Incorporating Background Knowledge in TuckER

Joss Rakotobe [1] **Haoyu Zhang** [1]

Abstract

Knowledge graphs are a way to represent a large amount of relational facts coming from real word knowledge. Since that knowledge is generally incomplete, a vast research area, known as *Link Prediction* is devoted to infer possible unknown facts based on the existing ones. We focus on a recent state-of-the-art linear model called TuckER that was introduced for the task of link prediction by Balazevich et al. (Balazevic et al., 2019). In this project, we propose ways to incorporate background knowledge about symmetric and asymmetric relations. We show that our model performs better than the TuckER model on those relations while requiring half of parameters.

1. Introduction

A knowledge graph (**KG**) is a dataset of relational facts between entities coming from real-word knowledge. It is represented as an oriented labelled graph where the vertices represent entities and the edges represent relations. Since they are typically incomplete, meaning that there are missing facts about certain relationships between some entities, there is an active area of research trying to infer possible missing edges in the knowledge graph. Several models have recently come to light to tackle this problem. In particular, Balazevic et al. (Balazevic et al., 2019) proposed a state-of-the-art linear model called TuckER, based on the Tucker decomposition of the binary tensor representation of the graph.

Sometimes, some relations are known to have particular properties such as symmetry, asymmetry or being the inverse of another relation. We refer to this type of knowledge as ”background knowledge”. We believe that enforcing these properties into a model should improve its performance. This kind of work was already performed by Minervini et al. (Minervini et al., 2017) on three linear models, namely TransE, DistMult and ComplEx, and also by Kazemi et al. (Kazemi & Poole, 2018) on SimplE. The results they obtained support our assumption.

In this project, we propose different ways to incorporate background knowledge into the TuckER model. We will show that our new model is fully expressive, i.e. given any ground truth over the triples, there exists an assignment of values to the entity and relation embedding that accurately separates the true triples from false ones, and we derive a dimension bound which guarantees full expressiveness. Finally, we made an extensive set of experiments in order to compare the efficiency of using background knowledge with TuckER.

2. Preliminaries

Let $\mathcal{E} = \{e_1, \dots, e_{n_e}\}$ be the set of entities (e.g. objects, people, words etc.) and $\mathcal{R} = \{r_1, \dots, r_{n_r}\}$ the set of relations between entities. A fact about how an entity subject e_s is in relation r_i with an entity object e_o is represented by a triple $(e_s, r_i, e_o) \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}$. We can represent a knowledge graph as a third order binary tensor $\mathcal{X} \in \mathbb{R}^{n_e \times n_r \times n_e}$ where the (i, j, k) element of $\mathcal{X}$ equals one if (e_i, r_j, e_k) is known to be a true fact, and zero otherwise.

2.1. Link Prediction

In link prediction, we are given a subset $\zeta \subset \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ which represents the a priori known facts. The goal is to learn a scoring function $\phi : \mathcal{E} \times \mathcal{R} \times \mathcal{E} \to \mathbb{R}$. The value of the scoring function $\phi(s, r, o)$ will give an indication about how likely the fact (s, r, o) is to be true or false.

2.2. TuckER (Balazevic et al., 2019)

The idea of the TuckER model was to do a Tucker decomposition of the third order binary tensor representation of the knowledge graph $\mathcal{X}$.

Recall that a Tucker decomposition of a third order tensor $\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ consists in writing

$$\mathcal{T} = \mathcal{Z} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C},$$

with $\mathcal{Z} \in \mathbb{R}^{d_a \times d_b \times d_c}, \mathbf{A} \in \mathbb{R}^{d_1 \times d_a}, \mathbf{B} \in^{d_2 \times d_b}, \mathbf{C} \in \mathbb{R}^{d_3 \times d_c}$.

The idea of TuckER is to view $\mathbf{A}$ as the embedding matrix of the subject entity, $\mathbf{C}$, as the embedding matrix of the object entity, $\mathbf{B}$ as the relation embedding matrix, and $\mathcal{W}$

as the core tensor, which encodes the interaction between the relations and the entities. Since it is more natural to have only one representation embedding for an entity, they chose to have $\mathbf{A} = \mathbf{C}$. Therefore, the objective becomes to learn a core tensor $\mathcal{W}$ and two matrices $\mathbf{E}, \mathbf{R}$ such that $\mathcal{X} \approx \mathcal{W} \times_1 \mathbf{E} \times_2 \mathbf{R} \times_3 \mathbf{E}$.

For that purpose, they defined the scoring function as $\phi(s,r,o) = \mathcal{W} \times_1 \mathbf{v}_s \times_2 \mathbf{w}_r \times_3 \mathbf{v}_o$, where $\mathbf{v}_s$, $\mathbf{w}_r$ and $\mathbf{v}_o$ are the embedding vectors of the entity subject s, the relation r and the entity object o respectively. Finally, they applied a logistic sigmoid function on the score, to get a probabilistic approach.

2.3. Background Knowledge

Typically in link prediction, we refer as *Background Knowledge* a set of prior that we know about the relations. For example, "sibling of" is a symmetric relation. A relation can be reflexive, symmetric, transitive, asymmetric, anti-symmetric or can be the inverse of another relation.
For completeness, we provide the formal definitions of these types of relations in Appendix A.

Since background knowledge is known to be true, it is natural to think that incorporating it to a model should improve the model's performance.
This kind of work was performed by Minervini et al. (Minervini et al., 2017). They focused on incorporating symmetric and inverse relations. Basically, their method was to use a soft regularization term that gets a high value when a background knowledge is not respected, and incorporated it on three link prediction models : TransE, DistMult and ComplEx. The regularization term was tuned with a hyper-parameter that controlled the softness of the constraint: a very high value of the hyper-parameter leads to a hard constraint, whereas a low value is equivalent to no constraint. They have empirically shown that the models could be improved via this regularization on some dataset. The experiment was repeated by Kazemi et al. (Kazemi & Poole, 2018) on a different model called SimplE, and the results were similarly positive.

3. Incorporating Background Knowledge Into TuckER

We propose two different approaches to incorporate background knowledge in TuckER. We focused on symmetric and asymmetric relations. We will denote by $\mathcal{S}$ and $\mathcal{A}$ the set of symmetric and asymmetric relations respectively, and by $\mathcal{O}$ the rest of the relations.

In order to encode these two relation types, we want

$$\left.\begin{aligned} \phi(s,r,o) &= \phi(o,r,s) \quad \text{if } \mathbf{r} \in \mathcal{A} \\ \phi(s,r,o) &= -\phi(o,r,s) \quad \text{if } \mathbf{r} \in \mathcal{S} \end{aligned}\right\} \tag{1}$$

3.1. First Method: TuckER-bk

Our first method to incorporate background knowledge will be to put a hard constraint on the core tensor $\mathcal{W}$ and the relation embedding of the TuckER model.
The following method is inspired by the fact that the scoring function of TuckER can also be expressed as :

$$\phi(s,r,o) = \mathbf{v}_s^\top (\mathcal{W} \times_2 \mathbf{w}_r) \mathbf{v}_o$$

Therefore, if we want Equation (1) to be satisfied, then one way is to enforce $\mathcal{W} \times_2 \mathbf{w}_r$ to be a symmetric (resp. anti-symmetric) matrix if $r \in \mathcal{S}$ (resp. $r \in \mathcal{A}$).

Formally, we let $n_r = n_s + n_a + n_o$ be the number of relations with $n_s = |\mathcal{S}|$ and $n_a = |\mathcal{A}|$. Let $d_r = d_s + d_a$ the dimension of the relation embedding with d_s typically representing approximately the proportion of symmetric relations among the symmetric and asymmetric ones.
We force the first d_a mode-2 slices of the core tensor $\mathcal{W}$ to be anti-symmetric matrices and the last d_s mode-2 slices to be symmetric matrices. Moreover, for each symmetric (resp. asymmetric) relation, we'll force the first d_a (resp. last d_s) components of its embedding to be zero (see Figure 3.1) so that $\mathcal{W} \times_2 \mathbf{w}_r$ will be a symmetric (resp. anti-symmetric) matrix for $r \in \mathcal{S}$ (resp. $r \in \mathcal{A}$). For other relations that are neither symmetric nor asymmetric, we put no restriction on its embedding and we note that any matrix can be represented as the sum of a symmetric matrix and an anti-symmetric matrix. The proof of the last claim can be found in Theorem 4.1.

3.2. Second Method: TuckER-bk-score

Our second method will be to put a hard constraint on the scoring function by setting a new scoring function that is built upon the old scoring function.

$$\phi_2(s,r,o) = \begin{cases} \dfrac{\phi(s,r,o) + \phi(o,r,s)}{2} & \text{if } \mathbf{r} \in \mathcal{S} \\ \dfrac{\phi(s,r,o) - \phi(o,r,s)}{2} & \text{if } \mathbf{r} \in \mathcal{A} \\ \phi(s,r,o) & \text{otherwise} \end{cases}$$

4. Theoretical Analysis

4.1. Full Expressiveness and Embedding Dimensionality

We now define what it means for a model to be fully expressive and show that the method that we used to modify

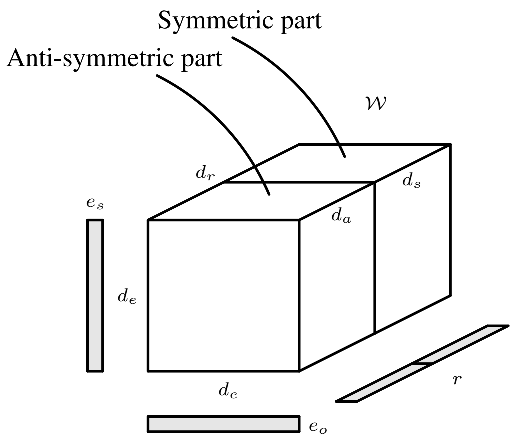

Figure 1. Visualization of the TuckER-bk architecture.

the original TuckER model into our newer models are fully expressive.

Definition 4.1. A link prediction model is fully expressive if for any ground truth over all entities and relations, there exist entity and relation embeddings that accurately separate true triples from the false.

Theorem 4.1. *Given any ground truth ζ over a set of entity $\mathcal{E}$ and relations $\mathcal{R}$, there exists a TuckER-bk model that is fully expressive.*

Proof. We adapt the proof of fully expressiveness of TuckER from the original paper (Balazevic et al., 2019). Suppose WLOG that $\mathcal{A} = \{r_1, \ldots, r_{n_a}\}$ and $\mathcal{S} = \{r_{n_r - n_s + 1}, \ldots, r_{n_r}\}$. We define a temporary tensor $\mathcal{W}' \in \mathbb{R}^{n_e \times n_r \times n_e}$ such that:

$$\mathcal{W}'_{ijk} = \begin{cases} 1 & \text{if } (e_i, r_j, e_k) \in \zeta \\ -1 & \text{if } r_j \in \mathcal{A} \text{ and } (e_k, r_j, e_i) \in \zeta \\ 0 & \text{otherwise;} \end{cases}$$

The core tensor $\mathcal{W}$ is defined to be of size $n_e \times (n_a + 2 \cdot n_o + n_s) \times n_e$ where $\mathcal{W}_{:j:} = \mathcal{W}'_{:j:}$ for $j \leq n_a$ and $\mathcal{W}_{:j+n_o:} = \mathcal{W}'_{:j:}$ for $j \geq n_r - n_s + 1$. For the $2 \cdot n_o$ middle mode-2 slices, we will use the fact that every matrix $\mathbf{A}$ is the sum of a symmetric and anti-symmetric matrix: $\mathbf{A} = \frac{\mathbf{A} + \mathbf{A}^\top}{2} + \frac{\mathbf{A} - \mathbf{A}^\top}{2}$, to define $\mathcal{W}_{:j:} = \frac{\mathcal{W}'_{:j:} - (\mathcal{W}'_{:j:})^\top}{2}$ for $j = n_a + 1, \ldots, n_a + n_o$ and $\mathcal{W}_{:j:} = \frac{\mathcal{W}'_{:j:} + (\mathcal{W}'_{:j:})^\top}{2}$ for $j = n_a + n_o + 1, \ldots, n_a + 2 \cdot n_o$. Finally, using the one hot encoding embedding for the entity, and concatenating $\mathcal{A}, \mathcal{O}, \mathcal{O}$ and $\mathcal{S}$ to get $\mathcal{R}' = (r_1, ..., r_{n_a}, r_{n_a+1}, \ldots, r_{n_a+n_o}, r_{n_a+1}, \ldots, r_{n_a+n_o}, r_{n_a+n_o+1}, \ldots, r_{n_r})$, we define the embedding of relation r_j as a vector where the i-th element equals 1 if the $i - th$ element of $\mathcal{R}'$ is r_j, else 0.

We obtain $\phi(s, r, o) = 1$ if $(s, r, o) \in \zeta$, and otherwise, $\phi(s, r, o) \leq 0$ □

We now show the result of fully expressiveness for our second approach.

Theorem 4.2. *Given any ground truth over a set of entity $\mathcal{E}$ and relations $\mathcal{R}$, there exists a TuckER-bk-score model that is fully expressive.*

Proof. Let $\mathbf{e_E}(\mathbf{s})$ and $\mathbf{e_E}(\mathbf{o})$ be the n_e dimensional one hot binary encoding vector be the embedding of the subject s and object o and $\mathbf{e_R}(\mathbf{r})$ be the n_r dimensional one-hot binary vector representation of relation r. For i^{th} subject entity j^{th} relation type and k^{th} object entity, the i, j, k element of the respective vectors $\mathbf{e_E}(\mathbf{s}), \mathbf{e_E}(\mathbf{o}), \mathbf{e_R}(\mathbf{r})$ be 1 and all other elements 0. Let (i, j, k) coordinates of the core tensor $\mathcal{W}$ be 1 if $(s, r, o) \in \zeta$ and -1 otherwise.

We have three situations:

- r is a symmetric relation:
  - if $(s, r, o) \in \zeta$ then $(o, r, s) \in \zeta$ which implies that
  $$\phi(s, r, o) = \phi(o, r, s) = 1.$$
  Therefore, $\phi_2(s, r, o) = \frac{1+1}{2} = 1$.
  - otherwise, $(o, r, s) \notin \zeta$ which implies that
  $$\phi(s, r, o) = \phi(o, r, s) = -1.$$
  Therefore, $\phi_2(s, r, o) = \frac{-1-1}{2} = 1$.
- r is an asymmetric relation:
  - if $(s, r, o) \in \zeta$ then $(o, r, s) \notin \zeta$ which implies that
  $$\phi(s, r, o) = 1 \text{ and } \phi(o, r, s) = -1.$$
  Therefore, $\phi_2(s, r, o) = \frac{1-(-1)}{2} = 1$.
  - otherwise,
  $$\phi(s, r, o) = -1 \text{ and } \phi(o, r, s) \geq -1.$$
  Therefore, $\phi_2(s, r, o) \leq \frac{-1-(-1)}{2} = 0$.
- r is not symmetric nor asymmetric:
$$\phi_2(s, r, o) = \phi(s, r, o) = \mathbb{1}_{(s,r,o) \in \zeta}.$$

The scores of triples of ζ and its complementary are separated by 0, hence the model is fully expressive.

□

4.2. Representing Asymmetric relations

We saw that another way to view the scoring function of TuckER: $\phi(s, r, o) = \mathbf{v}_s^\top(\mathcal{W} \times_2 \mathbf{w}_r)\mathbf{v}_o$. Namely, it is a bi-linear product of the embedding of the entities with the mode-2 product of the core tensor with the relation embedding of r.

For the original TuckER, the matrix resulting from a mode-2 product of the core tensor with the embedding of an asymmetric relation does not exhibit an obvious structure. However, for the model we get from the first approach, the mode-2 product of the core tensor with the embedding of an asymmetric relation would be an anti-symmetric matrix. Hence, the bi-linear product of entities embedding with the resulting anti-symmetric matrix will flip signs depending on if we were to compute the scoring of (s, r, o) or (o, r, s). Namely, the scoring of (s, r, o) would be negative of the scoring of (o, r, s).

The same is true for our second approach, the scoring of for (s, r, o) would be negative of (o, r, s) if r is an asymmetric relation.

5. Experiments and Results

5.1. Datasets

We now present the dataset that was used in the project.

We have used the WN18RR dataset which is derived from the dataset WN18. The WN18 dataset was introduced by Bordes et al. (Bordes et al., 2013). It included the full 18 relations scraped from WordNet for roughly 41,000 synsets. Similar to FB15K, this dataset was found to suffer from test leakage by Dettmers et al. (Dettmers et al., 2017).
As a way to overcome this problem, Dettmers et al. (2018) introduced the WN18RR dataset, derived from WN18, which features 11 relations only, no pair of which is reciprocal (but still include four internally symmetric relations like verb groups, allowing the rule-based system[1] to reach 35 on all three metrics).

5.2. Experiments

We implemented our models using as foundation the code in PyTorch of Balazevic et al.[2]. We made our code available on GitHub[3].

We provide some detailed explanations about both methods' implementations in Appendix B.

We choose all hyper-parameters based on a grid search. We choose the entity embedding dimension to be $d_e = 200$ and relation embedding dimension to be $d_r = d_s + d_a = 15 + 15$. We used a batch normalization and dropout to speed up the training and improve the generalization.

5.3. Results and interpretation

In order to evaluate the performance of our models, we used 4 metrics, typical of Link Prediction task:

- **MRR** is the Mean Reciprocal Rank of the true triples over all candidate triples.
- **Hits@k** measures the percentage of times a true triple is ranked within the first k candidates.

We used the **Hits@k**, $k \in \{1, 3, 10\}$ metric and the **MRR** to evaluate the performance of our models. Unless specified, all the reported results come from evaluating on a test set.

Our first experiment is plotted in Figure 2. We trained all the models for 200 epochs on the WN18RR dataset. As we can see, there is a threshold starting where the original TuckER model performs better than our models after certain number of training iterations. What we suspect to be happening is that after a while, the original TuckER model Learns to differentiate the symmetric and asymmetric relations as well. In Figure 2, you can also see two heat-maps of the matrices each resulting from the mode-2 product of core tensor with a relation embedding (on the left, we have a symmetric relation, on the right, we have a asymmetric relation). The resulting matrices are quite as we expected them: symmetric for the symmetric relation, anti-symmetric for the asymmetric relation.

We tried to make further investigations on the performance of each model, trying to find their strength and weaknesses. Our second set of experiments is designed to test the robustness of the models to small datasets. We trained our models, again for 200 epochs, but on a random subset of the original training set Figure 3 show the MRR performance result of the three models on different percentages of the training data (the other metric results may be found in Appendix C). We can see that the new models perform better than the original TuckER model for small percentage of data-sets. The small percentage of data means fewer interactions between relations, but for larger dataset, the interactions between relations become more complex. For our first model implementation, the embeddings of asymmetric relations are linearly independent to the symmetric ones. However, some asymmetric relations can be affected by the knowledge of some symmetric relations. For example, knowing a person A is the brother of a person B (symmetric relation) will have an impact on inferring where person B is born (asymmetric relation). This is true for our second implementation as well.

[1]The rule-based system is a simple algorithm that consists of completing the training set with the rules of the relations.

[2]`https://github.com/ibalazevic/TuckER`

[3]`https://github.com/rkbrary/Project-IFT6760A/tree/master/TuckER-master`

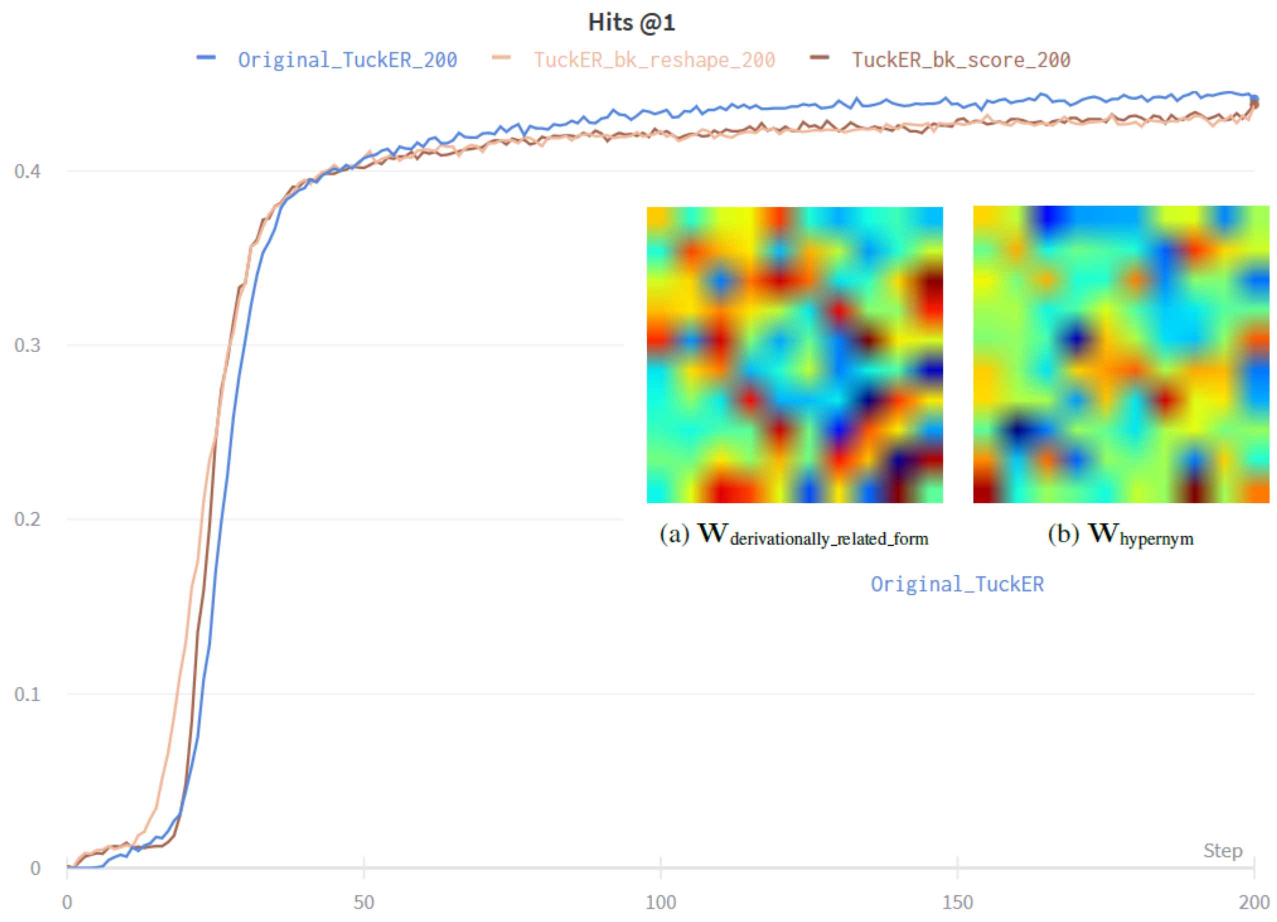

Figure 2. Typical running plots of the models. The evaluation was done at each epoch on a validation set, and one time, at the end of the training, on a test set. The tendency is similar with the other metrics as well.

	TuckER				TuckER-bk				TuckER-bk-score			
	Hits@10	Hits@3	Hits@1	MRR	Hits@10	Hits@3	Hits@1	MRR	Hits@10	Hits@3	Hits@1	MRR
Symmetric	0.9588	0.9570	0.9453	0.9512	0.9606	0.9588	0.9471	0.9530	0.9597	0.9570	0.9462	0.9520
Asymmetric	0.1658	0.1340	0.1022	0.1238	0.1692	0.1358	0.1055	0.1270	0.1691	0.1357	0.1089	0.1284
Both	0.6888	0.6766	0.6527	0.6670	0.6912	0.6760	0.6544	0.6677	0.6830	0.6667	0.6480	0.6600
Neither	0.0978	0.0887	0.0711	0.0808	0.0943	0.0844	0.0703	0.789	0.0971	0.0837	0.0669	0.0777

Table 1. Evaluation of the models on restricted relations

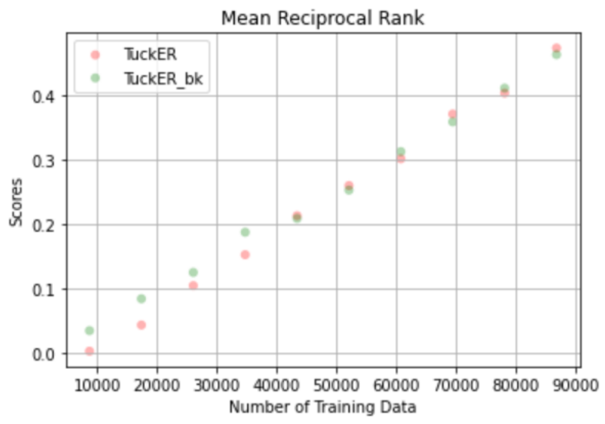

Figure 3. MRR of TuckER and TuckER-bk using different percentages of the training set

So our guess on why we see an increase in the performance of the original TuckER is that without regularizing through the core tensor or the scoring function, it can potentially capture more complex interactions between different types of relations.

Finally, we have examined the performance on each type of relations. We restricted our dataset with certain types of relations. Table 1 shows the results on all models, when restricting to symmetric relations, to asymmetric ones, to both of those two, or to anything but those two. As expected, enforcing symmetry/asymmetry is improving the performance of modelling the symmetric/asymmetric relations. Our first method put the knowledge of a relation being symmetric or anti-symmetric much more on the embedding of the relation itself rather than letting the core tensor figure out the nature of the relation. This is in contrast to the original TuckER model where the model figures out the interactions between the relations and determines the type of a relation. By restricting the data-set on one of the two symmetric or asymmetric relations, the complex interactions between the relations become irrelevant and need not be captured by the model and hence the newer models will perform better. However, when we train with both symmetric and asymmetric relations together, the interactions between the two type of relations is not captured by the new models and the original model can beat the new models, for example, in the **Hits@3** metric. Also, as expected, for relations that is neither symmetric nor asymmetric, the original TuckER model performs better.

6. Conclusion

In our project, we have presented two ways of incorporating the background knowledge of symmetric relations and asymmetric relations into the TuckER model. The advantages of our news models compared to the original TuckER model are that

- They learn faster in early stages in terms of SGD steps
- Tucker-bk needs less parameters to achieve similar results
- They are better at modelling symmetric/asymmetric relations
- They are more robust to small data sets

However, there are disadvantages as well:

- Tucker-bk is slightly slower in terms of computation time due to the reshaping of the parameters[4]
- They are slightly less good at generalizing
- They might not be able to capture more complex interactions between relations of different types.

[4]Python do not allow the use of pointers, as it handles referencing automatically. Perhaps an implementation in C++ could help resolve this issue.

We hope this project contributed to generating new ideas to the development of better models at link prediction tasks on the knowledge graphs.

7. Contribution

This project was done in a team of two:

- Haoyu contributed to modify the codes to do experiments, plot graphs, make good tables, involves in the proof of Fully expressiveness of the newer models, Write good analysis of the experiments and the conclusions, give the backbones of the project report, write the details on the implementation of the models and write the framework of the slides for the presentations.
- Joss contributed by implementing the models, supervising, participating at and verifying all the steps of the project (experiments, graphs and tables to do, slides, report).

We hereby state that all the work presented in this report is that of the authors.

References

Balazevic, I., Allen, C., and Hospedales, T. M. Tucker: Tensor factorization for knowledge graph completion. *CoRR*, abs/1901.09590, 2019.

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O. Translating embeddings for modeling multi-relational data. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 26*, pp. 2787–2795. Curran Associates, Inc., 2013. URL http://papers.nips.cc/paper/5071-translating-embeddings-for-modeling-multi-relational-data.pdf.

Dettmers, T., Minervini, P., Stenetorp, P., and Riedel, S. Convolutional 2d knowledge graph embeddings. *CoRR*, abs/1707.01476, 2017. URL http://arxiv.org/abs/1707.01476.

Kazemi, S. M. and Poole, D. Simple embedding for link prediction in knowledge graphs. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 4284–4295. Curran Associates, Inc., 2018.

Minervini, P., Costabello, L., Muñoz, E., Novácek, V., and Vandenbussche, P.-Y. *Regularizing Knowledge Graph Embeddings via Equivalence and Inversion Axioms*, pp. 668–683. 01 2017. ISBN 978-3-319-71248-2. doi: 10.1007/978-3-319-71249-9_40.

Appendices

A. Definitions of relation types

Definition A.1 (Reflexive). A relations r is called reflexive if

$$\forall s \in \mathcal{E}, (s, r, s) \in \zeta$$

Definition A.2 (Symmetric). A relations r is called symmetric if

$$\forall s, o \in \mathcal{E}, (s, r, o) \in \zeta \implies (o, r, s) \in \zeta$$

.

Definition A.3 (Transitive). A relations r is called transitive if $\forall s, o, t \in \mathcal{E}$,

$$((s, r, o) \in \zeta \wedge (o, r, t) \in \zeta \implies (s, r, t) \in \zeta$$

.

Definition A.4 (Asymmetric). A relations r is called asymmetric if

$$\forall s, o \in \mathcal{E}, (s, r, o) \in \zeta \implies (o, r, s) \notin \zeta$$

.

Definition A.5 (Anti-symmetric). A relations r is called anti-symmetric if $\forall s, o \in \mathcal{E}$,

$$(s, r, o) \wedge (o, r, s) \in \zeta \implies s = o$$

.

Definition A.6 (Inverse). A relation r is the inverse of a relation q if

$$\forall s, o \in \mathcal{E}, (s, r, o) \in \zeta \iff (o, q, s) \in \zeta$$

.

B. Implementation of both methods in Python

We let the maps $e_E : \mathcal{E} \to \mathbb{R}^{d_e}$, $e_R : \mathcal{R} \to \mathbb{R}^{d_r}$ represents the embedding of the entities or relations into finite dimensional vector space. d_e and d_r are the embedding dimensions.

We let the maps $enum_E : \mathcal{E} \to [n_e - 1] \cup \{0\}$ and $enum_r : \mathcal{R} \to [n_r - 1] \cup \{0\}$ be bijective mappings denoting enumerations of the entities and relations respectively. Here, each enumeration starts at 0.

Let d_1, d_2 be our hyper parameters. We let $d_e = d_1$ and $d_r = d_2 + d_2$. Let n_s, n_a be the numbers of symmetric and anti symmetric relations in $\mathcal{R}$. Consider a permutation $\pi \in S_{[n_r-1]\cup\{0\}}$, so that $\pi \circ enum_r(rel_i) \in \{0, 1, ..., n_s - 1\}$ if

rel_i is symmetric. $\pi \circ enum_r(rel_i) \in \{n_s, ..., n_s + n_a - 1\}$ if rel_i is anti-symmetric.

We let $\mathbf{R}_1$ be a random $n_s \times d_2$ matrix.

We let $\mathbf{R}_2$ be a random $n_a \times d_2$ matrix.

We let $\mathbf{R}_3$ be a random $(n_r - n_s - n_a) \times (d_2 + d_2)$ matrix.

We construct matrix $\mathbf{R}$ by using $\mathbf{R}_1, \mathbf{R}_2, \mathbf{R}_3$ as sub-matrices as follows:

$$\mathbf{R} = \left[\begin{array}{c|c} \mathbf{R}_1 & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{R}_2 \\ \hline \multicolumn{2}{c}{\mathbf{R}_3} \end{array}\right]$$

We let $\mathbf{W_1}$ be a $d_2 \times \frac{1+d_1}{2} \times d_1$ random matrix. This matrix will hold the parameters of the core tensor corresponding to slices that represent symmetric matrices.

We let $\mathbf{W_2}$ be a $d_2 \times \frac{d_1-1}{2} \times d_1$ random matrix. This matrix will hold the parameter of the core tensor corresponding to slices that represent anti-symmetric matrices.

We construct the core tensor $\mathcal{W}$ of size $(d_2 + d_2) \times d_1 \times d_1$ from $\mathbf{W_1}$ and $\mathbf{W_2}$. We do it as follows: Each rows of $\mathbf{W_1}$ will fill a upper diagonal $d_1 \times d_1$ matrix of of the form

$$\begin{bmatrix} \mathbf{W_1}[i,1] & \mathbf{W_1}[i,2] & ... & \mathbf{W_1}[i,d_1] \\ 0 & \mathbf{W_1}[i,d_1+1] & ... & \mathbf{W_1}[i,2d_1-1] \\ ... & ... & ... & ... \\ 0 & 0 & ... & \mathbf{W_1}[i, \frac{1+d_1}{2} * d_1] \end{bmatrix}$$

Then we make the matrix into a symmetric one by changing the lower triangular of the matrix accordingly.

$$\mathcal{W}_{i,:,:} = \begin{bmatrix} \mathbf{W_1}[i,1] & \mathbf{W_1}[i,2] & ... & \mathbf{W_1}[i,d_1] \\ \mathbf{W_1}[i,2] & \mathbf{W_1}[i,d_1+1] & ... & \mathbf{W_1}[i,2d_1-1] \\ ... & ... & ... & ... \\ \mathbf{W_1}[i,d_1] & \mathbf{W_1}[i,d_1+1] & ... & \mathbf{W_1}[i, \frac{1+d_1}{2} * d_1] \end{bmatrix},$$

Here $i \in \{0, ..., d_2 - 1\}$.

Each rows of $\mathbf{W_2}$ will first be used to create the upper triangular part of a antisymmetric matrix as follows:

$$\begin{bmatrix} 0 & \mathbf{W_1}[i,1] & \mathbf{W_1}[i,2] & ... & \mathbf{W_1}[i,d_1-1] \\ 0 & 0 & \mathbf{W_1}[i,d_1] & ... & \mathbf{W_1}[i,2d_1] \\ ... & ... & ... & ... & ... \\ 0 & 0 & 0 & ... & \mathbf{W_1}[i, \frac{d_1-1}{2} * d_1] \\ 0 & 0 & 0 & ... & 0 \end{bmatrix}$$

Where, $i \in \{d_2, ..., d_2 + d_2 - 1\}$. We then make it into an anti-symmetric matrix as we did for the symmetric one and we got out core tensor W.

Now, to compute the likelihood of a tuple (ent_i, rel_j, ent_k) being a true fact, we simply compute

$$\sigma(\mathcal{W} \times_2 \mathbf{e_R}(\mathbf{rel_j}) \times_1 \mathbf{e_E}(\mathbf{ent_i}) \times_3 \mathbf{e_E}(\mathbf{ent_k}))$$

where $\sigma(x)$ is the sigmoid function.

For out second approach, We label the relation types based on if they are symmetric, anti-symmetric or others. We construct a mapping that does this:

$$Cons : \mathcal{R} \to \{-1, 0, 1\}$$

where

$$Cons(r) = \begin{cases} 1, & \text{if r is a symmetric relation} \\ -1, & \text{if r is an antisymmetric relation} \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

Then with the third order core tensor $\mathcal{W} \in \mathbb{R}^{d_e \times d_r \times d_e}$, the embedding of entities and relations, e_E, e_R and enumerations of entities and relations, the scoring function for a tuple fact (s, r, o) is $\phi(s, r, o) = 1 - |0.5Cons(r)|(\mathcal{W} \times_2 \mathbf{e_R}(\mathbf{r}) \times_1 \mathbf{e_E}(\mathbf{s}) \times_3 \mathbf{e_E}(\mathbf{o})) + 0.5Cons(r)(\mathcal{W} \times_2 \mathbf{e_R}(\mathbf{r}) \times_3 \mathbf{e_E}(\mathbf{s}) \times_1 \mathbf{e_E}(\mathbf{o}))$. We apply the sigmoid function to the result to give us the likelihood that the tuple (s, r, o) is a true fact.

The training process was done using $1 - N$ scoring as in the original TuckER model. Namely, for each entity relation pairs, we score it against all other entities.

C. Graphs

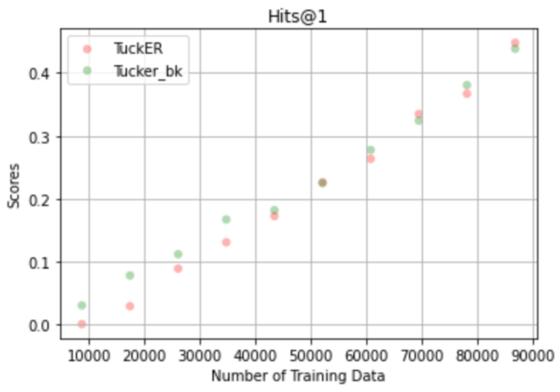

Figure 4. Performance of the models against the number of training data: Hits@1

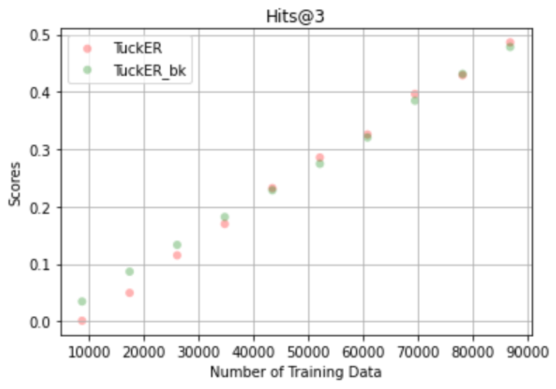

Figure 5. Performance of the models against the number of training data: Hits@3

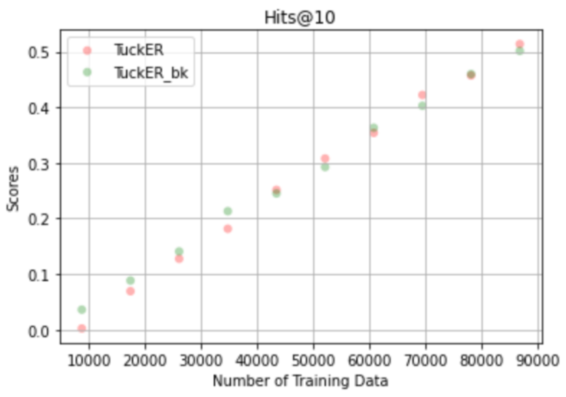

Figure 6. Performance of the models against the number of training data: Hits@10