Interpretable and Compositional Relation Learning by Joint Training with an Autoencoder

Ryo Takahashi*¹  Ran Tian*¹  Kentaro Inui¹,²

(* equal contribution)

¹Tohoku University  ²RIKEN, Japan
Task: Knowledge Base Completion

- Knowledge Bases (KBs) store a large amount of facts in the form of \(<\text{head entity}, \text{relation}, \text{tail entity}>\) triples:

- The Knowledge Base Completion (KBC) task aims to predict missing parts of an incomplete triple:

- Help discover missing facts in a KB
Vector Based Approach

A common approach to KBC is to model triples with a low dimension vector space, where

**Entity**: represented by a low dimension vector (so that similar entities are close to each other)

**Relation**: represented as transformation of the vector space, which can be:
- Vector Translation
- Linear map
- Non-linear map

Up to design choice
2 Popular Types of Representations for Relation

**TransE [Bordes+’13]**
- Relation as vector translation
  \[
  \begin{bmatrix}
  d \\
  \end{bmatrix}
  +
  \begin{bmatrix}
  d \\
  \end{bmatrix}
  \approx
  \begin{bmatrix}
  d \\
  \end{bmatrix}
  \]
- Intuitively suitable for 1-to-1 relation
- Relation as linear transformation (matrix)

**Bilinear [Nickel+’11]**
- Relation as linear transformation (matrix)
  \[
  \begin{bmatrix}
  d \\
  \end{bmatrix}
  \cdot
  \begin{bmatrix}
  d^2 \\
  \end{bmatrix}
  \cdot
  \begin{bmatrix}
  d \\
  \end{bmatrix}
  \]
- Flexibly modeling N-to-N relation

Same number of entities
Same distances within

Australia \(\rightarrow\) USD

Australia \(\rightarrow\) AUD

US \(\rightarrow\) AUD

USD \(\rightarrow\) AUD

currency of _country_

country of _film_

The Matrix
Finding Nemo

Australia
US
Matrices are Difficult to Train

- **More parameters** compared to entity vector

![Diagram showing comparison between entity vector and relation matrix dimensions](image)

- Objective is **highly non-convex**

\[
\begin{align*}
\mathbf{u}_h^\top \cdot M_r \cdot \mathbf{v}_t &= d \cdot d^2 \cdot d \\
\end{align*}
\]
In this work:

① Propose jointly training relation matrices with an autoencoder:
   • In order to reduce the high dimensionality

② Modified SGD with separated learning rates:
   • In order to handle the highly non-convex training objective

③ Use modified SGD to enhance joint training with autoencoder

④ Other techniques for training relation matrices

Achieve SOTA on standard KBC datasets
TRAINING TECHNIQUES
① Joint Training with an Autoencoder

**Base Model**
Represent relations as matrices in a bilinear model, can be extended with compositional training [Nickel+'11, Guu+'15, Tian+'16]

\[
\begin{align*}
&\begin{bmatrix} u_h^T \end{bmatrix} \cdot \begin{bmatrix} M_{r_1} \end{bmatrix} \cdot \begin{bmatrix} M_{r_2} \end{bmatrix} \cdot \begin{bmatrix} v_t \end{bmatrix} \\
&\begin{array}{c}
\phantom{\begin{bmatrix} u_h^T \end{bmatrix}} \\
\phantom{\begin{bmatrix} M_{r_1} \end{bmatrix}} \\
\phantom{\begin{bmatrix} M_{r_2} \end{bmatrix}} \\
\phantom{\begin{bmatrix} v_t \end{bmatrix}}
\end{array}
\end{align*}
\]

\[
d \cdot d^2 \cdot d^2 \cdot d
\]

Train jointly

**Proposed**
Train an **autoencoder** to reconstruct relation matrix from low dimension coding

\[
\begin{align*}
&\begin{bmatrix} M_r \end{bmatrix} \cdot \begin{bmatrix} c \end{bmatrix} \cdot \begin{bmatrix} M'_r \end{bmatrix} \\
&\begin{array}{c}
\phantom{\begin{bmatrix} M_r \end{bmatrix}} \\
\phantom{\begin{bmatrix} c \end{bmatrix}} \\
\phantom{\begin{bmatrix} M'_r \end{bmatrix}}
\end{array}
\end{align*}
\]

\[
d^2 \cdot c \cdot d^2
\]

Different from usual autoencoders in which the original input is not updated

**Finding**
1. Reduce the high dimensionality of relation matrices
2. Help learn composition of relations
Joint Training with an Autoencoder

Base Model

Represent relations as matrices in a bilinear model, can be extended with compositional training [Nickel+’11, Guu+’15, Tian+’16]

Proposed

Train an autoencoder to reconstruct relation matrix from low dimension coding

Not easy to carry out
Training objective is highly non-convex → Easily fall into local minimums
② Modified SGD (Separated Learning Rates)

Our strategy
Different learning rates for different parts of our model

Previous
The common practice for setting learning rates of SGD [Bottou, 2012]:

$$\alpha(\tau) := \frac{\eta}{1 + \eta \lambda \tau}$$

- $\eta$: initial learning rate
- $\lambda$: coefficient of L2-regularizer
- $\tau$: counter of trained examples

Modified
Different parts in a neural network may have different learning rates

$$\alpha_{\text{KB}}(\tau_r) := \frac{\eta_{\text{KB}}}{1 + \eta_{\text{KB}} \lambda_{\text{KB}} \tau_r}$$

$$\alpha_{\text{AE}}(\tau_r) := \frac{\eta_{\text{AE}}}{1 + \eta_{\text{AE}} \lambda_{\text{AE}} \tau_r}$$

- $\eta_{\text{KB}}$: $\eta$ for KB-learning objective
- $\eta_{\text{AE}}$: $\eta$ for autoencoder objective
- $\lambda_{\text{KB}}$: $\lambda$ for KB-learning objective
- $\lambda_{\text{AE}}$: $\lambda$ for autoencoder objective
- $\tau_e$: counter of each entity $e$
- $\tau_r$: counter of each relation $r$

July 18, 2018
② Modified SGD (Separated Learning Rates)

Our strategy
Different learning rates for different parts of our model

Previous
The common practice for setting learning rates of SGD [Bottou, 2012]:

\[ \alpha(\tau) := \frac{\eta}{1 + \eta \lambda \tau} \]

\( \eta \): initial learning rate
\( \lambda \): coefficient of L2-regularizer

Learning rates for frequent entities and relations can decay more quickly

Modified
Different parts in a neural network may have different learning rates

\[ \alpha_{KB}(\tau_r) := \frac{\eta_{KB}}{1 + \eta_{KB} \lambda_{KB} \tau_r} \]
\[ \alpha_{AE}(\tau_r) := \frac{\eta_{AE}}{1 + \eta_{AE} \lambda_{AE} \tau_r} \]

\( \eta_{KB} \): \( \eta \) for KB-learning objective
\( \eta_{AE} \): \( \eta \) for autoencoder objective
\( \lambda_{KB} \): \( \lambda \) for KB-learning objective
\( \lambda_{AE} \): \( \lambda \) for autoencoder objective

\( \tau_e \): counter of each entity \( e \)
\( \tau_r \): counter of each relation \( r \)
② Modified SGD (Separated Learning Rates)

Our strategy
Different learning rates for different parts of our model

Rationale
NN usually can be decomposed into several parts, each one is convex when other parts are fixed.
NN \approx joint co-training of many simple convex models
Natural to assume different learning rate for each part

Modified
Different parts in a neural network may have different learning rates

\[
\alpha_{\text{KB}}(\tau_e) := \frac{\eta_{\text{KB}}}{1 + \eta_{\text{KB}}\lambda_{\text{KB}}\tau_e}
\]

\[
\alpha_{\text{AE}}(\tau_r) := \frac{\eta_{\text{AE}}}{1 + \eta_{\text{AE}}\lambda_{\text{AE}}\tau_r}
\]

- \(\eta_{\text{KB}}\): \(\eta\) for KB-learning objective
- \(\eta_{\text{AE}}\): \(\eta\) for autoencoder objective
- \(\lambda_{\text{KB}}\): \(\lambda\) for KB-learning objective
- \(\lambda_{\text{AE}}\): \(\lambda\) for autoencoder objective
- \(\tau_e\): counter of each entity \(e\)
- \(\tau_r\): counter of each relation \(r\)
Learning Rates for Joint Training Autoencoder

**KB objective** trying to predict entities

$$\alpha_{KB}(\tau_r) := \frac{\eta_{KB}}{1 + \eta_{KB}\lambda_{KB}\tau_r}$$

**Autoencoder (AE) objective** trying to fit to low dimension coding

$$\alpha_{AE}(\tau_r) := \frac{\eta_{AE}}{1 + \eta_{AE}\lambda_{AE}\tau_r}$$

Beginning of training
- AE is initialized randomly
- Does not make much sense to fit matrices to AE

As the training proceeds
- $\alpha_{KB}$ and $\alpha_{AE}$ should balance
Normalization

normalize relation matrices to $\|M_r\| = \sqrt{d}$ during training.

Regularization

push $M_r$ toward an orthogonal matrix

Initialization

initialize $M_r$ as $(I + G)/2$ instead of pure Gaussian

Minimize $\left\|M_r^\top M_r - \frac{1}{d} \text{tr}(M_r^\top M_r) I \right\|$

+2.6 in Hits@10 on FB15k-237

+1.2 in Hits@10

+0.4 in Hits@10
EXPERIMENTS
Datasets for Knowledge Base Completion

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#Entity</th>
<th>#Relation</th>
<th>#Train</th>
<th>#Valid</th>
<th>#Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>WN18RR</td>
<td>40,943</td>
<td>11</td>
<td>86,835</td>
<td>3,034</td>
<td>3,134</td>
</tr>
<tr>
<td>[Dettmers+’18]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FB15k-237</td>
<td>14,541</td>
<td>237</td>
<td>272,115</td>
<td>17,535</td>
<td>20,466</td>
</tr>
<tr>
<td>[Toutanova&amp;Chen’15]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **WN18RR**: subset of WordNet [Miller ’95]
- **FB15k-237**: subset of Freebase [Bollacker+’08]
- The previous **WN18** and **FB15k** have an information leakage issue (refer our paper for test results)
- Evaluate models by how high the model ranks the gold test triples.
### Base Model vs. Joint Training with Autoencoder

<table>
<thead>
<tr>
<th>Model</th>
<th>WN18RR</th>
<th>FB15k-237</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MR↓</td>
<td>MRR↑</td>
</tr>
<tr>
<td>BASE</td>
<td>2447</td>
<td>.310</td>
</tr>
<tr>
<td>JOINT with AE</td>
<td>2268</td>
<td>.343</td>
</tr>
</tbody>
</table>

**Models:**
- **BASE**: The bilinear model [Nickel+’11]
- **Proposed JOINT Training**: Jointly train relation matrices with an autoencoder

**Metrics:**
- **MR** (Mean Rank): lower is better
- **MRR** (Mean Reciprocal Rank): higher is better
- **H10** (Hits at 10): higher is better

Joint training with an autoencoder improves upon the base bilinear model.
### Compared to Previous Research

<table>
<thead>
<tr>
<th>Model</th>
<th>WN18RR</th>
<th>FB15k-237</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MR ↓ MRR ↑ H10 ↑</td>
<td>MR ↓ MRR ↑ H10 ↑</td>
</tr>
<tr>
<td>Ours</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BASE</td>
<td>2447 .310 54.1</td>
<td>203 .328 51.5</td>
</tr>
<tr>
<td>JOINT with AE</td>
<td><strong>2268</strong> .343 <strong>54.8</strong></td>
<td><strong>197</strong> .331 <strong>51.6</strong></td>
</tr>
</tbody>
</table>

Re-experiments

<table>
<thead>
<tr>
<th>Model</th>
<th>WN18RR</th>
<th>FB15k-237</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransE [Bordes+’13]</td>
<td>4311 .202 45.6</td>
<td>278 .236 41.6</td>
</tr>
<tr>
<td>HoIE [Nickel+’16]</td>
<td>8096 .376 40.0</td>
<td>1172 .169 30.9</td>
</tr>
</tbody>
</table>

Published results

<table>
<thead>
<tr>
<th>Model</th>
<th>WN18RR</th>
<th>FB15k-237</th>
</tr>
</thead>
<tbody>
<tr>
<td>ComplEx [Trouillon+’16]</td>
<td>5261 .440 51.0</td>
<td>339 .247 42.8</td>
</tr>
<tr>
<td>ConvE [Dettmers+’18]</td>
<td>5277 <strong>.460</strong> 48.0</td>
<td>246 .316 49.1</td>
</tr>
</tbody>
</table>

- **Base model is competitive enough**
- **Our models achieved state-of-the-art results**
## Compared to Previous Research

<table>
<thead>
<tr>
<th>Model</th>
<th>WN18RR</th>
<th>FB15k-237</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MR ↓</td>
<td>MRR ↑</td>
</tr>
<tr>
<td>BASE</td>
<td>2447</td>
<td>.310</td>
</tr>
<tr>
<td>JOINT with AE</td>
<td>2268</td>
<td>.343</td>
</tr>
<tr>
<td>Re-experiments</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TransE [Bordes+'13]</td>
<td>4311</td>
<td>.202</td>
</tr>
<tr>
<td>HoIE [Nickel+'16]</td>
<td>8096</td>
<td>.376</td>
</tr>
<tr>
<td>Published results</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ComplEx [Trouillon+'16]</td>
<td>5261</td>
<td>.440</td>
</tr>
<tr>
<td>ConvE [Dettmers+'18]</td>
<td>5277</td>
<td><strong>.460</strong></td>
</tr>
</tbody>
</table>

- **Base model is competitive enough**
- **Our models achieved state-of-the-art results**
What Does the Trained Autoencoder Look Like?

- Sparse coding of relation matrices
- Interpretable to some extent
Composition of Relations

- Composition of **two relations** in a KB coincide with a **third relation**:

![Diagram showing composition of relations]

- Extracted 154 examples of compositional relations from FB15k-237
Joint Training Helps Find Compositional Relations

If there is a composition...

Learned relation matrices to indeed comply with the composition

\[ M_{\text{country_of_film}} \cdot M_{\text{currency_of_country}} \approx M_{\text{currency_of_film_budget}} \]

<table>
<thead>
<tr>
<th>Model</th>
<th>↓ MR</th>
<th>↑ MRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>BASE</td>
<td>150±3</td>
<td>.0280±.0010</td>
</tr>
<tr>
<td>JOINT with AE</td>
<td>130±27</td>
<td>.0481±.0090</td>
</tr>
</tbody>
</table>

Joint training with an autoencoder helps discovering compositional constraints
## Conclusion and Discussion

<table>
<thead>
<tr>
<th>Task</th>
<th>Knowledge Base Completion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approach</td>
<td>Entities as low dimension vectors, relations as matrices</td>
</tr>
<tr>
<td>Techniques</td>
<td>Joint training relation matrices with autoencoder to reduce</td>
</tr>
<tr>
<td></td>
<td>dimensionality</td>
</tr>
<tr>
<td></td>
<td>Modified SGD: different learning rates for different parts</td>
</tr>
<tr>
<td></td>
<td>Separated learning rates for updating relation matrices</td>
</tr>
<tr>
<td></td>
<td>Normalization, Regularization, Initialization of relation matrices</td>
</tr>
<tr>
<td>Results</td>
<td>SOTA on WN18RR and FB15k-237</td>
</tr>
<tr>
<td>Analysis</td>
<td>Autoencoder learns sparse and interpretable low dimensional</td>
</tr>
<tr>
<td></td>
<td>coding of relation matrices</td>
</tr>
<tr>
<td></td>
<td>Dimension reduction helps find compositional relations</td>
</tr>
<tr>
<td>Discussion</td>
<td>Modern NNs have a lot of parameters</td>
</tr>
<tr>
<td></td>
<td>Joint training with an autoencoder may reduce dimensionality</td>
</tr>
<tr>
<td></td>
<td>“while the NN is functioning”</td>
</tr>
<tr>
<td></td>
<td>More applications?</td>
</tr>
</tbody>
</table>

July 18, 2018