ROTATE

KNOWLEDGE GRAPH EMBEDDING BY RELATIONAL ROTATION IN COMPLEX SPACE

Authors

ROTATE: KNOWLEDGE GRAPH EMBEDDING BY RELATIONAL ROTATION IN COMPLEX SPACE

Zhiqing Sun [1]*, Zhi-Hong Deng[1], Jian-Yun Nie[3], Jian Tang[2,4,5]
[1]Peking University, China
[2]Mila-Quebec Institute for Learning Algorithms, Canada
[3]Université de Montréal, Canada
[4]HEC Montréal, Canada
[5]CIFAR AI Research Chair

- **Zhiqing Sun:** PKU; University of Montreal (UdeM); Montreal Institute for Learning Algorithms (MILA); NLC Group at Microsoft Research Asia (MSRA); Google Brain Team
- Papers:
  - Zhiqing Sun, Jian Tang, Pan Du, Zhi-Hong Deng, Jian-Yun Nie. DivGraphPointer: A Graph Pointer Network for Extracting Diverse Keyphrases. SIGIR, 2019.
  - Zhiqing Sun, Zhi-Hong Deng. Unsupervised Neural Word Segmentation for Chinese via Segmental Language Modeling. EMNLP, 2018.

Authors

Dr. Jian Tang

Assistant Professor,
HEC Montreal
Montreal Institute for Learning Alogorithms (MILA)

Montreal, Canada

Twitter

Github

Weibo

Google Scholar

Research Interests

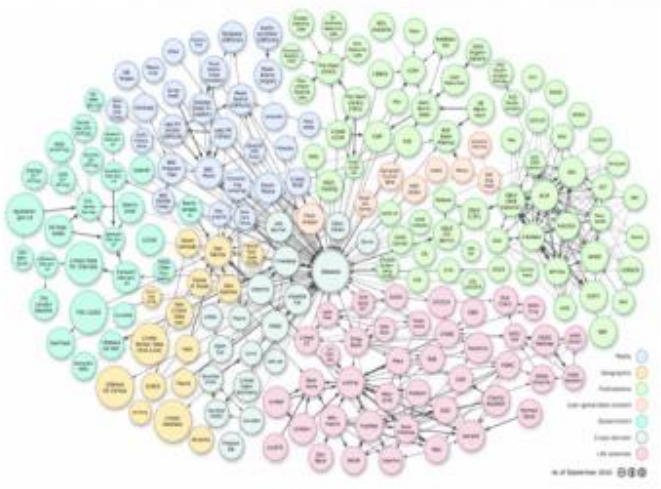
- Deep learning, deep generative models, reinforcement learning
- Graph representation learning, Graph Neural Networks
- Natural language understanding and reasoning, Knowledge graphs
- Drug discovery
- Recommender systems

Recent Papers

- New!! Meng Qu, Yoshua Bengio, and Jian Tang. "GMNN: Graph Markov Neural Networks". To appear at the 36th International Conference on Machine Learning (ICML'19), Long Beach, California, United States. codes
- New!! Zhiqing Sun, Jian Tang, Pan Du, Zhi-Hong Deng and Jian-Yun Nie. "DivGraphPointer: A Graph Pointer Network for Extracting Diverse Keyphrases." To appear at the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, July 21-25, 2019, Paris, France.
- New!! Zhaocheng Zhu, Shizhen Xu, Meng Qu, Jian Tang. "GraphVite: A High-Performance CPU-GPU Hybrid System for Node Embedding". To appear at the Web Conference 2019 (formerly known as WWW'2019), San Francisco, CA, USA, May 13-17, 2019,
- New!! Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, Jian Tang. "RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space." To appear at the Seventh International Conference on Learning Representations (ICLR'19), New Orleans, USA. codes slides

Knowledge Graphs

- Knowledge graphs are **heterogeneous** graphs
  - Multiple types of relations
- A set of facts represented as triplets
  - (head entity, relation, tail entity)
- A variety of applications
  - Question answering
  - Search
  - Recommender Systems
  - Natural language understanding
  - …



Google Knowledge Graph

Microsoft 悟 SATORI

NELL: Never-Ending Language Learning

Freebase

DBpedia

yago select knowledge

OpenIE (Reverb, OLLIE)

Related Work on Knowledge Graph Embedding

- Representing entities as **embeddings**
- Representing relations as **embeddings** or **matrices**

Model	Score Function	
SE (Bordes et al., 2011)	$-\|\|\boldsymbol{W}_{r,1}\mathbf{h}-\boldsymbol{W}_{r,2}\mathbf{t}\|\|$	$\mathbf{h},\mathbf{t}\in\mathbb{R}^k, \boldsymbol{W}_{r,\cdot}\in\mathbb{R}^{k\times k}$
TransE (Bordes et al., 2013)	$-\|\|\mathbf{h}+\mathbf{r}-\mathbf{t}\|\|$	$\mathbf{h},\mathbf{r},\mathbf{t}\in\mathbb{R}^k$
TransX	$-\|\|g_{r,1}(\mathbf{h})+\mathbf{r}-g_{r,2}(\mathbf{t})\|\|$	$\mathbf{h},\mathbf{r},\mathbf{t}\in\mathbb{R}^k$
DistMult (Yang et al., 2014)	$\langle\mathbf{r},\mathbf{h},\mathbf{t}\rangle$	$\mathbf{h},\mathbf{r},\mathbf{t}\in\mathbb{R}^k$
ComplEx (Trouillon et al., 2016)	$\mathrm{Re}(\langle\mathbf{r},\mathbf{h},\overline{\mathbf{t}}\rangle)$	$\mathbf{h},\mathbf{r},\mathbf{t}\in\mathbb{C}^k$
HolE (Nickel et al., 2016)	$\langle\mathbf{r},\mathbf{h}\otimes\mathbf{t}\rangle$	$\mathbf{h},\mathbf{r},\mathbf{t}\in\mathbb{R}^k$
ConvE (Dettmers et al., 2017)	$\langle\sigma(\mathrm{vec}(\sigma([\overline{\mathbf{r}},\overline{\mathbf{h}}]*\boldsymbol{\Omega}))\boldsymbol{W}),\mathbf{t}\rangle$	$\mathbf{h},\mathbf{r},\mathbf{t}\in\mathbb{R}^k$
RotatE	$-\|\|\mathbf{h}\circ\mathbf{r}-\mathbf{t}\|\|^1$	$\mathbf{h},\mathbf{r},\mathbf{t}\in\mathbb{C}^k, \|r_i\|=1$

Task: Knowledge Graph Completion

- A fundamental task: **predicting missing links**
- The Key Idea: model and infer the **relation patterns** in knowledge graphs according to observed knowledge facts.
  - The relationship between entities
- Example:

Barack_Obama **BornIn** United_States

Barack_Obama **Nationality** American

Parents of Parents are Grandparents

Relation Patterns

- **Symmetric/Antisymmetric** Relations
  - Symmetric: e.g., Marriage
  - Antisymmetric: e.g., Filiation
- Formally:

r is **Symmetric**: $r(x, y) \Rightarrow r(y, x)$ if $\forall\, x, y$

r is **Antisymmetric**: $r(x, y) \Rightarrow \neg r(y, x)$ if $\forall\, x, y$

Relation Patterns

- **Inverse** Relations
  - Hypernym and hyponym
  - Husband and wife
- Formally:

r_1 **is inverse to relation** r_2**:** $r_2(x, y) \Rightarrow r_1(y, x)$ if $\forall\ x, y$

Relation Patterns

- **Composition** Relations
  - My mother's husband is my father
- Formally:

r_1 is a **composition** of relation r_2 and relation r_3: $r_2(x, y) \wedge r_3(y, z) \Rightarrow r_1(x, z)$ if $\forall\, x, y, z$

Abilities in Inferring the Relation Patterns

- None of existing methods are able to model and infer all the three types of relation patterns

Model	Score Function	Symmetry	Antisymmetry	Inversion	Composition
SE	$-\|\|\boldsymbol{W}_{r,1}\mathbf{h}-\boldsymbol{W}_{r,2}\mathbf{t}\|\|$	✗	✗	✗	✗
TransE	$-\|\|\mathbf{h}+\mathbf{r}-\mathbf{t}\|\|$	✗	✓	✓	✓
TransX	$-\|\|g_{r,1}(\mathbf{h})+\mathbf{r}-g_{r,2}(\mathbf{t})\|\|$	✓	✓	✗	✗
DistMult	$\langle\mathbf{h},\mathbf{r},\mathbf{t}\rangle$	✓	✗	✗	✗
ComplEx	$\mathrm{Re}(\langle\mathbf{h},\mathbf{r},\bar{\mathbf{t}}\rangle)$	✓	✓	✓	✗
RotatE	$-\|\|\mathbf{h}\circ\mathbf{r}-\mathbf{t}\|\|$	✓	✓	✓	✓

Contributions

- A new knowledge graph embedding model RotatE
  - Each relation as a elementwise rotation from the source entity to the target entity in the complex vector space
- RotatE is able to model and infer all the three types of relation patterns
- An efficient and effective negative sampling algorithm for optimizing RotatE
- State-of-the-art results on all the benchmarks for link prediction on knowledge graphs

Relation as Elementwise Rotation

- Representing head and tail entities in complex vector space, i.e., $\mathbf{h}, \mathbf{t} \in \mathbb{C}^{\boldsymbol{k}}$
- Define each relation **r** as an element-wise rotation from the head entity **h** to the tail entity **t,** i.e.,

$$\mathbf{t} = \mathbf{h} \circ \boldsymbol{r}, \quad \text{where } |r_i|=1$$

- ° is the element-wise product. More specifically, we have $t_i = h_i r_i$, and

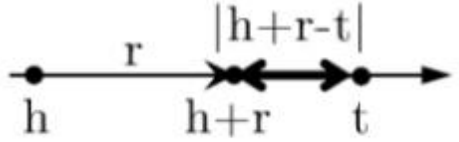
$$r_i = e^{i\theta_{r,i}},$$

- where $\theta_{r,i}$ is the phase angle of **r** in the i-th dimension.

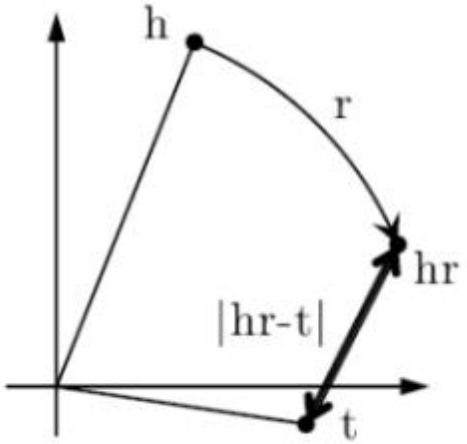
Geometric Interpretation

- Define the distance function of RotatE as

$$d_r(\boldsymbol{h}, \boldsymbol{t}) = ||\mathbf{h} \circ \mathbf{r} - \mathbf{t}||$$



(a) TransE models $\mathbf{r}$ as translation in real line.



(b) RotatE models $\mathbf{r}$ as rotation in complex plane.

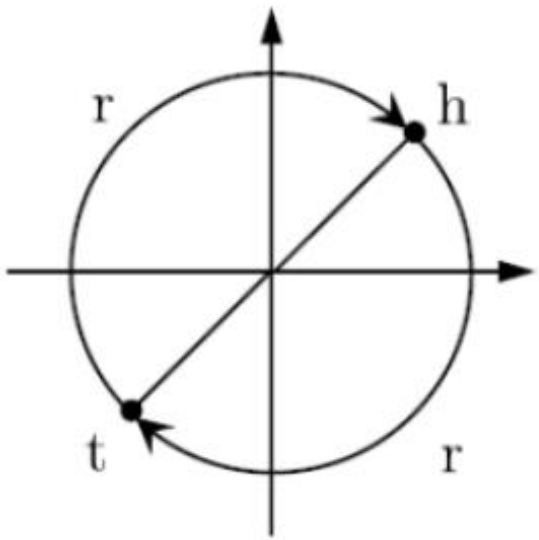
Modeling the Relation Patterns

- A relation **r** is **symmetric** if and only if $r_i = \pm 1$, i.e.,

$$\theta_{r,i} = 0 \; or \; \pi$$

- An example on the space of $\mathbb{C}$

$$r_i = -1 \text{ or } \theta_{r,i} = \pi$$



Modeling the Relation Patterns

- A relation r is **antisymmetric** if and only if $\mathbf{r}^{\circ}\ \mathbf{r} \neq \mathbf{1}$

- Two relations r_1 and r_2 are **inverse** if and only if $\mathbf{r}_2 = \bar{\mathbf{r}}_1$, i.e.,

$$\theta_{2,i} = -\theta_{1,i}$$

- A relation $\boldsymbol{r_3} = e^{i\boldsymbol{\theta_3}}$ is a **composition** of two relations $\boldsymbol{r_1} = e^{i\boldsymbol{\theta_1}}$ and $\boldsymbol{r_2} = e^{i\boldsymbol{\theta_2}}$ if only if $\boldsymbol{r_3} = \boldsymbol{r_1} \circ \boldsymbol{r_2}$, i.e.,

$$\boldsymbol{\theta_3} = \boldsymbol{\theta_1} + \boldsymbol{\theta_2}$$

Optimization

- Negative sampling loss

$$L = -\log \sigma\big(\gamma - d_r(\boldsymbol{h}, \boldsymbol{t})\big) - \sum_{i=1}^{k} \frac{1}{k} \log \sigma(d_r(\boldsymbol{h}'_i, \boldsymbol{t}'_i) - \gamma)$$

- γ is a fixed margin, σ is the sigmoid function, and $(\boldsymbol{h}'_i, \boldsymbol{r}, \boldsymbol{t}'_i)$ is the i-th negative triplet.

Self-adversarial Negative Sampling

- Traditionally, the negative samples are drawn in an uniform way
  - Inefficient as training goes on since many samples are obviously false
  - Does not provide useful information
- A self-adversarial negative sampling
  - Sample negative triplets according to the current embedding model
  - Starts from easier samples to more and more difficult samples
  - Curriculum Learning

$$p(h'_j, r, t'_j | \{(h_i, r_i, t_i)\}) = \frac{\exp \alpha f_r(\mathbf{h}'_j, \mathbf{t}'_j)}{\sum_i \exp \alpha f_r(\mathbf{h}'_i, \mathbf{t}'_i)}$$

- α is the temperature of sampling. $f_r(h'_j, t'_j)$ measures the salience of the triplet

The Final Objective

- Instead of sampling, treating the sampling probabilities as weights.

$$L = -\log \sigma(\gamma - d_r(\mathbf{h}, \mathbf{t})) - \sum_{i=1}^{n} p(h'_i, r, t'_i) \log \sigma(d_r(\mathbf{h}'_i, \mathbf{t}'_i) - \gamma)$$

Experiments: Data Sets

- **FB15K**: a subset of Freebase. The main relation types are **symmetry/antisymmetry** and **inversion** patterns.
- **WN18**: a subset of WordNet. The main relation types are **symmetry/antisymmetry** and **inversion** patterns.
- **FB15K-237**: a subset of FB15K, where inversion relations are deleted. The main relation types are **symmetry/antisymmetry** and **composition** patterns.
- **WN18RR**: a subset of WN18, where inversion relations are deleted. The main relation types are **symmetry/antisymmetry** and **composition** patterns.

Dataset	#entity	#relation	#training	#validation	#test
FB15k	14,951	1,345	483,142	50,000	59,071
WN18	40,943	18	141,442	5,000	5,000
FB15k-237	14,541	237	272,115	17,535	20,466
WN18RR	40,943	11	86,835	3,034	3,134

Compared Algorithms

- Baseline algorithms
  - TransE (Bordes et al. 2013)
  - DistMult (Yang et al. 2014)
  - ComplEx (Trouillon et al. 2016)
  - ConvE (Dettmers et al. 2017)
- Our algorithms
  - RotatE
  - pRotatE: a variant of our algorithm by constraining the modulus of the entity embeddings to be the same, i.e., $|h_i| = |t_i| = C$

Results on FB15k and WN18

- RotatE performs the best
- pRotatE performs similarly to RotatE

	FB15k					WN18				
	MR	MRR	H@1	H@3	H@10	MR	MRR	H@1	H@3	H@10
TransE [♥]	-	.463	.297	.578	.749	-	.495	.113	.888	.943
DistMult [♦]	42	.798	-	-	**.893**	655	.797	-	-	.946
HolE	-	.524	.402	.613	.739	-	.938	.930	.945	.949
ComplEx	-	.692	.599	.759	.840	-	.941	.936	.945	.947
ConvE	51	.657	.558	.723	.831	374	.943	.935	.946	.956
pRotatE	43	**.799**	**.750**	.829	.884	**254**	.947	.942	.950	.957
RotatE	**40**	.797	.746	**.830**	.884	309	**.949**	**.944**	**.952**	**.959**

Results on FB15k-237 and WN18RR

- RotatE performs the best
- RotatE performs significantly better than pRotatE
  - A lot of composition patterns on the two data sets
  - Modulus information are important for modeling the composition patterns

	FB15k-237					**WN18RR**				
	MR	MRR	H@1	H@3	H@10	MR	MRR	H@1	H@3	H@10
TransE [♥]	357	.294	-	-	.465	3384	.226	-	-	.501
DistMult	254	.241	.155	.263	.419	5110	.43	.39	.44	.49
ComplEx	339	.247	.158	.275	.428	5261	.44	.41	.46	.51
ConvE	244	.325	.237	.356	.501	4187	.43	.40	.44	.52
pRotatE	178	.328	.230	.365	.524	**2923**	.462	.417	.479	.552
RotatE	**177**	**.338**	**.241**	**.375**	**.533**	3340	**.476**	**.428**	**.492**	**.571**

Results on Countries

- A carefully designed dataset to explicitly test the capabilities for modeling the composition patterns
  - Three subtasks S1, S2, S3
  - From easy to difficult

	Countries (AUC-PR)			
	DistMult	ComplEx	ConvE	RotatE
S1	$\mathbf{1.00 \pm 0.00}$	0.97 ± 0.02	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$
S2	0.72 ± 0.12	0.57 ± 0.10	0.99 ± 0.01	$\mathbf{1.00 \pm 0.00}$
S3	0.52 ± 0.07	0.43 ± 0.07	0.86 ± 0.05	$\mathbf{0.95 \pm 0.00}$

Comparing Different Negative Sampling Techniques

- Different negative sampling techniques
  - Uniform sampling
  - KBGAN (Cai&Wang, 2017) : a generator for generating negative samples and a discriminator for training knowledge graph embeddings
  - our Self-adversarial technique

	FB15k-237		**WN18RR**		**WN18**	
	MRR	H@10	MRR	H@10	MRR	H@10
uniform	.242	.422	.186	.459	.433	.915
KBGAN (Cai & Wang, 2017)	.278	.453	.210	.479	.705	**.949**
self-adversarial	**.298**	**.475**	**.223**	**.510**	**.736**	.947

Table: The performance of different techniques with TransE

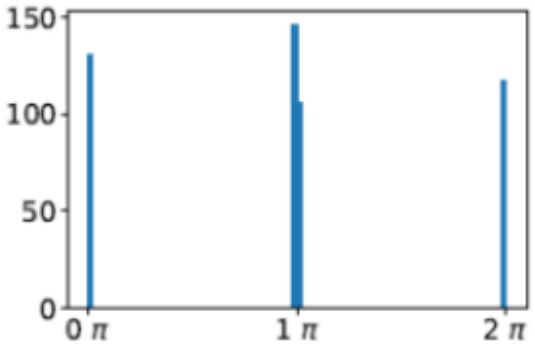
Further Experiments

- To make a fair comparison, we compare with other models, which are also trained with self-adversarial
- Similar results are observed

	FB15k		**FB15k-237**		**Countries (AUC-ROC)**		
	MRR	H@10	MRR	H@10	S1	S2	S3
TransE	.735	.871	.332	.531	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{0.96 \pm 0.00}$
ComplEx	.780	**.890**	.319	.509	$\mathbf{1.00 \pm 0.00}$	0.98 ± 0.00	0.88 ± 0.01
RotatE	**.797**	.884	**.338**	**.533**	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	0.95 ± 0.00

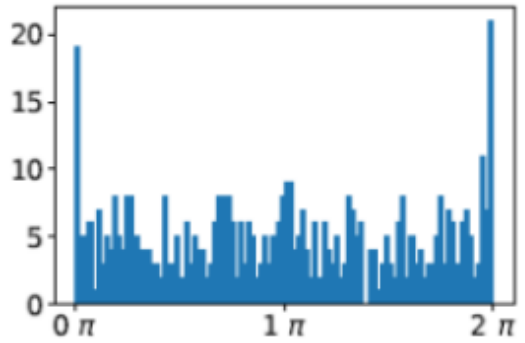
Implicit Relation Pattern Inference: Symmetric

- Ignore the positions in the relation embedding $\boldsymbol{\theta_r}$ and plot the histogram of the phrase angle of each element in the relation embedding, i.e., $\theta_{r,i}$



(a) similar_to

A **symmetric** relation

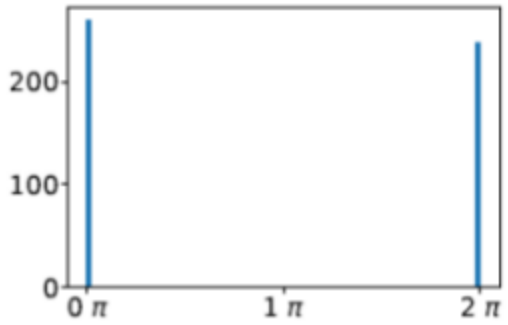


(b) hypernym

A **random** relation

Implicit Relation Pattern Inference: Inversion

- Ignore the positions in the relation embedding $\boldsymbol{\theta_r}$ and plot the histogram of $\theta_{1,i} + \theta_{2,i}$
- All the additive embedding phases are 0 or 2π, which represents that $\mathbf{r}_1 = \mathbf{r}_2^{-1}$.

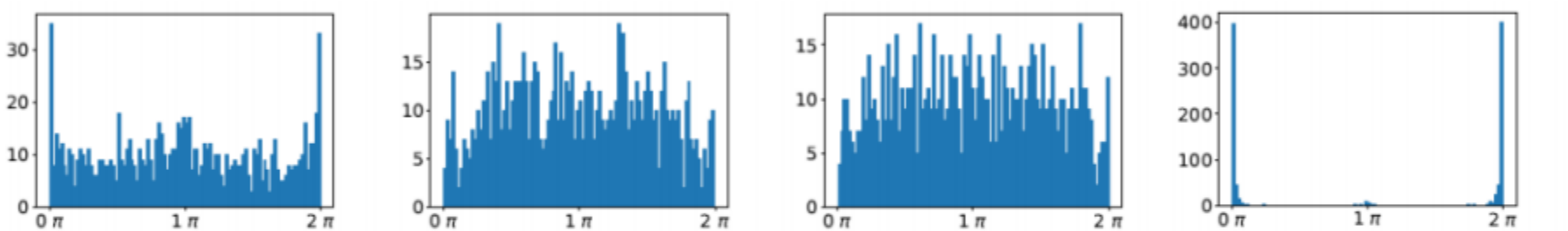


(c) hypernym ∘ hyponym

hypernym is the **inverse** relation of hyponym

Implicit Relation Pattern Inference: Composition

- Ignore the positions in the relation embedding $\boldsymbol{\theta_r}$ and plot the histogram $\theta_{1,i} + \theta_{2,i} - \theta_{3,i}$



(d) $\texttt{for}_1$ (e) $\texttt{winner}$ (f) $\texttt{for}_2$ (g) $\texttt{for}_2^{-1} \circ \texttt{winner} \circ \texttt{for}_1$

for2 is a **composition** of the relation for1 and winner.

for1: award_nominee/award_nominations./award/awrd_nomination/nominated_for

winner: award_category/winners./award/award_honor/award_winner

for2: award_category/nominees./award/award_nomination/nominated_for

Conclusion

- Modeling relation patterns is critical for knowledge base completion
  - Symmetric/Antisymmetric, Inverse, and composition
- RotatE: define each relation as a **elementwise rotation** from the head entity to the tail entity in the complex vector space
  - Capable of modeling and inferring all the three types of relation patterns
- A self-negative sampling techniques for training knowledge graph embeddings
- State-of-the-art results on all existing benchmark data sets

TuckER

Tensor Factorization for Knowledge Graph Completion

Authors

TuckER: Tensor Factorization for Knowledge Graph Completion

Ivana Balažević [1] Carl Allen [1] Timothy M. Hospedales [1]

- **Ivana Balazevic:** University of Edinburgh
- Papers:
  - Hypernetwork Knowledge Graph Embeddings. Ivana Balažević, Carl Allen, and Timothy M. Hospedales. arXiv preprint arXiv:1808.07018, 2018.

Ivana Balazevic

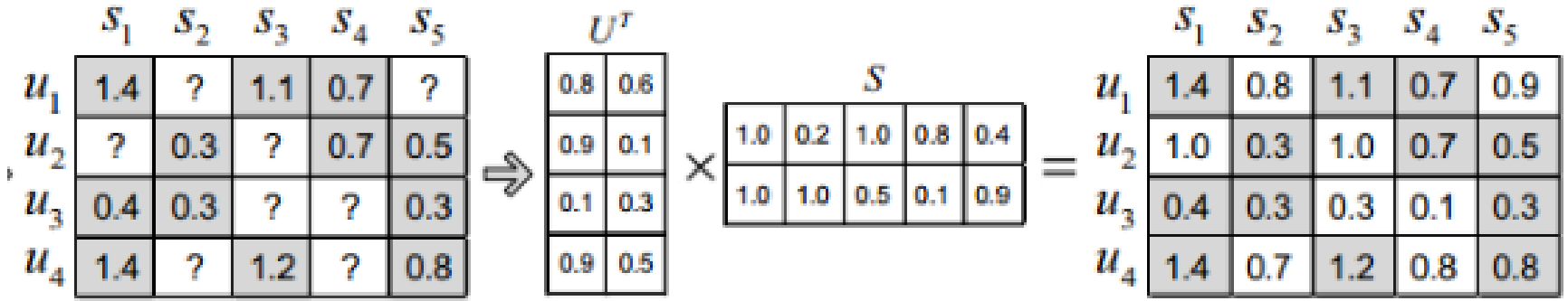
ibalazevic

Introduction

- TuckER (linear model):
  - E stands for entities, R for relations
  - Tucker decomposition (1966) of the third-order binary tensor of triples
    - Factorizes a tensor into a **core tensor** multiplied by a **matrix** along each mode.
- Contributions:
  - fully expressive
    - several relation types (symmetric, asymmetric, transitive, etc.)
  - derive a bound on the entity and relation embedding dimensionality that guarantees full expressiveness
    - lower than the bound of previous state-of-the-art models ComplEx and SimplE

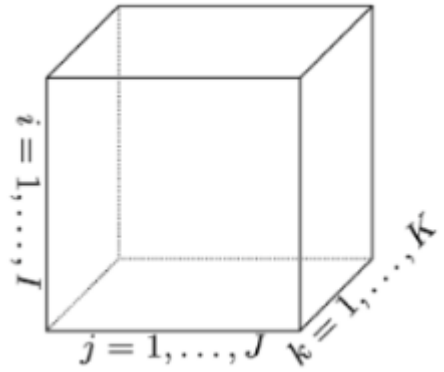
Tucker, Ledyard R. "Some mathematical notes on three-mode factor analysis." Psychometrika 31.3 (1966): 279-311.

Basic mf



- Basic MF is the most Basic decomposition method. The scoring matrix R is decomposed into the user matrix U and the item matrix S.
- Through continuous iterative training, the product of U and S is more and more close to the real matrix.
- By minimize the the difference between the predicted value and the true value, and then use the gradient descent method to calculate the U and S iteratively, when they converge, it is the decomposed matrix.

CP Decompositions



$$\mathcal{X} \in \mathbb{R}^{I \times J \times K}$$

The CP decomposition factorizes a tensor into a sum of component rank-one tensors. For example, given a third-order tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$, we wish to write it as

$$\mathcal{X} \approx \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r, \tag{3.1}$$

where R is a positive integer and $\mathbf{a}_r \in \mathbb{R}^I$, $\mathbf{b}_r \in \mathbb{R}^J$, and $\mathbf{c}_r \in \mathbb{R}^K$ for $r = 1, \ldots, R$. Elementwise, (3.1) is written as

$$x_{ijk} \approx \sum_{r=1}^{R} a_{ir}\, b_{jr}\, c_{kr} \text{ for } i = 1, \ldots, I,\ j = 1, \ldots, J,\ k = 1, \ldots, K.$$

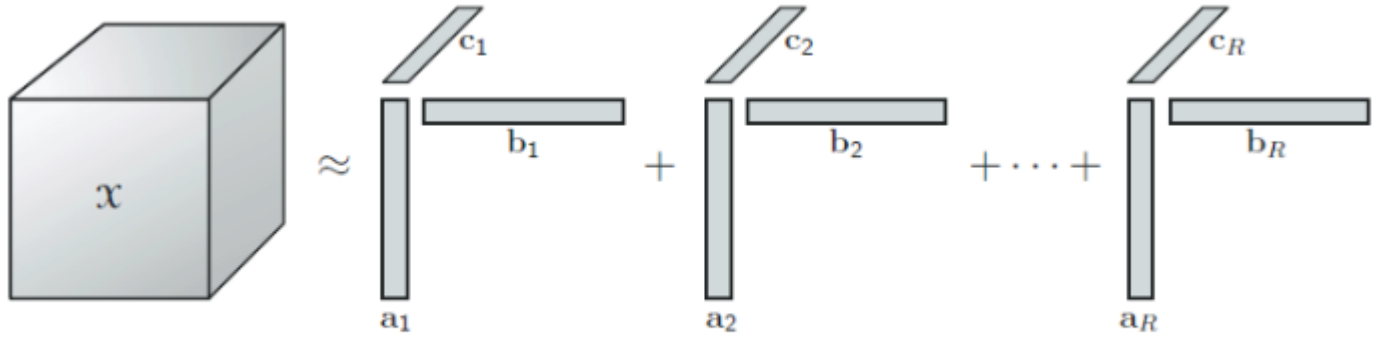


Fig. 3.1 *CP decomposition of a three-way array.*

CP Decompositions

The *factor matrices* refer to the combination of the vectors from the rank-one components, i.e., $\mathbf{A} = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_R \end{bmatrix}$ and likewise for $\mathbf{B}$ and $\mathbf{C}$. Using these definitions, (3.1) may be written in matricized form (one per mode; see section 2.4):

$$\begin{aligned} \mathbf{X}_{(1)} &\approx \mathbf{A}(\mathbf{C} \odot \mathbf{B})^{\mathsf{T}}, \\ \mathbf{X}_{(2)} &\approx \mathbf{B}(\mathbf{C} \odot \mathbf{A})^{\mathsf{T}}, \\ \mathbf{X}_{(3)} &\approx \mathbf{C}(\mathbf{B} \odot \mathbf{A})^{\mathsf{T}}. \end{aligned} \tag{3.2}$$

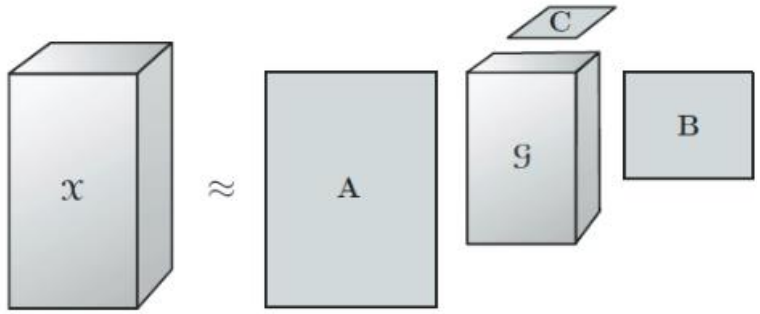
It is often useful to assume that the columns of $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ are normalized to length one with the weights absorbed into the vector $\boldsymbol{\lambda} \in \mathbb{R}^R$ so that

$$\mathcal{X} \approx [\![\boldsymbol{\lambda} ; \mathbf{A}, \mathbf{B}, \mathbf{C}]\!] \equiv \sum_{r=1}^{R} \lambda_r \, \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r. \tag{3.3}$$

We have focused on the three-way case because it is widely applicable and sufficient for many needs. For a general Nth-order tensor, $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, the CP decomposition is

$$\mathcal{X} \approx [\![\boldsymbol{\lambda} ; \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)}]\!] \equiv \sum_{r=1}^{R} \lambda_r \, \mathbf{a}_r^{(1)} \circ \mathbf{a}_r^{(2)} \circ \cdots \circ \mathbf{a}_r^{(N)},$$

Tucker decomposition



The Tucker decomposition is a form of higher-order PCA. It decomposes a tensor into a core tensor multiplied (or transformed) by a matrix along each mode. Thus, in the three-way case where $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$, we have

$$\mathcal{X} \approx \mathcal{G} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} = \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} g_{pqr}\, \mathbf{a}_p \circ \mathbf{b}_q \circ \mathbf{c}_r = [\![\mathcal{G} ; \mathbf{A}, \mathbf{B}, \mathbf{C}]\!]. \tag{4.1}$$

Here, $\mathbf{A} \in \mathbb{R}^{I \times P}$, $\mathbf{B} \in \mathbb{R}^{J \times Q}$, and $\mathbf{C} \in \mathbb{R}^{K \times R}$ are the factor matrices (which are usually orthogonal) and can be thought of as the principal components in each mode. The tensor $\mathcal{G} \in \mathbb{R}^{P \times Q \times R}$ is called the *core tensor* and its entries show the level of interaction between the different components. The last equality uses the shorthand $[\![\mathcal{G} ; \mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$ introduced in Kolda [134].

$$\mathcal{X} = \mathcal{G} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \cdots \times_N \mathbf{A}^{(N)} = [\![\mathcal{G} ; \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)}]\!]$$

$$\mathcal{X} \approx \mathcal{Z} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} = \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} z_{pqr} \mathbf{a}_p \otimes \mathbf{b}_q \otimes \mathbf{c}_r,$$

HOSVD & HOOI

procedure HOSVD($\mathcal{X}, R_1, R_2, \ldots, R_N$)
  for $n = 1, \ldots, N$ **do**
    $\mathbf{A}^{(n)} \leftarrow R_n$ leading left singular vectors of $\mathbf{X}_{(n)}$
  end for
  $\mathcal{G} \leftarrow \mathcal{X} \times_1 \mathbf{A}^{(1)\mathsf{T}} \times_2 \mathbf{A}^{(2)\mathsf{T}} \cdots \times_N \mathbf{A}^{(N)\mathsf{T}}$
  return $\mathcal{G}, \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)}$
end procedure

Fig. 4.3 *Tucker's "Method I" for computing a rank-$(R_1, R_2, \ldots, R_N)$ Tucker decomposition, later known as the HOSVD.*

procedure HOOI($\mathcal{X}, R_1, R_2, \ldots, R_N$)
  initialize $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R}$ for $n = 1, \ldots, N$ using HOSVD
  repeat
    for $n = 1, \ldots, N$ **do**
      $\mathcal{Y} \leftarrow \mathcal{X} \times_1 \mathbf{A}^{(1)\mathsf{T}} \cdots \times_{n-1} \mathbf{A}^{(n-1)\mathsf{T}} \times_{n+1} \mathbf{A}^{(n+1)\mathsf{T}} \cdots \times_N \mathbf{A}^{(N)\mathsf{T}}$
      $\mathbf{A}^{(n)} \leftarrow R_n$ leading left singular vectors of $\mathbf{Y}_{(n)}$
    end for
  until fit ceases to improve or maximum iterations exhausted
  $\mathcal{G} \leftarrow \mathcal{X} \times_1 \mathbf{A}^{(1)\mathsf{T}} \times_2 \mathbf{A}^{(2)\mathsf{T}} \cdots \times_N \mathbf{A}^{(N)\mathsf{T}}$
  return $\mathcal{G}, \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)}$
end procedure

Fig. 4.4 *ALS algorithm to compute a rank-$(R_1, R_2, \ldots, R_N)$ Tucker decomposition for an Nth-order tensor $\mathcal{X}$ of size $I_1 \times I_2 \times \cdots \times I_N$. Also known as the HOOI.*

Related Work

- RESCAL:
  - optimizes a scoring function containing a bilinear product between vector embeddings for each subject and object entity and a full rank matrix for each relation $\mathcal{X}_k \approx AR_kA^T, \text{ for } k = 1, \ldots, m$
  - very expressive and powerful model
  - 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.

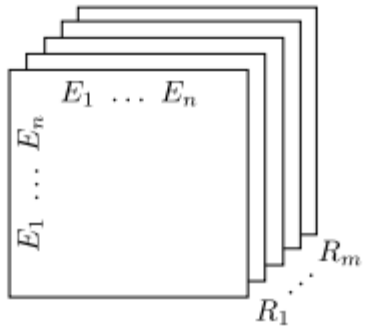
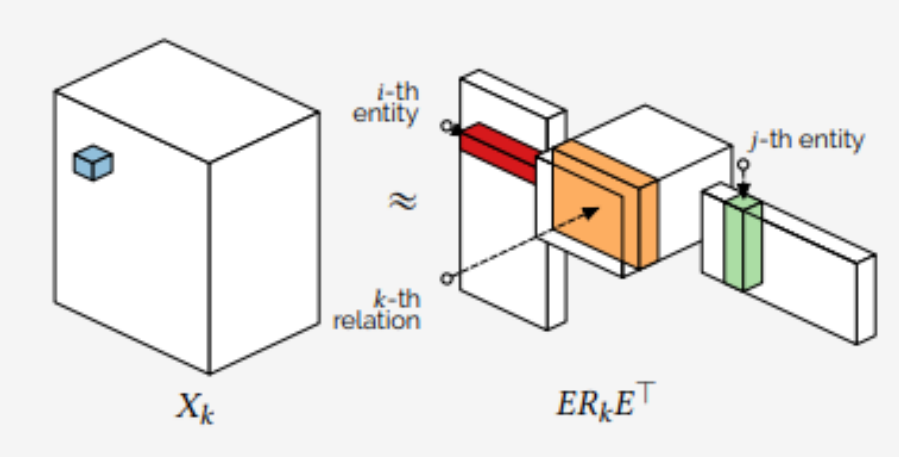


Figure 1: Tensor model for relational data. $E_1 \cdots E_n$ denote the entities, while $R_1 \cdots R_m$ denote the relations in the domain



Nickel M, Tresp V, Kriegel HP. A three-way model for collective learning on multi-relational data. In: Proc. of the 28th Int'l Conf. on Machine Learning. New York: ACM Press, 2011. 809-816.

Related Work

- DistMult:
  - a special case of RESCAL with a diagonal matrix per relation
  - The number of parameters of DistMult grows linearly with respect to the embedding dimension, reducing overfitting.
  - Given the equivalence of subject and object entity embeddings for the same entity, third-order binary tensor learned by DistMult is symmetric in the subject and object entity mode and thus DistMult cannot model asymmetric relations.
  - “We show that such rules can be effectively extracted by modeling the composition of relation embeddings, and that the embeddings learned from the bilinear objective are particularly good at capturing the compositional semantics of relations via matrix multiplication.”

Yang B, Yih WT, He X, Gao J, Deng L. Embedding entities and relations for learning and inference in knowledge bases.

Related Work

- ComplEx :
  - extends DistMult to the complex domain.
  - Subject and object entity embeddings for the same entity are no longer equivalent
  - enables ComplEx to model asymmetric relations.
- SimplE:
  - A linear model based on Canonical Polyadic (CP) decomposition
    - subject and object entity embeddings for the same entity are independent
    - DistMult is a special case of CP, where subject and object entity embeddings are equivalent
  - Alters CP to make subject and object entity embedding vectors dependent on each other, it computes the average of two terms

$$\frac{1}{2}(\langle h_{e_i}, v_r, t_{e_j} \rangle + \langle h_{e_j}, v_{r^{-1}}, t_{e_i} \rangle)$$

Yang B, Yih WT, He X, Gao J, Deng L. Embedding entities and relations for learning and inference in knowledge bases.
Kazemi, Seyed Mehran, and David Poole. "SimplE embedding for link prediction in knowledge graphs." *Advances in Neural Information Processing Systems*. 2018.

Tucker Decomposition for Link Prediction

- Uses Tucker decomposition for link prediction on the third-order binary tensor representation of a KG
- Entity embedding matrix E that is equivalent for subject and object entities
- Define the scoring function for TuckER as:

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

- $\mathbf{e}_s, \mathbf{e}_o \in \mathbb{R}^{d_e}$ are the rows of E representing the subject and object entity embedding vectors, $\mathbf{w}_r \in \mathbb{R}^{d_r}$ are the rows of R representing the relation embedding vector, $\mathcal{W} \in \mathbb{R}^{d_e \times d_r \times d_e}$ is the core tensor of Tucker decomposition
- $\times_n$ is the tensor product along the n-th mode.
- Apply logistic sigmoid to each score to obtain the predicted probability

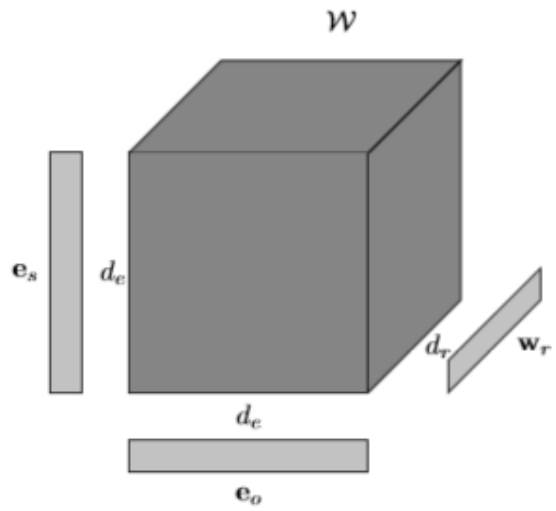


Figure 1. Visualization of the TuckER architecture.

- By having the core tensor 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.

Bound on Embedding Dimensionality

- Fully expressive :
  - for any ground truth over all entities and relations, there exist entity and relation embeddings that accurately separate the true triples from the false ones.
  - ComplEx : $d_e = d_r = n_e \cdot n_r$
  - SimplE : $d_e = d_r = min(n_e \cdot n_r, \gamma + 1)$ (γ represens the number of true facts)
  - not fully expressive: DistMult; TransE; STransE; FTransE

Theorem 1. *Given any ground truth over a set of entities $\mathcal{E}$ and relations $\mathcal{R}$, there exists a TuckER model with subject and object entity embeddings of dimensionality $d_e = n_e$ and relation embeddings of dimensionality $d_r = n_r$, where $n_e = |\mathcal{E}|$ is the number of entities and $n_r = |\mathcal{R}|$ the number of relations, that accurately represents that ground truth.*

- In practice, we expect the entity and relation embedding dimensionality needed for full reconstruction of the underlying binary tensor to be much smaller than the bound stated above.

Representing Asymmetric Relations

- Asymmetric :
  - For all subject entities that are related to corresponding object entities through r, the reciprocal necessarily does not hold
    - i.e. none of the object entities are related to the subject entities through r.

$$(e_1, r, e_2) \in \zeta \iff (e_2, r, e_1) \in \zeta'$$

- Ways to introduce asymmetry into factorization
  - To have distinct (although possibly related) embeddings for subject and object entities and a diagonal matrix (or equivalently a vector) for each relation: ComplEx and SimplE
  - For subject and object entity embeddings to be equivalent, but representing a relation as a full rank matrix: RESCAL
- TuckER introduces a novel approach:
  - By representing relations as vectors, which makes the parameter number grow linearly with the number of relations
  - By having an asymmetric relation-agnostic core tensor, which enables knowledge sharing between relations.
    - Multiplying $\mathcal{W} \in \mathbb{R}^{d_e \times d_r \times d_e}$ with $\mathbf{w}_r \in \mathbb{R}^{d_r}$ along the second mode, we obtain a full rank relation specific matrix $\mathbf{W}_r \in \mathbb{R}^{d_e \times d_e}$, which is capable of performing all possible linear transformations on the entity embeddings, i.e. rotation, reflection or stretch, and thus capable of modeling asymmetry.

Experiments: Data Sets

- **FB15K**: a subset of Freebase. The main relation types are **symmetry/antisymmetry** and **inversion** patterns.
- **WN18**: a subset of WordNet. The main relation types are **symmetry/antisymmetry** and **inversion** patterns.
- **FB15K-237**: a subset of FB15K, where inversion relations are deleted. The main relation types are **symmetry/antisymmetry** and **composition** patterns.
- **WN18RR**: a subset of WN18, where inversion relations are deleted. The main relation types are **symmetry/antisymmetry** and **composition** patterns.

Dataset	#entity	#relation	#training	#validation	#test
FB15k	14,951	1,345	483,142	50,000	59,071
WN18	40,943	18	141,442	5,000	5,000
FB15k-237	14,541	237	272,115	17,535	20,466
WN18RR	40,943	11	86,835	3,034	3,134

Implementation and Experiments

- For FB15k and FB15k-237:
  - de = dr = 200
- For WN18 and WN18RR :
  - de = 200 and dr = 30 (smaller number of relations)
- 1-N scoring
  - i.e. we simultaneously score a pair e_s and r with all entities e ∈ E, in contrast to 1-1 scoring, where individual triples (e_s, r, e_o) are trained one at a time.
- log-likelihood loss function:

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

Results

Table 2. Link prediction results on WN18RR and FB15k-237.

		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
M-Walk (Shen et al., 2018)	no	.437	–	.445	.414	–	–	–	–
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**

Influence of Embedding Dimensionality

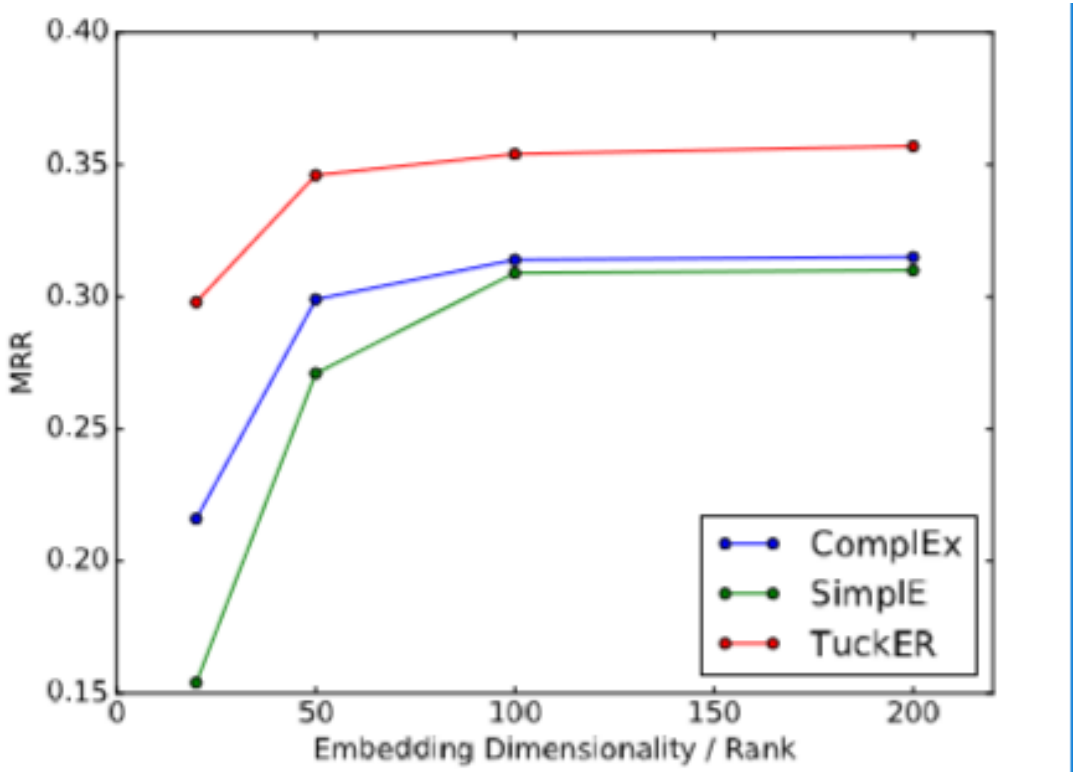


Figure 3. MRR for ComplEx, SimplE and TuckER for different embeddings sizes $d_e = d_r \in \{20, 50, 100, 200\}$ on FB15k-237.

Conclusion

- Introduce TuckER, a relatively simple yet highly flexible linear model for link prediction in knowledge graphs
- Based on the Tucker decomposition of a third-order binary tensor of training set triples
- Achieves state-of-the-art results on standard link prediction datasets.
- As well as being fully expressive, TuckER’s number of parameters grows linearly with respect to embedding dimension as the number of entities or relations in a knowledge graph increases.

END

THANK YOU