Reproducibility challenge: TuckER

Mateusz Parafinski, Peter Mernyei, Sharan Gopal, Yufeng Yang

Group 6

Introduction

The goal of this project was to reproduce the results achieved in [1] which introduces TuckER, a linear knowledge graph completion model. TuckER was introduced in 2019 as a powerful generalisation of other linear models such as RESCAL [2] or DistMult [3]. TuckER, despite being a linear model has been shown to perform better than some popular non-linear models such as ConvE [4] or HypER [5] and has the additional benefit of being easily interpretable.

Knowledge graph completion

The task of knowledge graph completion is defined as follows: given a set of entities $\mathcal{E}$, relations $\mathcal{R}$ and known facts $\mathcal{F} \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, we wish to infer any other facts that are likely true but not present in the dataset. An example could be a social network where people who are friends with each other are 'connected' and we want to determine if there are any pairs of people who are friends but not connected in the network.

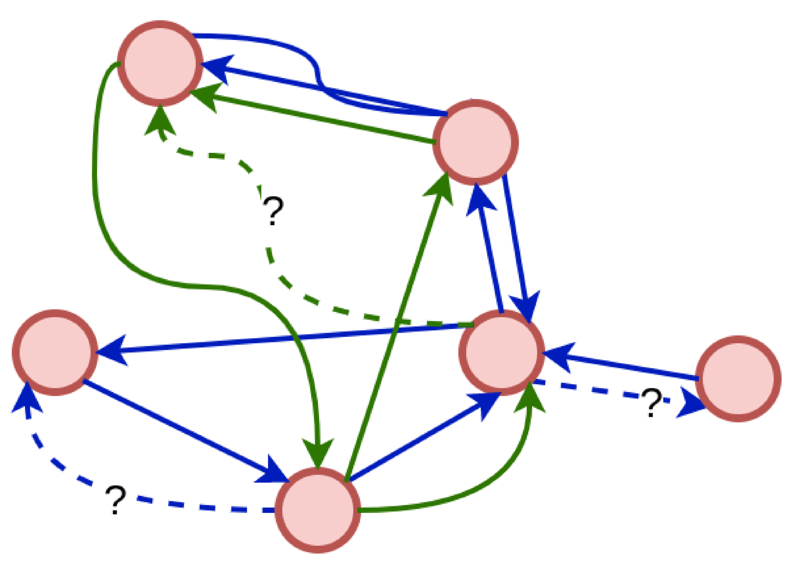

Figure 1:Inferring Missing Links

Mean reciprocal rank (MRR), Hits@k

MRR =

$$\frac{1}{2|G_{test}|} \sum_{r(h,t) \in G_{test}} (rank(h \,|\, r(\_, t)) + rank(t \,|\, r(h, \_)))$$

Hits@k =

$$\frac{1}{2|G_{test}|} \sum_{r(h,t) \in G_{test}} (\mathbf{1}(rank(h \,|\, r(\_, t)) \leq k) + \mathbf{1}(rank(t \,|\, r(h, \_)) \leq k)$$

Implementation (https://github.com/sharan-dce/tucker)

We wrote a TuckER model, as well as the **data loading/processing and evaluation code** in PyTorch. We spotted a bug in the authors' training procedure and **implemented a more efficient evaluation method** (leveraging parallel processing on the evaluation side), with high priority on the code quality.

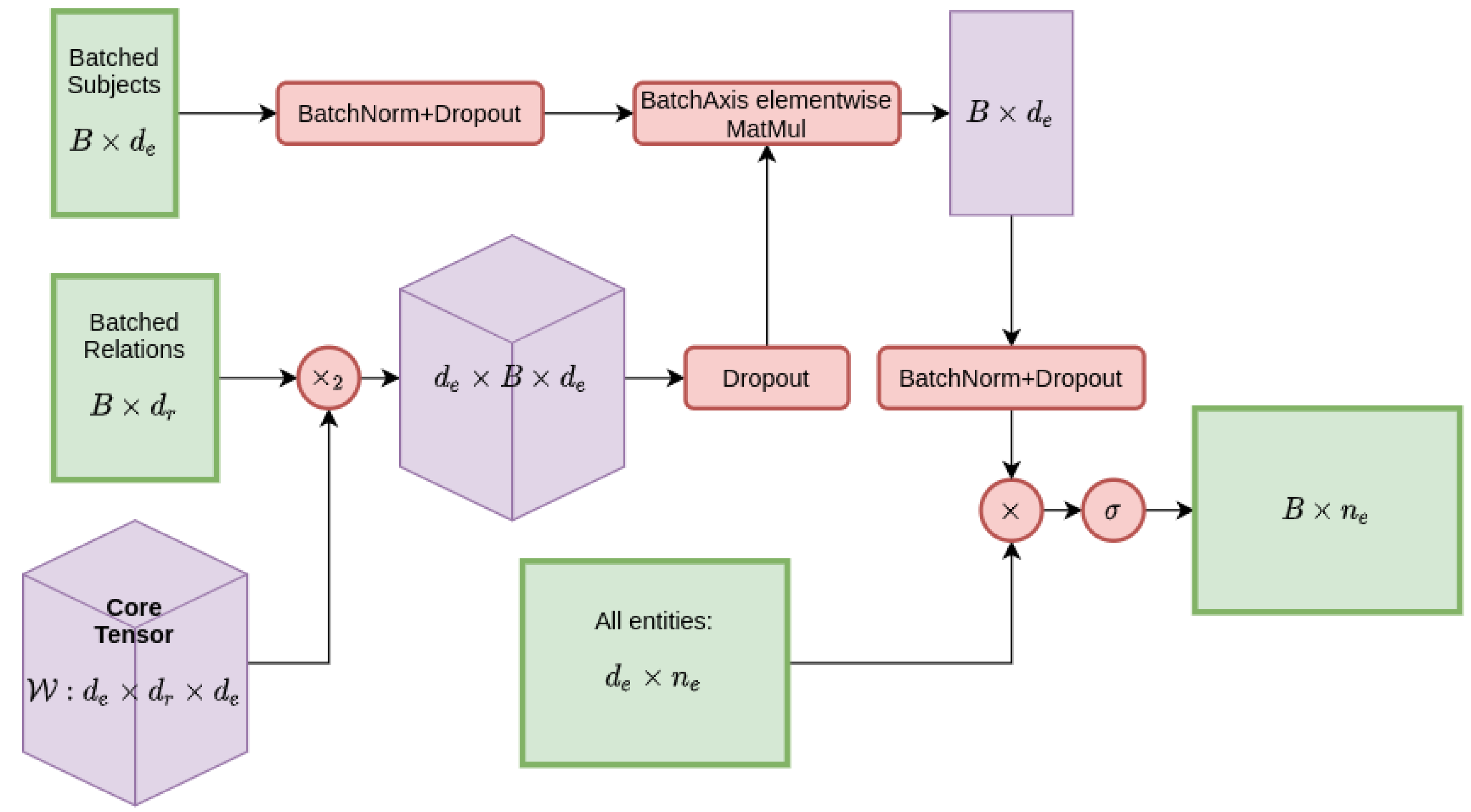

Figure 2:TuckER forward pass with batching.

Reproducibility results

	FB15k				FB15k-237			
	MRR	Hits@10	Hits@3	Hits@1	MRR	Hits@10	Hits@3	Hits@1
TuckER	.795	.892	.833	.741	.358	.544	.394	.266
Our TuckER	.760	.884	.813	.686	.354	.538	.389	.262

	WN18				WN18RR			
	MRR	Hits@10	Hits@3	Hits@1	MRR	Hits@10	Hits@3	Hits@1
TuckER	.953	.958	.955	.949	.470	.526	.482	.443
Our TuckER	.951	.957	.953	.948	.463	.516	.475	.435

Extension: DistMult and RESCAL

We decided to implement other linear models such as RESCAL and DistMult using our TuckER implementation. These are **subclasses of TuckER**:

```
distmult = DistMult(n_e, n_r, d_e)
rescal = RESCAL(n_e, n_r, d_e)
```

The table to the right presents their performance relatively to the original implementation of DistMult on FB15k.

DistMult/RESCAL Results

Our DistMult implementation closely matches the original one in the metrics.

	FB15k			
	MRR	Hits@10	Hits@3	Hits@1
DistMult	.654	.824	.733	.546
Our DistMult	.635	.813	.705	.531
Our RESCAL	.560	.753	.622	.453

Extension: Inference patterns

We created toy datasets to check whether the model is able to capture various inference patterns. The model was able to learn **inversion**, **symmetry** and **composition** relatively quickly and converge to a perfect accuracy – it took $\leq$ 100 epochs to achieve an MRR and hits@1 of 1.0 for each. For **hierarchy**, the model reached an MRR of almost 1.0 in 30 epochs, but then it started declining, ending at 0.76 after 100 epochs.

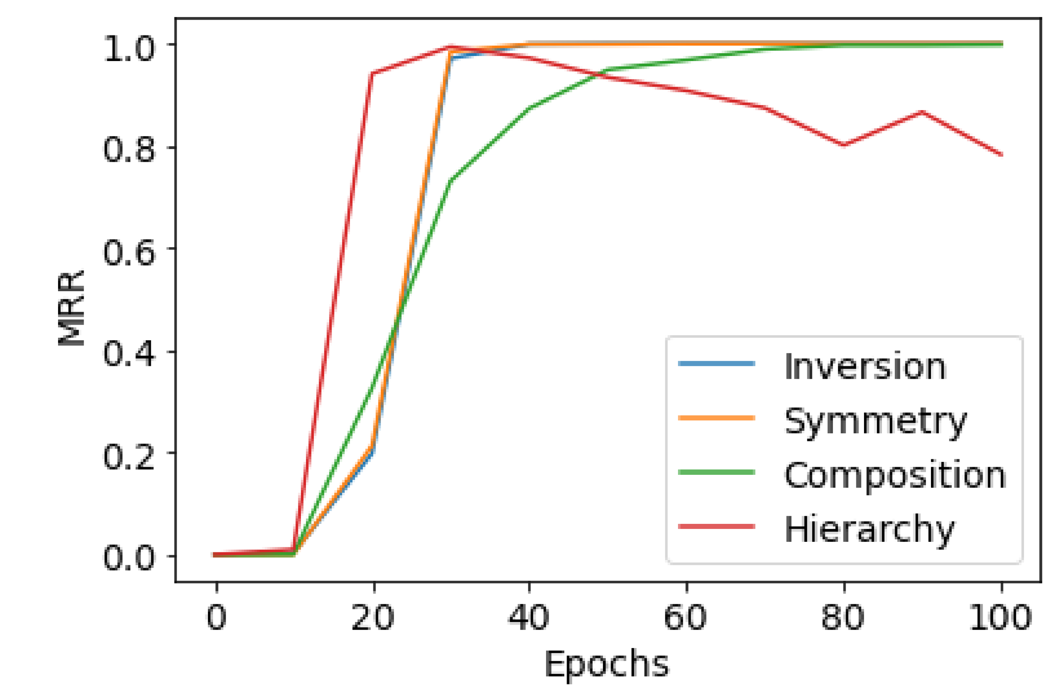

Figure 3:MRR convergence on toy datasets.

Conclusion

TuckER is a simple linear, fully expressive model. From the reproducibility challenge, we discovered that dropout, augmentation (using inverse relations) are integral to achieve good performance, and Xavier initialization significantly improves the convergence rate.

References

[1] Ivana Balazevic, Carl Allen, and Timothy M. Hospedales. Tucker: Tensor factorization for knowledge graph completion. *CoRR*, abs/1901.09590, 2019.

[2] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *International Conference on Machine Learning*, ICML'11, page 809–816, Madison, WI, USA, 2011. Omnipress.

[3] Bishan Yang, Wen tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases, 2015.

[4] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings, 2018.

[5] Ivana Balažević, Carl Allen, and Timothy M. Hospedales. Hypernetwork knowledge graph embeddings. *Lecture Notes in Computer Science*, page 553–565, 2019.