4 y_6 + x_5 y_7 - x_6 y_4 - x_7 y_5)\ \mathbf{e}_2 \\ &+(x_0 y_3 + x_1 y_2 - x_2 y_1 + x_3 y_0 + x_4 y_7 - x_5 y_6 + x_6 y_5 - x_7 y_4)\ \mathbf{e}_3 \\ &+(x_0 y_4 - x_1 y_5 - x_2 y_6 - x_3 y_7 + x_4 y_0 + x_5 y_1 + x_6 y_2 + x_7 y_3)\ \mathbf{e}_4 \\ &+(x_0 y_5 + x_1 y_4 - x_2 y_7 + x_3 y_6 - x_4 y_1 + x_5 y_0 - x_6 y_3 + x_7 y_2)\ \mathbf{e}_5 \\ &+(x_0 y_6 + x_1 y_7 + x_2 y_4 - x_3 y_5 - x_4 y_2 + x_5 y_3 + x_6 y_0 - x_7 y_1)\ \mathbf{e}_6 \\ &+(x_0 y_7 - x_1 y_6 + x_2 y_5 + x_3 y_4 - x_4 y_3 - x_5 y_2 + x_6 y_1 + x_7 y_0)\ \mathbf{e}_7. \end{aligned}$$

A d-dimensional octonion-valued vector is defined as $\{x_0 + x_1\mathbf{e}_1 + \cdots + x_7\mathbf{e}_7 : x_0, \ldots, x_7 \in \mathbb{R}^d\}$ with the vector operations being defined correspondingly to quaternions. $O^{\triangleleft} = O/|O|$ denotes a unit normalized octonion with $|O| = \sqrt{x_0^2 + x_1^2 + \cdots + x_7^2}$.

3 Knowledge Graph Embeddings

3.1 Shallom

Link prediction problem refers to predicting missing triples (see Section 2). Most approaches achieve this goal by predicting entities, given an entity and a relation. We predict missing triples via the relation prediction. To this end, we frame the relation prediction problem as a multi-label classification problem and propose a shallow neural model (Shallom) that accurately infers missing relations from entities. Shallom is analogous to C-BOW as both approaches predict a central token (p) given surrounding tokens ((s, o)). We defined Shallom as

$$\psi(s,o) = \sigma\Big(\mathbf{W} \cdot \mathrm{ReLU}\big(\mathbf{H} \cdot \Psi(s,o) + \mathbf{b}_1\big) + \mathbf{b}_2\Big), \tag{6}$$

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 (s, o), $\Psi(s,o)$ returns concatenated embeddings of (s, 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:

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{i}^{|\mathcal{R}|} \Big((\mathbf{y}_i \cdot \log(\hat{\mathbf{y}}_i)) + (1 - \mathbf{y}_i) \cdot \log(1 - \hat{\mathbf{y}}_i) \Big) \tag{7}$$

where $\hat{\mathbf{y}}$ is the vector of predicted probabilities and $\mathbf{y}$ is a binary vector of indicating multi labels. Equation Equation (6) shows that the space complexity of Shallom is linear in the number of entities of the input knowledge graph.

The architecture of Shallom is visualized in Figure 1. To obtain a composite representation of (s, 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.

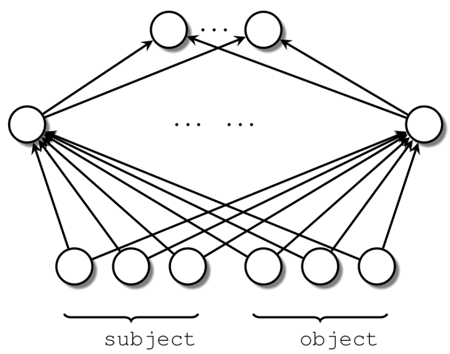

Figure 1: Visualization of Shallom.

3.2 ConEx

Inspired by the previous works ComplEx [Trouillon et al., 2016] and ConvE [Dettmers et al., 2018], we dub our approach ConEx (convolutional complex knowledge graph embeddings).

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 (8)), 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[5]. 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

$$\textsc{ConEx}(h, r, t) = \mathrm{conv}(\mathbf{e}_h, \mathbf{e}_r) \circ \mathrm{Re}(\langle \mathbf{e}_h, \mathbf{e}_r, \overline{\mathbf{e}_t} \rangle), \tag{8}$$

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

$$\mathrm{conv}(\mathbf{e}_h, \mathbf{e}_r) = f\big(\mathrm{vec}(f([\mathbf{e}_h, \mathbf{e}_r] * \omega)) \cdot \mathbf{W} + \mathbf{b}\big), \tag{9}$$

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 to the vanishing gradient problem as the gradients of losses (see Equation (12)) w.r.t. $(\mathbf{e}_h, \mathbf{e}_r, \mathbf{e}_t)$ are allowed to backpropagate through $\mathrm{conv}(\mathbf{e}_h, \mathbf{e}_r)$ or $\mathrm{Re}(\langle \mathbf{e}_h, \mathbf{e}_r, \overline{\mathbf{e}_t} \rangle)$. Equation (8) can be equivalently expressed by expanding its real and imaginary parts:

$$\begin{aligned} \textsc{ConEx}(h, r, t) &= \sum_{k=1}^{d} \mathrm{Re}(\gamma)_k \mathrm{Re}(\mathbf{e}_h)_k \mathrm{Re}(\mathbf{e}_r)_k . \mathrm{Re}(\overline{\mathbf{e}_t})_k && (10) \\ &= \langle \mathrm{Re}(\gamma), \mathrm{Re}(\mathbf{e}_h), \mathrm{Re}(\mathbf{e}_r), \mathrm{Re}(\mathbf{e}_t) \rangle \\ &\quad + \langle \mathrm{Re}(\gamma), \mathrm{Re}(\mathbf{e}_h), \mathrm{Im}(\mathbf{e}_r), \mathrm{Im}(\mathbf{e}_t) \rangle \\ &\quad + \langle \mathrm{Im}(\gamma), \mathrm{Im}(\mathbf{e}_h), \mathrm{Re}(\mathbf{e}_r), \mathrm{Im}(\mathbf{e}_t) \rangle \\ &\quad - \langle \mathrm{Im}(\gamma), \mathrm{Im}(\mathbf{e}_h), \mathrm{Im}(\mathbf{e}_r), \mathrm{Re}(\mathbf{e}_t) \rangle . && (11) \end{aligned}$$

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

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[6]. 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:

$$L = -\frac{1}{|\mathcal{E}|}\sum_{i=1}^{|\mathcal{E}|}(\mathbf{y}^{(i)}\log(\hat{\mathbf{y}}^{(i)}) + (1 - \mathbf{y}^{(i)})\log(1 - \hat{\mathbf{y}}^{(i)})), \tag{12}$$

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 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.

3.3 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.

Given a triple (h, r, t), $\text{QMult} : \mathbb{H}^{3d} \mapsto \mathbb{R}$ 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

$$\text{QMult}(h, r, t) = \mathbf{e}_h \otimes \mathbf{e}_r \cdot \mathbf{e}_t, \tag{13}$$

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

$$\text{OMult}(h, r, t) = \mathbf{e}_h \bigstar \mathbf{e}_r \cdot \mathbf{e}_t, \tag{14}$$

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

[6] 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.

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 (15) and Equation (16):

$$\textsc{ConvQ}(h, r, t) = \mathrm{conv}(\mathbf{e}_h, \mathbf{e}_r) \circ (\mathbf{e}_h \otimes \mathbf{e}_r) \cdot \mathbf{e}_t, \tag{15}$$

$$\textsc{ConvO}(h, r, t) = \mathrm{conv}(\mathbf{e}_h, \mathbf{e}_r) \circ (\mathbf{e}_h \bigstar \mathbf{e}_r) \cdot \mathbf{e}_t, \tag{16}$$

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

$$\mathrm{conv}(\mathbf{e}_h, \mathbf{e}_r) = f\Big(\mathrm{vec}(f([\mathbf{e}_h, \mathbf{e}_r] * \omega)) \cdot \mathbf{W}\Big), \tag{17}$$

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].

4 From Tabular Data to Knowledge Graph Embedding

4.1 Vectograph

We developed an open-source software library for automatically creating a graph structured data from a given tabular data[7]. The worfklow of the vectograph library is twofold: (1) discretization and (2) knowledge graph generation.

Discretization. Discretization is the process of mapping continuous values into discrete counterparts. We are interested in creating a knowledge graph via discretizing input tabular data. In our work, we consider the input tabular data as a continuous dense matrix, i.e., $X \in \mathbb{R}^{n \times d}$. In our notation, $X[i, j]$ stands for the value in i.th row and j.th column, while $X[:, j]$ stands for all values indicated under the j.th column. Given X, we first select those columns such that have a more than k number of unique values, e.g. $|\{X[:, j]\}| > k$. Thereafter, we discretize values indicated with the selected columns into equal-sized buckets. For this purpose, we rely on the quantile-based discretization function provided within the Pandas open-source library. For each selected column of $X[:, j]$, we generate q number of number of quantiles. Note we denote discretized X as $\hat{X}$. $\hat{X}$ is n by d matrix that contains at max d number of categorical features/columns.

Knowledge Graph Generation. We consider i.th row of $\hat{X}[i, :]$ as a Concise Bounded Description[8] (CBD) of i.th node in an RDF knowledge graph. In this setting,

- $\hat{X}$ corresponds to $n \times d$ number of RDF triples,
- $\hat{X}[i, :]$ corresponds to CBD of i.th row/node, and
- The j-th column of $\hat{X}$ is considered as $j.th$ **predicate**

Figure 3 illustrates simplified representation of the generated knowledge graph, where `num_quantile` denotes the number of quantiles for each column to be created, while `min_unique_val_per_column` stands for the minimum number of unique values per column to apply discretization. Hence, the `min_unique_val_per_column` parameter corresponds to the aforementioned k parameter.

We show that the Vectograph library can be readily applied on standard datasets of the scikit-learn library [Pedregosa et al., 2011]. In our project page,[9] we provide several example to ease the usage of our open-source software library. Moreover, the Vectograph library works seamless with the Dice Embeddings open-source library (see Section 4.2) and is already available in the Python Package Index (`https://pypi.org/project/vectograph`).

4.2 Dice Embeddings

The Dice Embeddings open-source library contains scalable implementation of many knowledge graph embedding approaches, including Shallom, ConEx, QMult, OMult, ConvQ, ConvO, DistMult and ComplEx[10]. The all aforementioned models can be trained by using CPUs, GPUs and even TPUs [Demir, 2021]. Embeddings of knowledge graphs are readily created in the comma-separated value (CSV) format after models are trained.

[7] `https://github.com/dice-group/vectograph`
[8] `https://www.w3.org/Submission/CBD`
[9] `https://github.com/dice-group/Vectograph/examples`
[10] `https://github.com/dice-group/DAIKIRI-Embedding`

```
In [1]:
from vectograph.transformers import GraphGenerator
from vectograph.quantizer import QCUT
import pandas as pd
from sklearn import datasets

X, y = datasets.fetch_california_housing(return_X_y=True)
n, m = X.shape
print(f'Shape of input tabular data: {n} x {m}\n')

# Apply Quantile-based discretization function
X_transformed = QCUT(min_unique_val_per_column=6, num_quantile=5).transform(pd.DataFrame(X))

# Add prefix for subjects and Generate knowledge graph
X_transformed.index = 'Event_' + X_transformed.index.astype(str)
kg = GraphGenerator().transform(X_transformed)

# Triples representing first row in X
for s, p, o in kg[:m]:
    print(s, p, o)
```

```
Shape of input tabular data: 20640 x 8

Event_0 Feature_Category_0 0_quantile_4
Event_0 Feature_Category_1 1_quantile_4
Event_0 Feature_Category_2 2_quantile_4
Event_0 Feature_Category_3 3_quantile_1
Event_0 Feature_Category_4 4_quantile_0
Event_0 Feature_Category_5 5_quantile_1
Event_0 Feature_Category_6 6_quantile_4
Event_0 Feature_Category_7 7_quantile_0
```

Figure 3: Usage of the vectograph library on a benchmark dataset provided in the sklearn library.

By using the Dice Embeddings open-source project, we have already made embeddings of following datasets publicly available:

- DBpedia embeddings[11],
- Biopax embeddings[12],
- Carcinogenesis embeddings[13],
- Mutagenesis embeddings[14].

5 Results

In this section, we report the link prediction results on benchmark datasets. We used five of the most commonly used benchmark datasets (WN18, WN18RR, FB15K, FB15K-237 and YAGO3-10). In Table Table 3, we provide a brief overview of benchmark datasets. We relied on the standard metrics (MRR and Hit@N) to quantify prediction performances. For further details pertaining to the metrics, we refer Section 4 in [Demir et al., 2021, Demir and Ngomo, 2021].

Table 1, and Table 2 report relation prediction performances of Shallom on benchmark datasets. Results indicate that training Shallom on benchmark datasets is completed within a few minutes. This is an important result, as it means that our approach can be applied on large knowledge graphs without requiring high-performance hardware. Our experiments on a subset of DBpedia confirmed this observation. Shallom required only few hours on learning embeddings of more than 6 mullion entities. Table 4 and Table 5 report link prediction performances of ConEx on benchmark datasets. We refer Demir and Ngomo [2021] for further details. Table 6 reports link prediction performances of QMult, OMult, ConvQ, and ConvO on WN18RR, FB15K-237 and YAGO3-10 benchmark datasets.

[11] https://hobbitdata.informatik.uni-leipzig.de/KGE/shallom/DBpedia
[12] https://hobbitdata.informatik.uni-leipzig.de/KGE/shallom/Biopax
[13] https://hobbitdata.informatik.uni-leipzig.de/KGE/shallom/Carcinogenesis
[14] https://hobbitdata.informatik.uni-leipzig.de/KGE/shallom/Mutagenesis

Table 1: The mean of Hits@N relation prediction and runtime results on WN18RR and FB15K-237.

	WN18RR				**FB15K-237**			
			Hits			Hits		
	RT	@1	@3	@5	RT	@1	@3	@5
RESCAL	1860±6	.331	.529	.734	5160±4	.115	.327	.456
TransE	960±11	.507	.761	.864	540±10	.774	.899	.918
ComplEx	2160±15	.515	.652	.758	5880±30	.153	.300	.378
CP	840±15	.332	.518	.659	8040±39	.467	.609	.675
DistMult	780±13	.497	.677	.799	1140±8	.092	.176	.428
KGML	840±15	.868	.954	.975	1080±10	.921	.960	.976
RDFDNN	540±8	.819	.967	.985	720±10	.913	.934	.953
RDF2Vec$_{\text{Skip-Gram}}$	310±5	.534	.815	.940	482±6	.518	.600	.677
RDF2Vec$_{\text{CBOW}}$	337±10	.451	.785	.932	472±8	.522	.608	.687
URC		.095	.265	.446		.003	.013	.020
Shallom	610±13	**.874**	**.982**	**.995**	404±8	**.948**	**.993**	**.997**

The superior performance of Shallom stems from: (1) it being a shallow neural model, (2) optimizing the width of the hidden layer, (3) the task and evaluation measures used. By virtue of being a shallow Neural Network (NN), Shallom requires only 562 seconds to train on $|\mathcal{G}| > 10^6$ on a commodity computer. NNs are required to be wide enough (larger than the input dimension) to learn disconnected decision regions [Nguyen et al., 2018]. Lastly, given the example (`Obama`, `Hawaii`), Shallom assigns high scores for `BirthPlace` and low scores for `SpouseOf`. This stems from the fact that input $\mathcal{G}$ does not involve triples such as (`SpouseOf`, `Hawaii`), while it involves many triples (`BirthPlace`, `Hawaii`). Shallom assigns presumably a high score (`Obama`, `BirthPlace`, `Paderborn`) although such a triple is not contained in $\mathcal{G}$. Since the test splits of the benchmark datasets do not involve such false triples, the Hit@N metric quantifies merely the performances of the relation prediction approaches on the valid triples. Ergo, the idea of corrupted triples is not necessary for relation prediction as each entity pair found in the test split is linked with a relation.

The superior performance of ConEx stems from the composition of a 2D convolution with a Hermitian inner product of complex-valued embeddings. Applying the convolution operation on complex-valued embeddings of subjects and predicates permits ConEx to recognize interactions between subjects and predicates in the form of complex-valued feature maps. Through the affine transformation of feature maps and their inclusion into a Hermitian inner product involving the conjugate-transpose of complex-valued embeddings of objects, ConEx can accurately infer various types of relations. Moreover, the number and shapes of the kernels permit to adjust the expressiveness , while ConEx retains the parameter efficiency due to the parameter sharing property of convolutions. ConvQ and ConvO generalize ConEx on the quaternion and octonion-valued embeddings. By virtue of the novel design, the expressiveness of ConEx, ConvQ and ConvO may be further improved by increasing the depth of the conv$(\cdot, \cdot)$ via the residual learning block [He et al., 2016].

Table 2: The mean of Hits@N relation prediction and runtime results on YAGO3-10.

	YAGO3-10			
		Hits		
	RT	@1	@3	@5
$\text{RDF2Vec}_{\text{Skip-Gram}}$	593±11	.487	.796	.875
$\text{RDF2Vec}_{\text{CBOW}}$	625±12	.491	.803	.873
SHALLOM	562±19	**.630**	**.983**	**.996**

Table 3: 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)	$\lvert\mathcal{G}^{\text{Train}}\rvert$	$\lvert\mathcal{G}^{\text{Validation}}\rvert$	$\lvert\mathcal{G}^{\text{Test}}\rvert$
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

Table 4: Link prediction results on WN18 and FB15K. Results are obtained from Balažević et al. [2019b], Zhang et al. [2019].

	WN18				FB15K			
	MRR	Hits@10	Hits@3	Hits@1	MRR	Hits@10	Hits@3	Hits@1
DistMult	.822	.936	.914	.728	.654	.824	.733	.546
ComplEx	.941	.947	.936	.936	.692	.840	.759	.599
ANALOGY	.942	.947	.944	.939	.725	.854	.785	.646
R-GCN	.819	.964	.929	.697	.696	.842	.760	.601
TorusE	.947	.954	.950	.943	.733	.832	.771	.674
ConvE	.943	.956	.946	.935	.657	.831	.723	.558
HypER	.951	.958	.955	.947	.790	.885	.829	.734
SimplE	.942	.947	.944	.939	.727	.838	.773	.660
TuckER	.953	.958	.955	.949	.795	.892	.833	.741
QuatE	.950	.962	.954	.944	.833	.900	.859	.800
CONEX	**.976**	**.980**	**.978**	**.976**	**.872**	**.930**	**.896**	**.837**

Table 5: Link prediction results on WN18RR and FB15K-237. ‡ represents recently reported results of corresponding models.

	WN18RR				**FB15K-237**			
	MRR	Hits@10	Hits@3	Hits@1	MRR	Hits@10	Hits@3	Hits@1
DistMult	.430	.490	.440	.390	.241	.419	.263	.155
ComplEx	.440	.510	.460	.410	.247	.428	.275	.158
ConvE	.430	.520	.440	.400	.335	.501	.356	.237
RESCAL†	.467	.517	.480	.439	.357	.541	.393	.263
DistMult†	.452	.530	.466	.413	.343	.531	.378	.250
ComplEx†	.475	.547	.490	.438	.348	.536	.384	.253
ConvE†	.442	.504	.451	.411	.339	.521	.369	.248
HypER	.465	.522	.477	.436	.341	.520	.376	.252
NKGE	.450	.526	.465	.421	.330	.510	.365	.241
RotatE	.476	.571	.492	.428	.338	.533	.375	.241
TuckER	.470	.526	.482	.443	.358	.544	.394	.266
QuatE	**.482**	**.572**	**.499**	.436	**.366**	**.556**	.401	**.271**
DistMult	.439	.527	.455	.399	.353	.539	.390	.260
ComplEx	.453	.546	.473	.408	.332	.509	.366	.244
TuckER	.466	.515	.476	.441	.363	.553	.400	.268
ConEx	.481	.550	.493	**.448**	**.366**	.555	**.403**	**.271**

6 Conclusion

In this work, we introduced our knowledge graph embedding models that were developed within the DAIKIRI project. Experiments showed that our approaches are not only effective at predicting missing information on a given knowledge graph but retain a linear space complexity in the number of entities in knowledge graphs. This implies that our models can scale on large knowledge graphs. For instance, our experiments show that Shallom computes embeddings of benchmark datasets within a few minutes. In future we will work on

- investigating introducing constraints in knowledge graph embeddings and
- include evaluation scenarios in the DICE embeddings software library.

References

John Baez. The octonions. *Bulletin of the American Mathematical Society*, 39(2):145–205, 2002.

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.

Caglar Demir. Dice embeddings. *Official DICE github project*, 1, 2021. URL `https://github.com/dice-group/dice-embeddings`.

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, 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, 2021.

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

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.

William Rowan Hamilton. Lxxviii. on quaternions; or on a new system of imaginaries in algebra: To the editors of the philosophical magazine and journal. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 25(169):489–495, 1844.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

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.

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.

Quynh Nguyen, Mahesh Chandra Mukkamala, and Matthias Hein. Neural networks should be wide enough to learn disconnected decision regions. *arXiv preprint arXiv:1803.00094*, 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.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

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.

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.

Yanjie Wang, Daniel Ruffinelli, Rainer Gemulla, Samuel Broscheit, and Christian Meilicke. On evaluating embedding models for knowledge base completion. *arXiv preprint arXiv:1810.07180*, 2018.

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.

Table 6: Link prediction results on the WN18RR, F15K-237 and YAGO3-10 datasets in terms of mean reciprocal rank (MRR), and Hits @1, @3 and @10. The models' performance is taken from the corresponding papers with the star(*) denoting values missing in the papers. Param. denotes the reported number of parameters. Bold and underlined entries denote best and second-best results per column. Second rows denote link prediction results of models trained on the training plus the validation sets.

	WN18RR					**FB15K-237**					**YAGO3-10**				
	Param.	MRR	@1	@3	@10	Param.	MRR	@1	@3	@10	Param.	MRR	@1	@3	@10
RESCAL [2018]	*	.420	*	*	.447	*	.270	*	*	.427	*	*	*	*	*
ConvE [2019]	*	.442	*	*	.504	*	.339	*	*	.521	*	*	*	*	*
NKGE [2019]	*	.45	.42	.47	.53	*	.33	.24	.37	.51	*	*	*	*	*
DistMult [2019]	*	.43	.39	.44	.49	*	.28	.20	.30	.44	*	*	*	*	*
A2N [2019]	*	.450	.420	.460	.510	*	.317	.232	.348	.486	*	*	*	*	*
QuatE [2019]	16.38M	**.481**	.436	**.50**	.564	5.82M	.311	.221	.342	.495	*	*	*	*	*
pRotatE [2019]	*	.462	.417	.479	.552	*	.328	.230	.365	.524	*	*	*	*	*
HypER [2019a]	*	.465	.436	.477	.522	*	.341	.252	.376	.520	*	.533	.455	.580	.678
DistMult [2018]	*	.430	.390	.440	.490	*	.240	.160	.260	.420	*	.340	.240	.380	.540
ConvE [2018]	*	.430	.400	.440	.520	*	.335	.237	.356	.501	*	.440	.350	.490	.620
ComplEx [2018]	*	.440	.410	.460	.510	*	.247	.158	.275	.428	*	.360	.260	.400	.550
RotatE [2019]	40.95M	.476	.428	.492	**.571**	29.32M	.338	.241	.375	.533	*	.495	.402	.550	.670
QMult	16.39M	.439	.394	.458	.535	6.01M	.347	.252	.383	.535	49.30M				
		.457	.412	.472	.554		.366	.273	**.410**	**.561**		**.547**	**.463**	**.597**	**.697**
ConvQ	21.51M	.457	.424	.469	.524	11.13M	.343	.251	.376	.528	40.26M				
		.474	**.441**	.488	.539		.365	.271	.402	.552		.538	.456	.588	.689
OMult	16.38M	.449	.410	.470	.539	6.01M	.341	.250	.376	.525	49.30M				
		.465	.425	.476	.553		**.367**	.271	**.410**	.557		.533	.446	.581	.693
ConvO	21.51M	.460	.427	.473	.521	11.13M	.341	.250	.376	.525	40.26M				
		.474	**.441**	.488	.539		**.367**	**.274**	.403	.553		.513	.426	.560	.675