TuckER: Tensor Factorization for Knowledge Graph Completion

Anonymous Authors[1]

Abstract

Knowledge graphs are structured representations of real world facts. They typically contain only a small subset of all possible facts. Link prediction is a task of inferring missing facts based on existing ones. We propose TuckER, a relatively simple but powerful linear model based on Tucker decomposition of the binary tensor representation of knowledge graph triples. Due to the presence of the core tensor, parameters are shared between relations, enabling multi-task learning. TuckER outperforms previous state-of-the-art models across standard link prediction datasets.

1. Introduction

Vast amounts of information available in the world can be represented succinctly as *entities* and *relations* between them. *Knowledge graphs* are large, graph-structured databases which store facts in triple form (e_s, r, e_o), with e_s and e_o representing subject and object entities and r a relation. However, far from all available information is stored in existing knowledge graphs, which creates the need for algorithms that automatically infer missing facts. Knowledge graphs can be represented by a third-order binary tensor, where each element corresponds to a triple, 1 indicating a true fact and 0 indicating the unknown (either a false or a missing fact). The task of *link prediction* is to infer which of the 0 entries in the tensor are indeed false, and which are missing but actually true.

A large number of approaches to link prediction so far have been linear, based on various methods of factorizing the third-order binary tensor (Nickel et al., 2011; Yang et al., 2015; Trouillon et al., 2016; Kazemi & Poole, 2018). Recently, state-of-the-art results have been achieved using non-linear convolutional models (Dettmers et al., 2018; Balažević et al., 2018). Despite achieving very good performance, the fundamental problem with deep, non-linear models is that they are non-transparent and poorly understood, as opposed to more mathematically principled and widely studied tensor decomposition models.

In this paper, we introduce TuckER (E stands for entities, R for relations), a simple linear model for link prediction in knowledge graphs, based on *Tucker decomposition* (Tucker, 1966) of the binary tensor of triples. Tucker decomposition factorizes a tensor into a core tensor multiplied by a matrix along each mode. In our case, rows of the matrices contain entity and relation embeddings, while entries of the core tensor determine the level of interaction between them. Due to having the core tensor, unlike simpler models, such as RESCAL, DistMult and ComplEx, where parameters for each relation are often learned separately, TuckER makes use of *multi-task learning* between different relations (Yang & Hospedales, 2017). Subject and object entity embedding matrices are assumed equivalent, i.e. we make no distinction between the embeddings of an entity depending on whether it appears as a subject or as an object in a particular triple. Our experiments show that TuckER achieves state-of-the-art results across all standard link prediction datasets.

2. Related Work

Several *linear* models for link prediction have previously been proposed. An early linear model, RESCAL (Nickel et al., 2011), optimizes a scoring function containing a bilinear product between subject and object entity vectors and a full rank matrix for each relation. RESCAL is prone to overfitting due to its large number of parameters, which increases quadratically in the embedding dimension with the number of relations in a knowledge graph. DistMult (Yang et al., 2015) is a special case of RESCAL with a diagonal matrix per relation, which reduces overfitting. However, DistMult cannot model asymmetric relations. ComplEx (Trouillon et al., 2016) extends DistMult to the complex domain. Subject and object entity embeddings for the same entity are complex conjugates, which enables ComplEx to model asymmetric relations. SimplE (Kazemi & Poole, 2018) is a model based on Canonical Polyadic (CP) decomposition (Hitchcock, 1927).

Scoring functions of all models described above and TuckER are summarized in Table 1.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

Table 1. Scoring functions of state-of-the-art link prediction models, the dimensionality of their relation parameters, and significant terms of their space complexity. d_e and d_r are the dimensionalities of entity and relation embeddings, while n_e and n_r denote the number of entities and relations respectively. $\overline{\mathbf{e}}_o \in \mathbb{C}^{d_e}$ is the complex conjugate of $\mathbf{e}_o$, $\mathbf{h}_{e_s}, \mathbf{t}_{e_s} \in \mathbb{R}^{d_e}$ are the head and tail entity embedding of entity e_s, and $\mathbf{w}_{r^{-1}} \in \mathbb{R}^{d_r}$ is the embedding of relation r^{-1} which is the inverse of relation r. $\langle \cdot \rangle$ denotes the dot product and $\times_n$ denotes the tensor product along the n-th mode, f is a non-linear function, and $\mathcal{W} \in \mathbb{R}^{d_e \times d_e \times d_r}$ is the core tensor of a Tucker decomposition.

Model	Scoring Function	Relation Parameters	Space Complexity
RESCAL (Nickel et al., 2011)	$\mathbf{e}_s^\top \mathbf{W}_r \mathbf{e}_o$	$\mathbf{W}_r \in \mathbb{R}^{{d_e}^2}$	$\mathcal{O}(n_e d_e + n_r d_r^2)$
DistMult (Yang et al., 2015)	$\langle \mathbf{e}_s, \mathbf{w}_r, \mathbf{e}_o \rangle$	$\mathbf{w}_r \in \mathbb{R}^{d_e}$	$\mathcal{O}(n_e d_e + n_r d_e)$
ComplEx (Trouillon et al., 2016)	$\text{Re}(\langle \mathbf{e}_s, \mathbf{w}_r, \overline{\mathbf{e}}_o \rangle)$	$\mathbf{w}_r \in \mathbb{C}^{d_e}$	$\mathcal{O}(n_e d_e + n_r d_e)$
SimplE (Kazemi & Poole, 2018)	$\frac{1}{2}(\langle \mathbf{h}_{e_s}, \mathbf{w}_r, \mathbf{t}_{e_o} \rangle + \langle \mathbf{h}_{e_o}, \mathbf{w}_{r^{-1}}, \mathbf{t}_{e_s} \rangle)$	$\mathbf{w}_r \in \mathbb{R}^{d_e}$	$\mathcal{O}(n_e d_e + n_r d_e)$
TuckER (ours)	$\mathcal{W} \times_1 \mathbf{e}_s \times_2 \mathbf{w}_r \times_3 \mathbf{e}_o$	$\mathbf{w}_r \in \mathbb{R}^{d_r}$	$\mathcal{O}(n_e d_e + n_r d_r)$

3. Background

Let $\mathcal{E}$ denote the set of all entities and $\mathcal{R}$ the set of all relations present in a knowledge graph. A triple is represented as (e_s, r, e_o), with $e_s, e_o \in \mathcal{E}$ denoting subject and object entities respectively and $r \in \mathcal{R}$ the relation between them.

3.1. Link Prediction

In link prediction, we are given a subset of all true triples and the aim is to learn a *scoring function* ϕ that assigns a score $s = \phi(e_s, r, e_o) \in \mathbb{R}$ which indicates whether a triple is true, with the ultimate goal of being able to correctly score all missing triples. The scoring function is either a specific form of tensor factorization in the case of linear models or a more complex (deep) neural network architecture for non-linear models. Typically, a positive score for a particular triple indicates a true fact predicted by the model, while a negative score indicates a false one.

3.2. Tucker Decomposition

Tucker decomposition, named after Ledyard R. Tucker (Tucker, 1964), decomposes a tensor into a set of matrices and a smaller core tensor. In a three-mode case, given the original tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$, Tucker decomposition outputs a tensor $\mathcal{Z} \in \mathbb{R}^{P \times Q \times R}$ and three matrices $\mathbf{A} \in \mathbb{R}^{I \times P}$, $\mathbf{B} \in \mathbb{R}^{J \times Q}$, $\mathbf{C} \in \mathbb{R}^{K \times R}$:

$$\mathcal{X} \approx \mathcal{Z} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}, \tag{1}$$

with $\times_n$ indicating the tensor product along the n-th mode. Elements of the *core tensor* $\mathcal{Z}$ show the level of interaction between the different components. Typically, P, Q, R are smaller than I, J, K respectively, so $\mathcal{Z}$ can be thought of as a compressed version of $\mathcal{X}$ (Kolda & Bader, 2009).

4. Tucker Decomposition for Link Prediction

We propose a model that uses Tucker decomposition for link prediction on the binary tensor representation of a knowledge graph, with entity embedding matrix $\mathbf{E}$ that is equivalent for subject and object entities, i.e. $\mathbf{E} = \mathbf{A} = \mathbf{C} \in \mathbb{R}^{n_e \times d_e}$ and relation embedding matrix $\mathbf{R} = \mathbf{B} \in \mathbb{R}^{n_r \times d_r}$, where n_e and n_r represent the number of entities and relations and d_e and d_r the dimensionality of entity and relation embedding vectors.

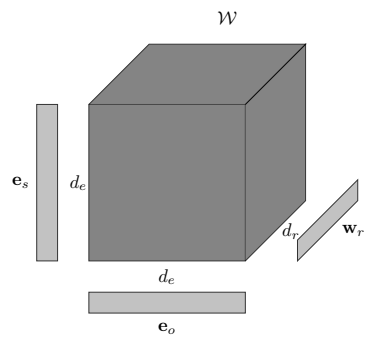

Figure 1. Visualization of the TuckER architecture.

We define the scoring function for TuckER as:

$$\phi(e_s, r, e_o) = \mathcal{W} \times_1 \mathbf{e}_s \times_2 \mathbf{w}_r \times_3 \mathbf{e}_o \tag{2}$$

where $\mathbf{e}_s, \mathbf{e}_o \in \mathbb{R}^{d_e}$ are the rows of $\mathbf{E}$ representing the subject and object entity embedding vectors, $\mathbf{w}_r \in \mathbb{R}^{d_r}$ the rows of $\mathbf{R}$ representing the relation embedding vector and $\mathcal{W} \in \mathbb{R}^{d_e \times d_r \times d_e}$ is the core tensor. We apply logistic sigmoid to each score $\phi(e_s, r, e_o)$ to obtain the predicted probability p of a triple being true. Visualization of the TuckER model architecture can be seen in Figure 1. The number of parameters of TuckER increases *linearly* with respect to entity and relation embedding dimensionality d_e and d_r, as the number of entities and relations increases, since the number of parameters of $\mathcal{W}$ depends only on the entity and relation embedding dimensionality and not on the number of entities or relations. By having the core tensor $\mathcal{W}$, unlike simpler models such as DistMult, ComplEx and SimplE, TuckER does not encode all the learned knowledge into the embeddings; some is stored in the core tensor and shared between all entities and relations through multi-task learning (Yang & Hospedales, 2017).

4.1. Training

Following the training procedure introduced by Dettmers et al. (2018), we use *1-N scoring*, i.e. we simultaneously score a pair (e_s, r) with all entities $e_o \in \mathcal{E}$, in contrast to *1-1 scoring*, where individual triples (e_s, r, e_o) are trained one at a time. We assume that a knowledge graph is only locally complete by including only the non-existing triples $(e_s, r, \cdot)$ and $(\cdot, r, e_o)$ of the observed pairs (e_s, r) and (r, e_o) as negative samples and all observed triples as positive samples. We train our model to minimize the Bernoulli negative log-likelihood loss function. A component of the loss for one subject entity and all the object entities is defined as:

$$L = -\frac{1}{n_e} \sum_{i=1}^{n_e} (\mathbf{y}^{(i)} \log(\mathbf{p}^{(i)}) + (1 - \mathbf{y}^{(i)}) \log(1 - \mathbf{p}^{(i)})), \tag{3}$$

where $\mathbf{p} \in \mathbb{R}^{n_e}$ is the vector of predicted probabilities and $\mathbf{y} \in \mathbb{R}^{n_e}$ is the binary label vector.

5. Experiments and Results

5.1. Datasets

We evaluate TuckER using standard link prediction datasets. FB15k (Bordes et al., 2013) is a subset of Freebase, a large database of real world facts. FB15k-237 (Toutanova et al., 2015) was created from FB15k by removing the inverse of many relations that are present in the training set from validation and test sets. WN18 (Bordes et al., 2013) is a subset of WordNet, containing lexical relations between words. WN18RR (Dettmers et al., 2018) is a subset of WN18, created by removing the inverse relations.

5.2. Implementation and Experiments

We implement TuckER in PyTorch (Paszke et al., 2017) and make our code available on Github [1]. We choose all hyper-parameters by random search based on validation set performance. For FB15k and FB15k-237, we set entity and relation embedding dimensionality to $d_e = d_r = 200$. For WN18 and WN18RR, which both contain a significantly smaller number of relations relative to the number of entities as well as a small number of relations compared to FB15k and FB15k-237, we set $d_e = 200$ and $d_r = 30$. We use batch normalization (Ioffe & Szegedy, 2015) and dropout (Srivastava et al., 2014) to speed up training. We choose the learning rate from $\{0.01, 0.005, 0.003, 0.001, 0.0005\}$ and learning rate decay from $\{1, 0.995, 0.99\}$. We find the following combinations of learning rate and learning rate decay to give the best results: $(0.003, 0.99)$ for FB15k, $(0.0005, 1.0)$ for FB15k-237, $(0.005, 0.995)$ for WN18 and $(0.01, 1.0)$ for WN18RR. We train the model using Adam (Kingma & Ba, 2015) with the batch size 128.

[1] http://anonymized/

We evaluate each triple from the test set as in (Bordes et al., 2013): for a given triple, we generate $2n_e$ test triples by keeping the subject entity e_s and relation r fixed and replacing the object entity e_o with all possible entities $\mathcal{E}$ and vice versa. We then rank the scores obtained. We use the filtered setting, i.e. we remove all true triples apart from the currently observed test triple. For evaluation, we use the evaluation metrics used across the link prediction literature: mean reciprocal rank (MRR) and hits@k, $k \in \{1, 3, 10\}$. Mean reciprocal rank is the average of the inverse of a mean rank assigned to the true triple over all n_e generated triples. Hits@k measures the percentage of times the true triple is ranked in the top k of the n_e generated triples.

5.3. Link Prediction Results

Link prediction results on all datasets are shown in Tables 2 and 3. Overall, TuckER outperforms previous state-of-the-art models on all metrics across all datasets (apart from hits@10 on WN18). Results achieved by TuckER are not only better than those of other linear models, such as DistMult, ComplEx and SimplE, but also better than those of many more complex deep neural network and reinforcement learning architectures, e.g. MINERVA, ConvE and HypER, demonstrating the expressive power of linear models.

Even though at entity embedding dimensionality $d_e = 200$ and relation embedding dimensionality $d_r = 30$ on WN18RR TuckER has fewer parameters ($\sim$9.4 million) than ComplEx and SimplE ($\sim$16.4 million), it consistently obtains better results than any of those models. We believe this is achieved by exploiting knowledge sharing between relations through the core tensor. We find that lower dropout values $(0.1, 0.2)$ are required for datasets with a higher number of training triples per relation and thus less risk of overfitting (WN18 and WN18RR) and higher dropout values $(0.3, 0.4, 0.5)$ required for FB15k and FB15k-237. We further note that TuckER improves the results of all other linear models by a larger margin on datasets with a large number of relations (e.g. $+14\%$ improvement on FB15k results over ComplEx, $+8\%$ improvement over SimplE on the toughest hits@1 metric), which supports our belief that TuckER makes use of the parameters shared between similar relations to improve predictions by multi-task learning.

5.4. Influence of Decomposition Rank

The presence of core tensor which allows for knowledge sharing between relations suggests that TuckER should need a lower number of parameters for obtaining good results than ComplEx or SimplE. To test this, we re-implement ComplEx and SimplE with 1-N scoring, batch normalization and dropout for fair comparison, perform random search to choose best hyper-parameters and train all three models on FB15k-237 with embedding sizes $d_e = d_r \in$

Table 2. Link prediction results on WN18RR and FB15k-237. We report results for ComplEx-N3 (Lacroix et al., 2018) at $d_e = 115$ for WN18RR and $d_e = 400$ for FB15k-237 to ensure comparability with TuckER in terms of the overall number of parameters (original paper reports results at $d_e = 2000$). The RotatE (Sun et al., 2019) results are reported without their self-adversarial negative sampling (see Appendix H in the original paper) for fair comparison, given that it improves the results by $\sim 4\%$ and it is not specific to that model only.

		WN18RR				FB15k-237			
	Linear	MRR	Hits@10	Hits@3	Hits@1	MRR	Hits@10	Hits@3	Hits@1
DistMult (Yang et al., 2015)	yes	.430	.490	.440	.390	.241	.419	.263	.155
ComplEx (Trouillon et al., 2016)	yes	.440	.510	.460	.410	.247	.428	.275	.158
Neural LP (Yang et al., 2017)	no	–	–	–	–	.250	.408	–	–
R-GCN (Schlichtkrull et al., 2018)	no	–	–	–	–	.248	.417	.264	.151
MINERVA (Das et al., 2018)	no	–	–	–	–	–	.456	–	–
ConvE (Dettmers et al., 2018)	no	.430	.520	.440	.400	.325	.501	.356	.237
HypER (Balažević et al., 2018)	no	.465	.522	.477	.436	.341	.520	.376	.252
ComplEx-N3 (Lacroix et al., 2018)	yes	.462	.523	.476	.430	.354	.543	.389	.262
M-Walk (Shen et al., 2018)	no	.437	–	.445	.414	–	–	–	–
RotatE (Sun et al., 2019)	no	–	–	–	–	.297	.480	.328	.205
TuckER (ours)	yes	**.470**	**.526**	**.482**	**.443**	**.358**	**.544**	**.394**	**.266**

Table 3. Link prediction results on WN18 and FB15k.

		WN18				FB15k			
	Linear	MRR	Hits@10	Hits@3	Hits@1	MRR	Hits@10	Hits@3	Hits@1
TransE (Bordes et al., 2013)	no	–	.892	–	–	–	.471	–	–
DistMult (Yang et al., 2015)	yes	.822	.936	.914	.728	.654	.824	.733	.546
ComplEx (Trouillon et al., 2016)	yes	.941	.947	.936	.936	.692	.840	.759	.599
ANALOGY (Liu et al., 2017)	yes	.942	.947	.944	.939	.725	.854	.785	.646
Neural LP (Yang et al., 2017)	no	.940	.945	–	–	.760	.837	–	–
R-GCN (Schlichtkrull et al., 2018)	no	.819	**.964**	.929	.697	.696	.842	.760	.601
TorusE (Ebisu & Ichise, 2018)	no	.947	.954	.950	.943	.733	.832	.771	.674
ConvE (Dettmers et al., 2018)	no	.943	.956	.946	.935	.657	.831	.723	.558
HypER (Balažević et al., 2018)	no	.951	958	**.955**	.947	.790	.885	.829	.734
SimplE (Kazemi & Poole, 2018)	yes	.942	.947	.944	.939	.727	.838	.773	.660
TuckER (ours)	yes	**.953**	.958	**.955**	**.949**	**.795**	**.892**	**.833**	**.741**

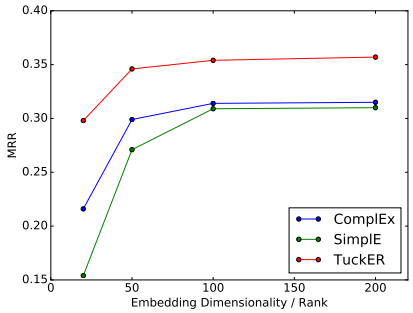

Figure 2. MRR for ComplEx, SimplE and TuckER for different embeddings sizes on FB15k-237.

$\{20, 50, 100, 200\}$. Figure 2 shows the obtained MRR on the test set for each model. It is important to note that at embedding dimensionalities 20, 50 and 100, TuckER has fewer parameters than ComplEx and SimplE (e.g. ComplEx and SimplE have $\sim$3 million and TuckER has $\sim$2.5 million parameters for embedding dimensionality 100).

We can see that the difference between the MRRs of ComplEx, SimplE and TuckER is approximately constant for embedding sizes 100 and 200. However, for lower embedding sizes, the difference between MRRs increases e.g. by 4.2% for embedding size 20 for ComplEx and by 9.9% for embedding size 20 for SimplE. At embedding size 20 ($\sim$300k parameters), the performance of TuckER is almost as good as the performance of ComplEx and SimplE at embedding size 200 ($\sim$6 million parameters), which supports our initial assumption.

6. Conclusion

In this work, we introduce TuckER, a relatively simple yet highly flexible linear model for link prediction in knowledge graphs based on the Tucker decomposition of a binary tensor of training set triples, which achieves state-of-the-art results on standard link prediction datasets. TuckER's number of parameters grows linearly with respect to embedding dimension as the number of entities or relations in a knowledge graph increases, which makes it easily scalable to large knowledge graphs. Future work might include exploring how to incorporate background knowledge on individual relation properties into the existing model.

References

Balažević, I., Allen, C., and Hospedales, T. M. Hypernetwork Knowledge Graph Embeddings. *arXiv preprint arXiv:1808.07018*, 2018.

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O. Translating Embeddings for Modeling Multi-relational Data. In *Advances in Neural Information Processing Systems*, 2013.

Das, R., Dhuliawala, S., Zaheer, M., Vilnis, L., Durugkar, I., Krishnamurthy, A., Smola, A., and McCallum, A. Go for a Walk and Arrive at the Answer: Reasoning over Paths in Knowledge Bases Using Reinforcement Learning. In *International Conference on Learning Representations*, 2018.

Dettmers, T., Minervini, P., Stenetorp, P., and Riedel, S. Convolutional 2D Knowledge Graph Embeddings. In *Association for the Advancement of Artificial Intelligence*, 2018.

Ebisu, T. and Ichise, R. TorusE: Knowledge Graph Embedding on a Lie Group. In *Association for the Advancement of Artificial Intelligence*, 2018.

Hitchcock, F. L. The Expression of a Tensor or a Polyadic as a Sum of Products. *Journal of Mathematics and Physics*, 6(1-4):164–189, 1927.

Ioffe, S. and Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *International Conference on Machine Learning*, 2015.

Kazemi, S. M. and Poole, D. SimplE Embedding for Link Prediction in Knowledge Graphs. In *Advances in Neural Information Processing Systems*, 2018.

Kingma, D. P. and Ba, J. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*, 2015.

Kolda, T. G. and Bader, B. W. Tensor Decompositions and Applications. *SIAM review*, 51(3):455–500, 2009.

Lacroix, T., Usunier, N., and Obozinski, G. Canonical Tensor Decomposition for Knowledge Base Completion. In *International Conference on Machine Learning*, 2018.

Liu, H., Wu, Y., and Yang, Y. Analogical Inference for Multi-relational Embeddings. In *International Conference on Machine Learning*, 2017.

Nickel, M., Tresp, V., and Kriegel, H.-P. A Three-Way Model for Collective Learning on Multi-Relational Data. In *International Conference on Machine Learning*, 2011.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic Differentiation in PyTorch. In *NIPS-W*, 2017.

Schlichtkrull, M., Kipf, T. N., Bloem, P., van den Berg, R., Titov, I., and Welling, M. Modeling Relational Data with Graph Convolutional Networks. In *European Semantic Web Conference*, 2018.

Shen, Y., Chen, J., Huang, P.-S., Guo, Y., and Gao, J. M-Walk: Learning to Walk over Graphs using Monte Carlo Tree Search. In *Advances in Neural Information Processing Systems*, 2018.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Sun, Z., Deng, Z.-H., Nie, J.-Y., and Tang, J. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. In *International Conference on Learning Representations*, 2019.

Toutanova, K., Chen, D., Pantel, P., Poon, H., Choudhury, P., and Gamon, M. Representing Text for Joint Embedding of Text and Knowledge Bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015.

Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., and Bouchard, G. Complex Embeddings for Simple Link Prediction. In *International Conference on Machine Learning*, 2016.

Tucker, L. R. The Extension of Factor Analysis to Three-Dimensional Matrices. *Contributions to Mathematical Psychology*, 110119, 1964.

Tucker, L. R. Some Mathematical Notes on Three-Mode Factor Analysis. *Psychometrika*, 31(3):279–311, 1966.

Yang, B., Yih, W.-t., He, X., Gao, J., and Deng, L. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *International Conference on Learning Representations*, 2015.

Yang, F., Yang, Z., and Cohen, W. W. Differentiable Learning of Logical Rules for Knowledge Base Reasoning. In *Advances in Neural Information Processing Systems*, 2017.

Yang, Y. and Hospedales, T. Deep Multi-task Representation Learning: A Tensor Factorisation Approach. In *International Conference on Learning Representations*, 2017.