



LowFER: Low-rank Bilinear Pooling for Link Prediction

Saadullah Amin, Stalin Varanasi, Katherine Ann Dunfield, Günter Neumann

{saadullah.amin, stalin.varanasi, katherine.dunfield, neumann}@dfki.de

*Multilinguality and Language Technology Lab (MLT), German Research Center for Artificial Intelligence (DFKI), Saarbrücken, Germany
Department of Language Science and Technology, Saarland University, Saarbrücken, Germany*

Problem

- A knowledge graph (KG) is a collection of fact triples of the form $\langle \text{subject}, \text{relation}, \text{object} \rangle$
- Since all the facts are not observed, the problem of **link prediction** (LP) or **knowledge graph completion** (KGC) is the task to infer missing links.
- Specifically, given $\langle \text{subject}, \text{relation} \rangle$, the model learns to predict the missing entity.
- For example, in $\langle \text{Donald Trump}, \text{born-in}, ? \rangle$ an LP model should be able to predict New York
- Applications:
 - Extend existing KGs
 - Identifying the truthfulness of a fact
 - In multi-task learning, such as distant relation extraction
 - ...

Contributions

- We propose a simple and parameter efficient linear model by extending multi-modal factorized bilinear pooling (MFB) (Yu et al., 2017) for link prediction.
- We prove that our model is *fully expressive*, providing bounds on embedding dimensions and the factorization rank.
- We provide relationships to the family of bilinear models (RESCAL (Nickel et al., 2011), DistMult (Yang et al., 2015), ComplEx (Trouillon et al., 2016), and SimpleE (Kazemi & Poole, 2018)) Tucker decomposition (Tucker, 1966) based TuckER (Balažević et al., 2019a), generalizing them as special cases. We also show relation to 1D convolutions based HypER (Balažević et al., 2019b).
- We test our model on four real-world datasets, reaching on par or state-of-the-art performance.

LOWFER*

$E \in \mathbb{R}^{n_e \times d_e}$ (Entity Embeddings)

$R \in \mathbb{R}^{n_r \times d_r}$ (Relation Embeddings)

$U \in \mathbb{R}^{d_e \times kd_e}$ (Low-rank Matrix-1)

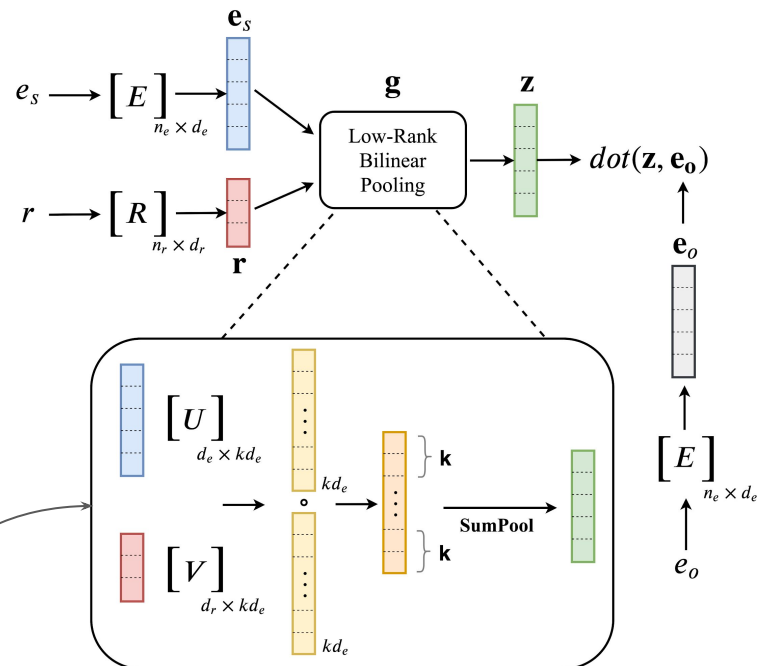
$V \in \mathbb{R}^{d_r \times kd_e}$ (Low-rank Matrix-2)

k (Factorization Rank)

$(e_s, r, e_o) = (\text{subject, relation, object})$

Introduced by [Yu et al. \(2017\)](#) as MFB.

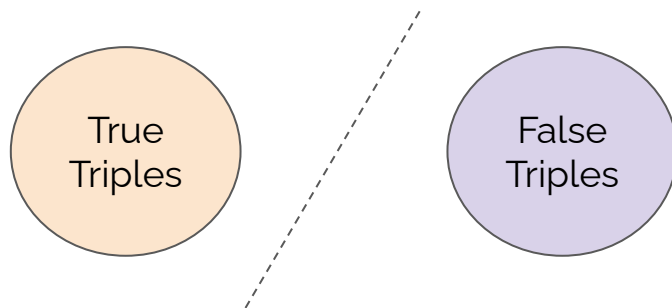
$$f(e_s, r, e_o) = (\mathbf{S}^k \text{diag}(\mathbf{U}^T \mathbf{e}_s) \mathbf{V}^T \mathbf{r})^T \mathbf{e}_o$$



* **Low-rank Factorization** trick of bilinear maps with k -sized non-overlapping summation pooling for **Entities** and **Relations** (**LowFER**)

Theoretical Analysis - I

- An important property of link prediction models is their ability to be **fully expressive**.
- Potential to separate true triples from incorrect ones.
- A full expressive model can learn all types of relations (*symmetric, anti-symmetric*, etc.)



- LowFER is fully expressive under following conditions: $d_e = n_e, d_r = n_r$ and $k = \min(d_e, d_r)$

Theoretical Analysis - II

- We show that LowFER can be seen as providing low-rank approximation to TuckER.
- Under certain conditions, it can accurately represent TuckER.
- We provide conditions under which LowFER generalizes:
 - RESCAL
 - DistMult
 - ComplEx
 - SimpleE
 - HypER (upto a non-linearity)

Experiments

- We experimented with four datasets: WN18, WN18RR, FB15k, FB15k-237
- Main results with standard evaluation metrics:

Dataset	MRR	Hits@1	Hits@3	Hits@10
WN18RR	<u>0.465</u>	0.434	<u>0.479</u>	0.526
FB15k-237	0.359	0.266	0.396	0.544
WN18	0.950	0.946	<u>0.952</u>	<u>0.958</u>
FB15k	0.824	0.782	0.852	0.897

Best results per metric **boldfaced** and second best underlined.

Key Findings

- Outperforms several more complicated modeling paradigms: 1D/2D Convolutional Networks ([Balažević et al., 2019a](#); [Dettmers et al., 2018](#)), Graph Convolutional Networks ([Schlichtkrull et al., 2018](#)), Complex Embeddings ([Trouillon et al., 2016](#)), Complex Rotation ([Sun et al., 2019](#)), Holographic Embeddings ([Trouillon et al., 2015](#)), Lie Group Embeddings ([Ebisu & Ichise, 2018](#)), Graph Walks with Reinforcement Learning and MC Tree Search ([Das et al., 2018](#); [Shen et al., 2018](#)), and Neural Logic Programming ([Yang et al., 2017](#)).
- Outperforms all the bilinear models and translational Models.
- LowFER performs extremely well at low-ranks $(1, 10)$, staying parameter efficient and performant.
- Reaches same or better performance than Tucker ([Balažević et al., 2019b](#)) with low-rank approximation and less parameters.

End of Spotlight

Problem

- A short summary of notation:

Notation	Definition
(e_s, r, e_o)	Triple of (subject, relation, object)
n_e	Number of entities
n_r	Number of relations
\mathcal{E}	Set of entities with $n_e = \mathcal{E} $
\mathcal{R}	Set of relations with $n_r = \mathcal{R} $
n	Number of triples in a knowledge graph \mathcal{KG}
d_e	Entity dimension
d_r	Relation dimension

Problem (Cont.)

- In link prediction, we learn to assign score to a triple of $\langle \text{subject}, \text{relation}, \text{object} \rangle$:

$$s = f(e_s, r, e_o)$$

- The scoring function can be seen as estimating the true binary tensor of triples:

$$\mathbf{T} \in |\mathcal{E}| \times |\mathcal{R}| \times |\mathcal{E}| : \mathbf{T}_{i,j,k} = 1 \text{ if } (e_i, r_j, e_k) \in \mathcal{KG} \text{ else } 0$$

- The scoring function can be linear or non-linear.
- Many linear models can be seen as factorizing this binary tensor.

Key Modelling Attributes in LP

- Model expressiveness
- Parameter efficiency
- Robustness to overfitting
- Fully expressive
- Model interpretability
- Parameter sharing
- Linear

Bilinear Models

Compared to a linear map, a bilinear map $\mathbf{W} \in \mathbb{R}^{m \times n}$ takes input as two vectors and produces a score i.e.

$$z = \mathbf{x}^T \mathbf{W} \mathbf{y} \in \mathbb{R}$$

It is expressive as it allows pairwise interactions between two feature vectors. In RESCAL, a bilinear model, the number of parameters grow quadratically with the number of relations. To circumvent:

LP

Impose structural constraints on bilinear maps is prevalent.

MML

Approximate the bilinear product.

Low-rank Bilinear Pooling Trick

Compared to a linear map, a bilinear map $\mathbf{W} \in \mathbb{R}^{m \times n}$ takes input as two vectors and produces a score i.e.

$$z = \mathbf{x}^T \mathbf{W} \mathbf{y} \in \mathbb{R}$$

Note that one can factorize it with two low-rank matrices, $\mathbf{U} \in \mathbb{R}^{m \times k}$, $\mathbf{V} \in \mathbb{R}^{n \times k}$:

$$\begin{aligned} z &= \mathbf{x}^T \mathbf{U} \mathbf{V}^T \mathbf{y} \\ &= \sum_{i=1}^k \mathbf{x}^T \mathbf{u}_i \mathbf{v}_i^T \mathbf{y} \\ &= \mathbf{1}^T (\mathbf{U}^T \mathbf{x} \circ \mathbf{V}^T \mathbf{y}) \end{aligned}$$

Low-rank Bilinear Pooling Trick (Cont.)

Since it returns a score only, an o -dimensional vector can be obtained with two 3D tensors:

$$\mathbf{W}_x = [\mathbf{U}_1, \mathbf{U}_2, \dots, \mathbf{U}_o] \in \mathbb{R}^{m \times k \times o} \rightarrow \mathbf{W}'_x \in \mathbb{R}^{m \times ko}$$
$$\mathbf{W}_y = [\mathbf{V}_1, \mathbf{V}_2, \dots, \mathbf{V}_o] \in \mathbb{R}^{n \times k \times o} \rightarrow \mathbf{W}'_y \in \mathbb{R}^{n \times ko}$$

The final vector in o is then obtained by k -sized non-overlapping sum pooling:

$$\mathbf{z} = \text{SumPool}(\mathbf{W}'_x{}^T \mathbf{x}, \mathbf{W}'_y{}^T \mathbf{y}, k) \in \mathbb{R}^o$$

Low-rank Bilinear Pooling Trick (Cont.)

This model, called Multi-modal Factorized Bilinear pooling (**MFB**), was introduced by [Yu et al., 2017](#).

At $k=1$, model encodes Multi-modal Low-rank Bilinear pooling (**MLB**) ([Kim et al., 2017](#)).

Earlier work of Multi-modal Compact Bilinear pooling (**MCB**) ([Fukui et al., 2016](#); [Gao et al., 2016](#)) uses sampling-based approximation that exploits the property that outer product of count sketch ([Charikar et al., 2002](#)) of two vectors can be represented as their sketches convolution. With convolution theorem:

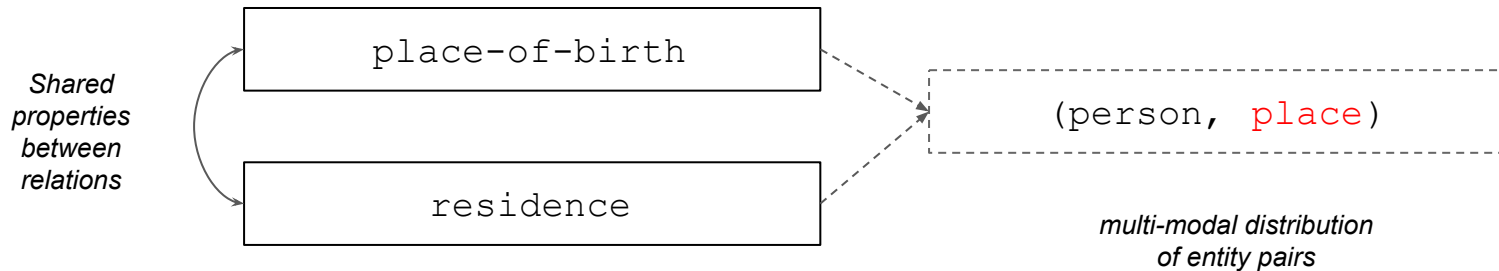
$$(\mathbf{x}\mathbf{y}^T)_{\text{proj}} = \mathbf{x}_{\text{proj}} * \mathbf{y}_{\text{proj}} = \mathcal{F}^{-1}(\mathcal{F}(\mathbf{x}_{\text{proj}}) \circ \mathcal{F}(\mathbf{y}_{\text{proj}}))$$

But requires very high-dimensional vectors (upto 16K) to perform well. MCB can be seen as closely related to Holographic Embeddings (HolE) ([Nickel et al., 2015](#)), where authors use circular correlation:

$$\mathbf{e}_s * \mathbf{e}_o = \mathcal{F}^{-1}(\overline{\mathcal{F}(\mathbf{e}_s)} \circ \mathcal{F}(\mathbf{e}_o))$$

LowFER

- MFB is simple, parameter efficient and works well in practice.
- Allows good fusion between features for better downstream performance.
- We argue that
 - good fusion between entities and relations,
 - modeling multi-relational (latent) factors of entities,
 - and parameter sharingis important for link prediction.



LowFER (Cont.)

- We therefore apply MFB in link prediction setting.
- We show that it is theoretically well sound and generalize to existing linear link prediction models.
- We show that it performs well in practice and already outperforms deep learning models at low-ranks.

LowFER (Cont.)

LowFER scoring function is defined as:

$$f(e_s, r, e_o) := \text{SumPool}(\mathbf{U}^T \mathbf{e}_s \circ \mathbf{V}^T \mathbf{r}, k)^T \mathbf{e}_o$$

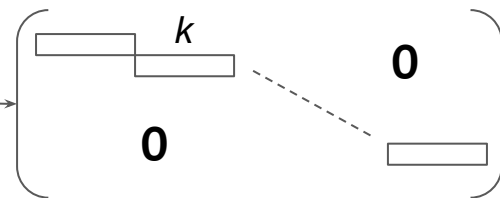
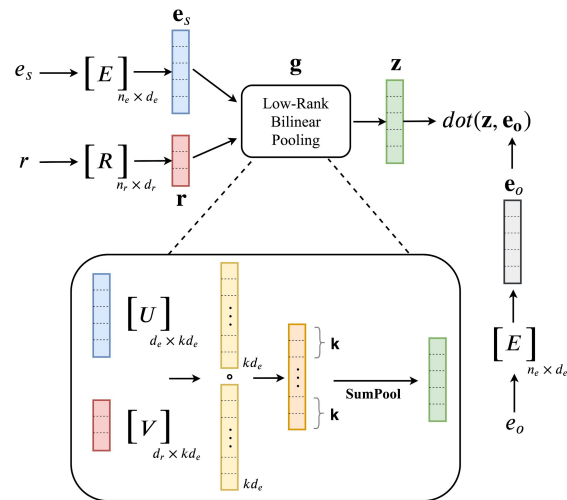
One can compactly represent the above as:

$$f(e_s, r, e_o) = (\mathbf{S}^k \text{diag}(\mathbf{U}^T \mathbf{e}_s) \mathbf{V}^T \mathbf{r})^T \mathbf{e}_o$$

where,

$$\mathbf{S}_{i,j}^k = \begin{cases} 1, & \forall j \in [(i-1)k + 1, ik] \\ 0, & \text{otherwise} \end{cases}$$

is a block diagonal matrix of k -sized one vectors.



1s Vector

Training

- Since KG only contains true triples, training requires generating negative triples with open-world assumption.
- Different negative sampling techniques exist but [Dettmers et al. \(2018\)](#) introduced a faster approach of 1-N scoring.
- For every (e_s, r, e_o) , an inverse triple is created (e_o, r^{-1}, e_s) to create the training set and for any input entity-relation pair in training set $(e_i, r_j) \in \mathcal{D}$, we score against all entities.
- Model is trained with binary cross-entropy over mini-batches instead of margin-based ranking loss, which is prone to overfitting for link prediction:

$$\min_{\Theta} \frac{1}{m} \sum_{(e,r) \in \mathcal{B}} \frac{1}{n_e} \sum_{i=1}^{n_e} (y_i \log(p(y_{(e,r,e_i)})) + (1 - y_i) \log(1 - p(y_{(e,r,e_i)})))$$

- Following [Yu et al. \(2017\)](#), to stabilize training from large values of Hadamard product, we use L2-normalization $\mathbf{x} \leftarrow \mathbf{x} / \|\mathbf{x}\|$ and power normalization $\mathbf{x} \leftarrow \text{sign}(\mathbf{x}) |\mathbf{x}|^{0.5}$.

Theoretical Analysis - I

- One of the key theoretical property of link prediction models is their ability to learn all-types of relations (*symmetric, anti-symmetric, transitive, reflexive* etc.), i.e., **fully expressive** model:

Definition 1. *Given a set of: entities \mathcal{E} , relations \mathcal{R} , correct triples $\mathcal{T} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ and incorrect triples $\mathcal{T}' = \mathcal{E} \times \mathcal{R} \times \mathcal{E} \setminus \mathcal{T}$, then a model \mathcal{M} with scoring function $f(e_s, r, e_o)$ is said to be fully expressive iff it can accurately separate \mathcal{T} from \mathcal{T}' for all $e_s, e_o \in \mathcal{E}$ and $r \in \mathcal{R}$.*

Theoretical Analysis - I (Cont.)

- Transitive models are simple and interpretable but they are theoretically limited:
 - It was first shown by [Wang et al. \(2018\)](#) that **TransE** ([Bordes et al., 2013](#)) is not fully expressive.
 - This was expanded by [Kazemi & Poole \(2018\)](#) to other translational variants including **TransH** ([Wang et al., 2014](#)), **TransR** ([Lin et al., 2015](#)), **FTransE** ([Feng et al., 2016](#)) and **STransE** ([Nguyen et al., 2016](#)).
- **DistMult** ([Yang et al., 2015](#)) enforces symmetry therefore not fully expressive.
- **ComplEx** ([Trouillon & Nickel, 2017](#)), **Simple** ([Kazemi & Poole, 2018](#)) and **TuckER** ([Balažević et al., 2019a](#)) belongs to the family of fully expressive linear models.
- Under certain conditions, by universal approximation theorem ([Hornik, 1991](#)), feed-forward neural networks can be considered fully expressive.

Theoretical Analysis - I (Cont.)

- With Proposition 1, we establish that **LowFER** is also fully expressive.

Proposition 1. *For a set of entities \mathcal{E} and a set of relations \mathcal{R} , given any ground truth \mathcal{T} , there exists an assignment of values in the LowFER model with entity embeddings of dimension $d_e = |\mathcal{E}|$, relation embeddings of dimension $d_r = |\mathcal{R}|$ and the factorization rank $k = \min(d_e, d_r)$ that makes it fully expressible.*

Theoretical Analysis - I (Cont.)

Table 1. Bounds for *fully expressive* linear models, where n is the number of true facts and the trivial bound is given by $n_e^2 n_r$.

Model	Full Expressibility Bounds
RESCAL (Nickel et al., 2011)	$(d_e, d_r) = (n_e, n_e^2)$
HolE (Nickel et al., 2016)	$d_e = d_r = 2n_e n_r + 1$
Complex (Trouillon et al., 2016)	$d_e = d_r = n_e n_r$
Simple (Kazemi & Poole, 2018)	$d_e = d_r = \min(n_e n_r, n + 1)$
Tucker (Balažević et al., 2019a)	$(d_e, d_r) = (n_e, n_r)$
LowFER	$(d_e, d_r) = (n_e, n_r)$ for $k = \min(n_e, n_r)$

Theoretical Analysis - II

Another important aspect to study is the relationships with other linear models:

- RESCAL (Nickel et al., 2011)
- DistMult (Yang et al., 2015)
- ComplEx (Trouillon et al., 2016)
- SimpleE (Kazemi and Poole, 2018)
- HypER (Balažević et al., 2019b)
- TuckER (Balažević et al., 2019a) [sota]
- LowFER

Showed that SimpleE and all models above can be consider in a family of bilinear models.

Showed that Hypernetworks based 1D-convolutional model can be seen as tensor factorization up to a non-linearity.

Showed that Tucker decomposition can generalize the family of bilinear models in LP.

We show that LowFER scoring function subsumes TuckER as low-rank approximation and further that it can accurately represent it. We further show it generalizes the family of bilinear models in LP. We also show it can generalize HypER upto a non-linearity.

Theoretical Analysis - II (TuckER) (Cont.)

TuckER (Balažević et al., 2019a) is a simple and powerful model based on Tucker decomposition (Tucker, 1966). It learns a 3D core tensor without any constraints that aims to approximate the KG binary tensor. TuckER's scoring function is given as:

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

One can re-write LowFER scoring function in terms of outer products:

$$\mathbf{S}^k(\mathbf{U}^T \mathbf{e}_s \circ \mathbf{V}^T \mathbf{r}) = \begin{bmatrix} \mathbf{e}_s^T (\sum_{i=1}^k \mathbf{u}_i \otimes \mathbf{v}_i) \mathbf{r} \\ \vdots \\ \mathbf{e}_s^T (\sum_{i=(j-1)k+1}^{jk} \mathbf{u}_i \otimes \mathbf{v}_i) \mathbf{r} \\ \vdots \\ \mathbf{e}_s^T (\sum_{i=k(d_e-1)}^{kd_e} \mathbf{u}_i \otimes \mathbf{v}_i) \mathbf{r} \end{bmatrix}$$

Theoretical Analysis - II (TuckER) (Cont.)

From last formulation, pulling the entity and relation embeddings out, one can realize it another way. Take k -distance apart slices from \mathbf{U} and \mathbf{V} such that l -th slice is as:

$$\mathbf{W}_U^{(l)} = [\mathbf{u}_l, \mathbf{u}_{k+l}, \dots, \mathbf{u}_{k(d_e-1)+l}] \in \mathbb{R}^{d_e \times d_e}$$

$$\mathbf{W}_V^{(l)} = [\mathbf{v}_l, \mathbf{v}_{k+l}, \dots, \mathbf{v}_{k(d_e-1)+l}] \in \mathbb{R}^{d_r \times d_e}$$

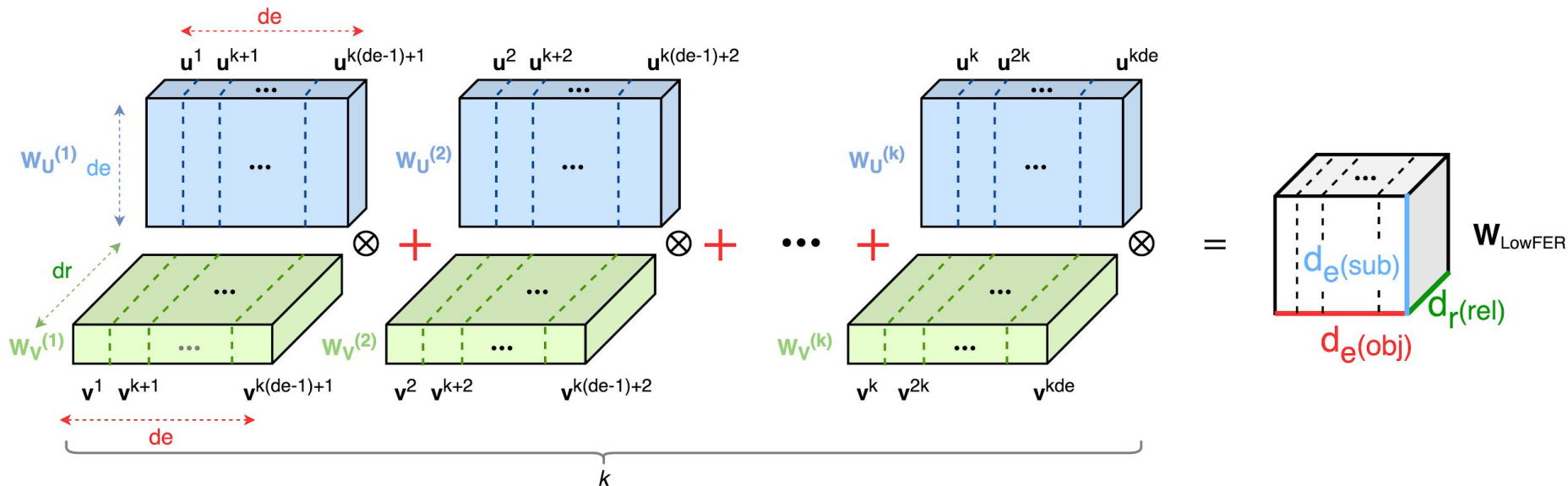
Taking the column wise outer product (commonly referred as the mode-2 Khatri-Rao product ([Cichocki et al., 2016](#))) generates k 3D tensors, which added together approximates TuckER tensor:

$$\mathbf{W} = \sum_{i=1}^k \mathbf{W}_U^{(i)} \otimes \mathbf{W}_V^{(i)}$$

$\otimes \triangleq$ Column-wise outer product of two matrices, where the resultant matrices are stacked to form 3D tensor.

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

Theoretical Analysis - II (Tucker) (Cont.)



Low-rank approximation of the core tensor of Tucker with LowFER by summing k low-rank 3D tensors, where each tensor is obtained by stacking d_e rank-1 matrices obtained by the outer product of k -part columns of \mathbf{U} and \mathbf{V} .

Theoretical Analysis - II (TuckER) (Cont.)

- It is straightforward to show that LowFER can accurately model TuckER under following conditions:

Proposition 2. *Given a TuckER model with entity embedding dimension d_e , relation embedding dimension d_r and core tensor \mathcal{W} , there exists a LowFER model with $k \leq \min(d_e, d_r)$, entity embedding dimension d_e and relation embedding dimension d_r that accurately represents the former.*

Theoretical Analysis - II (Cont.)

Hence, following two equations can be interchangeably used for LowFER (they are equivalent):

$$f(e_s, r, e_o) = (\mathbf{S}^k \text{diag}(\mathbf{U}^T \mathbf{e}_s) \mathbf{V}^T \mathbf{r})^T \mathbf{e}_o$$

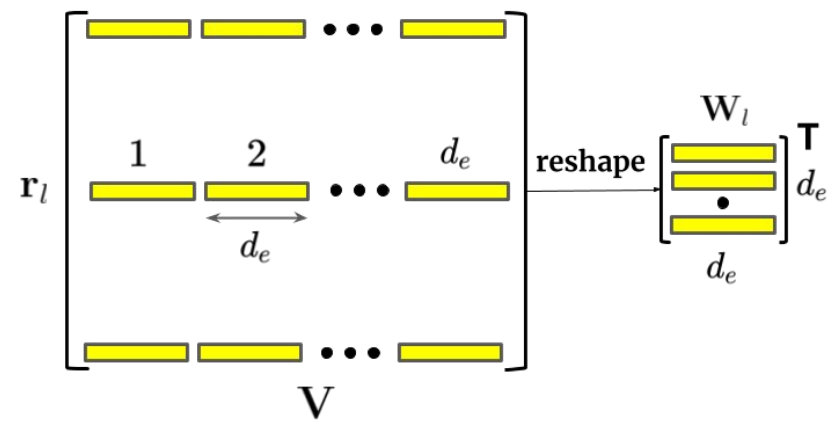
$$f(e_s, r, e_o) = \left(\sum_{i=1}^k \mathbf{W}_U^{(i)} \otimes \mathbf{W}_V^{(i)} \right) \times_1 \mathbf{e}_s \times_2 \mathbf{e}_o \times_3 \mathbf{r}$$

Theoretical Analysis - II (RESCAL) (Cont.)

(Nickel et al., 2011) $\text{RESCAL}(e_s, r_l, e_o) = \mathbf{e}_s^T \mathbf{W}_l \mathbf{e}_o$ ●

$$k = d_e, d_r = n_r, \mathbf{R} = \mathbf{I}_{n_r}, \mathbf{V} \in \mathbb{R}^{n_r \times d_e^2}$$

$$\mathbf{U} = [\mathbf{I}_{d_e} \mid \mathbf{I}_{d_e} \mid \dots \mid \mathbf{I}_{d_e}] \in \mathbb{R}^{d_e \times d_e^2}$$



With these conditions and Eq. ● == Eq. ●

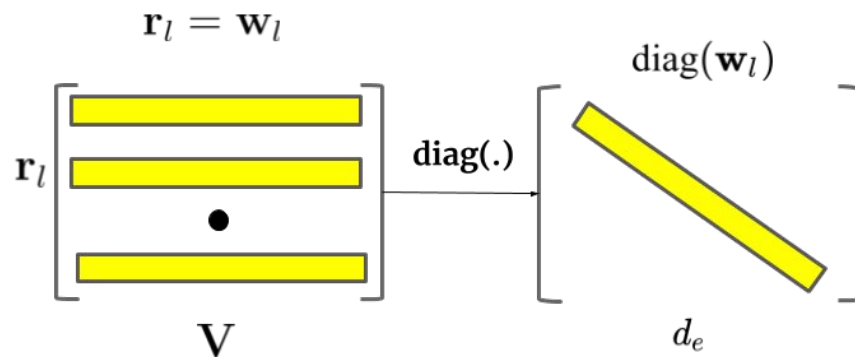
Parameters Vector

Theoretical Analysis - II (DistMult) (Cont.)

(Yang et al., 2015) $\text{DistMult}(e_s, r_l, e_o) = \mathbf{e}_s^T \text{diag}(\mathbf{w}_l) \mathbf{e}_o$ ●

$$k = 1, d_r = n_r, \mathbf{R} = \mathbf{I}_{n_r}, \mathbf{V} \in \mathbb{R}^{n_r \times d_e}$$

$$\mathbf{U} = \mathbf{I}_{d_e} \in \mathbb{R}^{d_e \times d_e}$$



With these conditions and Eq. ● == Eq. ●

 Parameters Vector

Theoretical Analysis - II (SimpleE) (Cont.)

$$\text{SimpleE}(e_s, r_l, e_o) = \frac{1}{2}(\mathbf{h}_{e_s}^T \text{diag}(\mathbf{r}_l) \mathbf{t}_{e_o} + \mathbf{h}_{e_o}^T \text{diag}(\mathbf{r}_l^{-1}) \mathbf{t}_{e_s}) \quad (\text{Kazemi \& Poole, 2018})$$

Reformulation to simple bilinear form:

$$\text{SimpleE}(e_s, r_l, e_o) = \frac{1}{2} \hat{\mathbf{e}}_s^T \text{diag}(\hat{\mathbf{r}}_l) \mathbf{e}_o \quad \bullet$$

$$\hat{\mathbf{e}}_s = [\mathbf{t}_{e_s}; \mathbf{h}_{e_s}] \in \mathbb{R}^{2d}$$

$$\mathbf{e}_o = [\mathbf{h}_{e_o}; \mathbf{t}_{e_o}] \in \mathbb{R}^{2d}$$

$$\hat{\mathbf{r}}_l = [\mathbf{r}_l^{-1}; \mathbf{r}_l] \in \mathbb{R}^{2d}$$

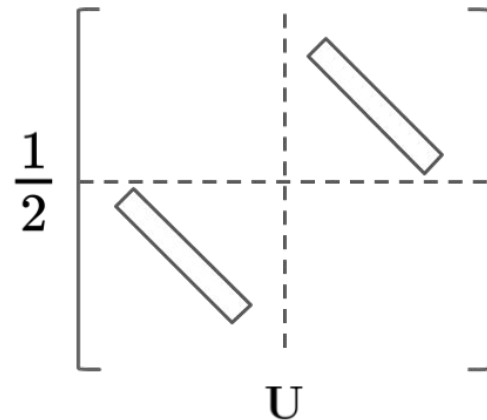
$$d_e = 2d$$

$$k = 1, d_e = 2d, d_r = n_r, \mathbf{R} = \mathbf{I}_{n_r}, \mathbf{V} \in \mathbb{R}^{n_r \times 2d}$$

$$\mathbf{U} \in \mathbb{R}^{2d \times 2d} : \mathbf{U}_{12} = \mathbf{U}_{21} = \frac{1}{2} \mathbf{I}_d \text{ else } 0$$

With these conditions and Eq. ● == Eq. ●

 1s Vector



Theoretical Analysis - II (Complex) (Cont.)

$$\begin{aligned}\text{Complex}(e_s, r_l, e_o) = & \text{Re}(\mathbf{e}_s)^T \text{diag}(\text{Re}(\mathbf{r}_l)) \text{Re}(\mathbf{e}_o) \\ & + \text{Im}(\mathbf{e}_s)^T \text{diag}(\text{Re}(\mathbf{r}_l)) \text{Im}(\mathbf{e}_o) \\ & + \text{Re}(\mathbf{e}_s)^T \text{diag}(\text{Im}(\mathbf{r}_l)) \text{Im}(\mathbf{e}_o) \\ & - \text{Im}(\mathbf{e}_s)^T \text{diag}(\text{Im}(\mathbf{r}_l)) \text{Re}(\mathbf{e}_o)\end{aligned}$$

(Trouillon et al., 2016)

$$\text{Complex}(e_s, r_l, e_o) = \hat{\mathbf{e}}_s^T \mathbf{W}_l \hat{\mathbf{e}}_o \quad \bullet$$


$$\hat{\mathbf{e}}_s = [\text{Re}(\mathbf{e}_s); \text{Im}(\mathbf{e}_s)] \in \mathbb{R}^{2d}$$


$$\hat{\mathbf{e}}_o = [\text{Re}(\mathbf{e}_o); \text{Im}(\mathbf{e}_o)] \in \mathbb{R}^{2d}$$

$$\mathbf{W}_l \in \mathbb{R}^{2d \times 2d} : \mathbf{W}_l^{(11)} = \mathbf{W}_l^{(22)} = \text{diag}(\text{Re}(\mathbf{r}_l)),$$

$$\mathbf{W}_l^{(12)} = \text{diag}(\text{Im}(\mathbf{r}_l)), \mathbf{W}_l^{(21)} = \text{diag}(-\text{Im}(\mathbf{r}_l))$$

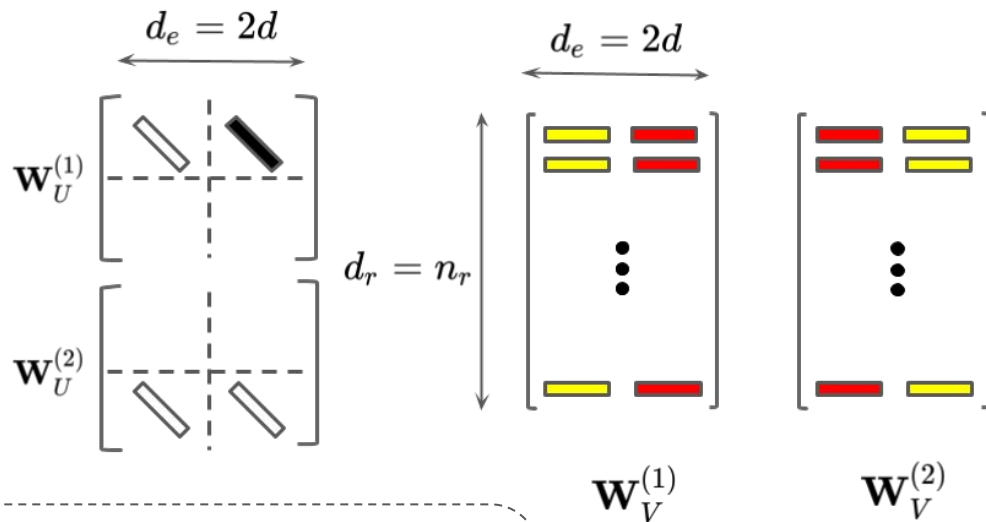
Theoretical Analysis - II (Complex) (Cont.)

 1s Vector

 -1s Vector

 Real Parameters Vector



 Imaginary Parameters Vector



$$k = 2, d_e = 2d, d_r = n_r, \mathbf{R} = \mathbf{I}_{n_r}$$

$$\mathbf{U} \in \mathbb{R}^{2d \times 4d} : \mathbf{W}_{U_{11}}^{(1)} = \mathbf{I}_d, \mathbf{W}_{U_{12}}^{(1)} = -\mathbf{I}_d, \text{ else } 0$$

$$\mathbf{V} \in \mathbb{R}^{n_r \times 4d} : \mathbf{W}_V^{(1)}[l] = [\text{Re}(\mathbf{r}_l); \text{Im}(\mathbf{r}_l)], \mathbf{W}_V^{(2)}[l] = [\text{Im}(\mathbf{r}_l); \text{Re}(\mathbf{r}_l)]$$

With these conditions and Eq.  == Eq. 

Theoretical Analysis - II (HypER) (Cont.)

HypER (Balažević et al., 2019b) is a convolutional model that uses hypernetworks (Ha et al., 2017) to generate 1D convolutional filters. The authors showed that it can be seen as tensor factorization method upto a non-linearity. The scoring function is defined as:

$$\text{HypER}(e_s, r, e_o) = \text{ReLU}(\text{vec}(\mathbf{e}_s * \mathbf{F}_r) \mathbf{W})^T \mathbf{e}_o$$

Convolution can be expressed as matrix multiplication, where the filter is a Toeplitz matrix. For n filters and d dimensions, this results in sparse 4D tensor where each 3D tensor is a block diagonal Toeplitz matrix representing each filter, then:

$$\begin{aligned} \text{vec}(\mathbf{e}_s * \mathbf{F}_r) \mathbf{W} &= (\mathcal{F} \otimes \mathcal{W}) \times_2 \mathbf{e}_s \times_1 \mathbf{r} \quad \bullet \\ \mathcal{T} = \mathcal{F} \otimes \mathcal{W} &\in \mathbb{R}^{d_r \times d_e \times d_e} \rightarrow \mathbf{T} \in \mathbb{R}^{d_r \times d_e^2} \end{aligned}$$

With these conditions and Eq. ●
== Eq. ●

$$\begin{aligned} k &= d_e, \mathbf{V} = \mathbf{T} \\ \mathbf{U} &= [\mathbf{I}_{d_e} \mid \mathbf{I}_{d_e} \mid \dots \mid \mathbf{I}_{d_e}] \in \mathbb{R}^{d_e \times d_e^2} \end{aligned}$$

LowFER Complexity

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)	$\text{Re}(\langle \mathbf{e}_s, \mathbf{w}_r, \bar{\mathbf{e}}_o \rangle)$	$\mathbf{w}_r \in \mathbb{C}^{d_e}$	$\mathcal{O}(n_e d_e + n_r d_e)$
ConvE (Dettmers et al., 2018)	$f(\text{vec}(f([\underline{\mathbf{e}}_s; \underline{\mathbf{w}}_r] * w))) \mathbf{W} \mathbf{e}_o$	$\mathbf{w}_r \in \mathbb{R}^{d_r}$	$\mathcal{O}(n_e d_e + n_r d_r)$
Simple (Kazemi and Poole, 2018)	$\frac{1}{2}(\langle \mathbf{h}_{e_s}, \mathbf{w}_r, \mathbf{t}_{e_o} \rangle + \langle \mathbf{h}_{e_o}, \mathbf{w}_{r-1}, \mathbf{t}_{e_s} \rangle)$	$\mathbf{w}_r \in \mathbb{R}^{d_e}$	$\mathcal{O}(n_e d_e + n_r d_e)$
HypER (Balažević et al., 2019)	$f(\text{vec}(\mathbf{e}_s * \text{vec}^{-1}(\mathbf{w}_r \mathbf{H})) \mathbf{W} \mathbf{e}_o$	$\mathbf{w}_r \in \mathbb{R}^{d_r}$	$\mathcal{O}(n_e d_e + n_r d_r)$
TuckER (Balažević et al., 2019)	$\mathcal{W} \times_1 \mathbf{e}_s \times_2 \mathbf{w}_r \times_3 \mathbf{e}_o$	$\mathbf{w}_r \in \mathbb{R}^{d_r}$	$\mathcal{O}(n_e d_e + n_r d_r)$
LowFER	$(\mathbf{S}^k \text{diag}(\mathbf{U}^T \mathbf{e}_s) \mathbf{V}^T \mathbf{w}_r)^T \mathbf{e}_o$	$\mathbf{w}_r \in \mathbb{R}^{d_r}$	$\mathcal{O}(n_e d_e + n_r d_r)$

* Table taken from TuckER (Balažević et al., 2019a)

LowFER Complexity (Cont.)

Model	Majority Parameters	Shared Parameters	Time Complexity
Tucker	$\mathcal{O}(n_e d_e + n_r d_r)$	$\mathcal{O}(d_e^2 d_r)$	$\mathcal{O}(d_e^2 d_r)$
LowFER	$\mathcal{O}(n_e d_e + n_r d_r)$	$\mathcal{O}(k(d_e^2 + d_e d_r))$	$\mathcal{O}(k(d_e^2 + d_e d_r))$
Tucker	$\mathcal{O}((n_e + n_r)d)$	$\mathcal{O}(d^3)$	$\mathcal{O}(d^3)$
LowFER	$\mathcal{O}((n_e + n_r)d)$	$\mathcal{O}(k d^2)$	$\mathcal{O}(k d^2)$

$: d = d_e = d_r$

Consider [Lacroix et al \(2018\)](#), where authors take $d=2000$; Tucker requires $> 8B$ parameters (~ 24 times larger than sota NLU models like BERT-large); LowFER needs only $k \times 4M$, with k controlling the growth. At $k=d_e/2$ we expect similar performance as LowFER has same number of parameters as Tucker.

Experiments - Data

We conducted the experiments on four benchmark datasets: WN18 ([Bordes et al., 2013](#)), WN18RR ([Dettmers et al., 2018](#)), FB15k ([Bordes et al., 2013](#)) and FB15k-237 ([Toutanova et al., 2015](#)).

Dataset	n_e	n_r	$\sim n_e/n_r$	Training	Validation	Testing
WN18	40,943	18	2275	141,442	5,000	5,000
WN18RR	40,943	11	3722	86,835	3,034	3,134
FB15k	14,951	1,345	11	483,142	50,000	59,071
FB15k-237	14,541	237	61	272,115	17,535	20,466

Experiments - Main Results I

Linear	Model	WN18RR				FB15k-237				
		MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10	
No	ConvE (Dettmers et al., 2018)	0.430	0.400	0.440	0.520	0.325	0.237	0.356	0.501	5.1
	R-GCN (Schlichtkrull et al., 2018)	–	–	–	–	0.248	0.151	0.264	0.417	
	RotatE (Sun et al., 2019)	–	–	–	–	0.297	0.205	0.328	0.480	
	HypER (Balažević et al., 2019b)	<u>0.465</u>	<u>0.436</u>	0.477	0.522	0.341	0.252	0.376	0.520	4.3
Yes	DistMult (Yang et al., 2015)	0.430	0.390	0.440	0.490	0.241	0.155	0.263	0.419	
	ComplEx (Trouillon et al., 2016)	0.440	0.410	0.460	0.510	0.247	0.158	0.275	0.428	6.0
	TuckER (Balažević et al., 2019a)	0.470	0.443	0.482	0.526	<u>0.358</u>	0.266	<u>0.394</u>	0.544	11.0
	LowFER-1	0.454	0.422	0.470	0.515	0.318	0.233	0.348	0.483	3.0
	LowFER-10	0.464	0.433	0.477	<u>0.523</u>	0.352	<u>0.261</u>	0.386	<u>0.533</u>	3.8
	LowFER- k^*	<u>0.465</u>	0.434	<u>0.479</u>	0.526	0.359	0.266	0.396	0.544	11.3

Model parameters (in millions)

Experiments - Main Results II

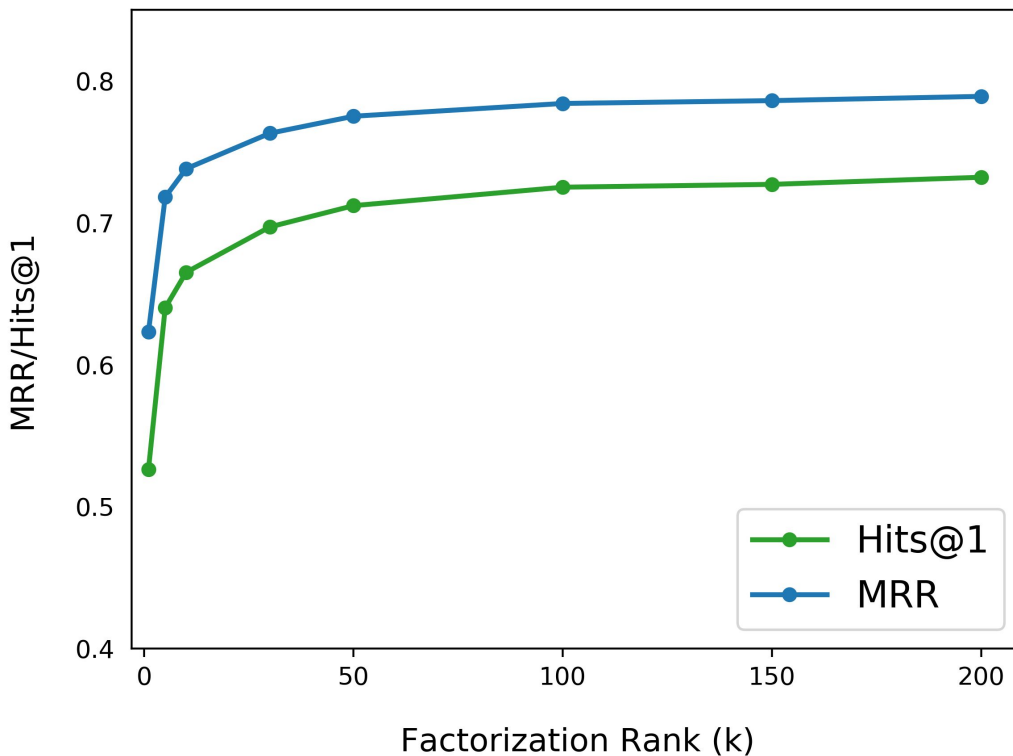
Linear	Model	WN18				FB15k				
		MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10	
No	TransE (Bordes et al., 2013)	0.454	0.089	0.823	0.934	0.380	0.231	0.472	0.641	
	R-GCN (Schlichtkrull et al., 2018)	0.819	0.697	0.929	0.964	0.696	0.601	0.760	0.842	
	HolE (Nickel et al., 2016)	0.938	0.930	0.945	0.949	0.524	0.402	0.613	0.739	
	TorusE (Ebisu & Ichise, 2018)	0.947	0.943	0.950	0.954	0.733	0.674	0.771	0.832	
	ConvE (Dettmers et al., 2018)	0.943	0.935	0.946	0.956	0.657	0.558	0.723	.831	
	HypER (Balažević et al., 2019b)	<u>0.951</u>	<u>0.947</u>	0.955	<u>0.958</u>	0.790	0.734	0.829	0.885	
Yes	ANALOGY (Liu et al., 2017)	0.942	0.939	0.944	0.947	0.725	0.646	0.785	0.854	
	DistMult (Yang et al., 2015)	0.822	0.728	0.914	0.936	0.654	0.546	0.733	0.824	
	ComplEx (Trouillon et al., 2016)	0.941	0.936	0.936	0.947	0.692	0.599	0.759	0.840	6.5
	Simple (Kazemi & Poole, 2018)	0.942	0.939	0.944	0.947	0.727	0.660	0.773	0.838	6.5
	TuckER (Balažević et al., 2019a)	0.953	0.949	0.955	<u>0.958</u>	0.795	0.741	0.833	0.892	11.3
	LowFER-1	0.949	0.945	0.951	0.956	0.720	0.639	0.774	0.859	4.6
	LowFER-10	0.950	0.946	<u>0.952</u>	<u>0.958</u>	<u>0.810</u>	<u>0.760</u>	<u>0.843</u>	<u>0.896</u>	5.5
LowFER- k^*	0.950	0.946	<u>0.952</u>	<u>0.958</u>	0.824	0.782	0.852	0.897	9.5	

Model parameters (in millions)

Experiments - Effect of k

- LowFER performs extremely well at low-ranks ($k \leq 10$) in practice compared to extremely large k as in Prop. 1.
- Suggests, TuckER is extremely over-parameterized.
- We note that the effect of k is inversely related to the ratio ne/nr .
- At $k=de/2$, performance almost equivalent to TuckER.
- As $ne \gg de$, the influence of shared parameters is negligible as most of knowledge learned by embeddings (WN18/RR).

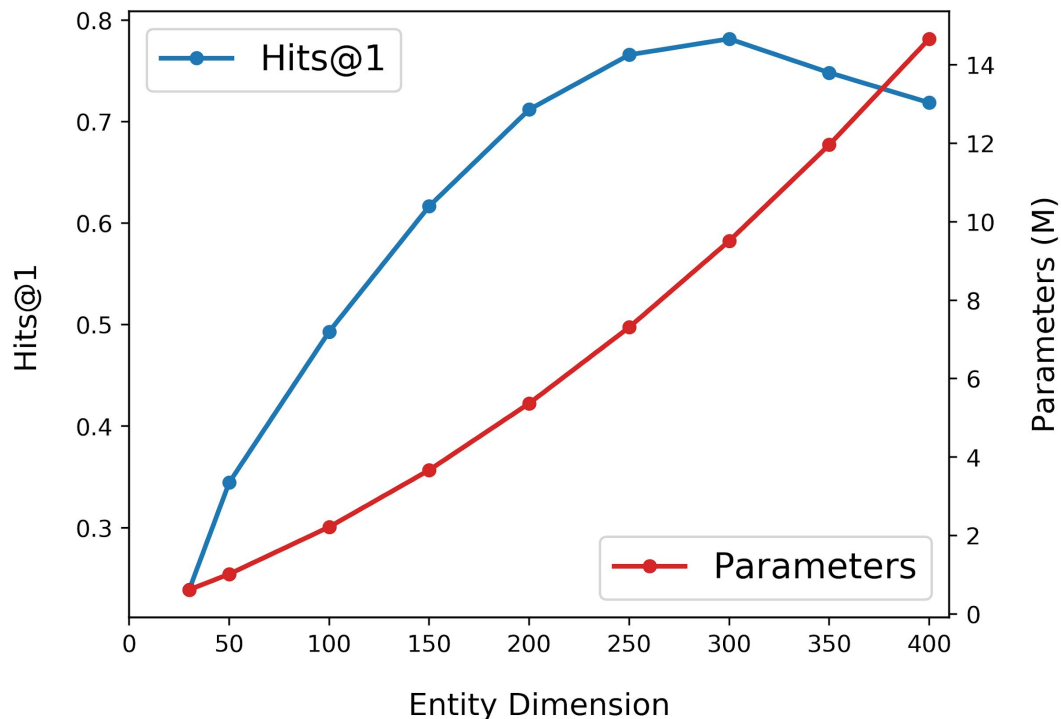
Effect of changing k on FB15k ($de=200$, $dr=30$).



Experiments - Embedding dimension I

Effect of changing de on FB15k (dr=30, k=50)

- Changing entity embedding dimension has significant effect on Hits@1.
- Starts overfitting after de=300.



Experiments - Relation Results

WN18 contains redundant relations $\forall e_i, e_j \in \mathcal{E} : (e_i, r_1, e_j) \in \mathcal{T} \Leftrightarrow (e_j, r_2, e_i) \in \mathcal{T}$ because one can be inferred from the other (Kazemi & Poole, 2018) and therefore WN18RR (a harder dataset) was created with those artifacts removed by Dettmers et al. (2018).

On WN18RR, we see that both TuckER and LowFER retain their performance on symmetric relations (such as `derivationally_related_form`) but perform poorly on anti-symmetric relations (such as `hypernym`)

Only SimpleE (Kazemi & Poole, 2018) has formally shown to incorporate background knowledge with weight tying (cf. Prop. 3, 4, 5). Since, LowFER can subsume SimpleE, such rules can be studied to extend for LowFER.

LowFER: **-70.6%** TuckER: **-69.3%**

	WN18		WN18RR	
	LowFER	TuckER	LowFER	TuckER
<code>also_see</code>	0.638	0.630	0.627	0.614
<code>derivationally_related_form</code>	0.954	0.956	0.957	0.957
<code>has_part</code>	0.944	0.945	0.138	0.129
<code>hypernym</code>	0.961	0.962	0.189	0.189
<code>instance_hypernym</code>	0.986	0.982	0.576	0.591
<code>member_meronym</code>	0.930	0.927	0.155	0.131
<code>member_of_domain_region</code>	0.885	0.885	0.060	0.083
<code>member_of_domain_usage</code>	0.917	0.917	0.025	0.096
<code>similar_to</code>	1.0	1.0	1.0	1.0
<code>synset_domain_topic_of</code>	0.956	0.952	0.494	0.499
<code>verb_group</code>	0.974	0.974	0.974	0.974

Conclusion

- We proposed a simple and parameter efficient linear model.
- Exhibits interesting theoretical properties.
- Generalizes existing linear models in KGC literature.
- Constraints-free, allows for parameter sharing.
- Outperforms deep learning models by large-margin.
- Performs really well at low-ranks, indicating over-parameterization in TuckER.
- Reaches on-par or state-of-the-art performance on several datas
- Parameter sharing is key to improve the sota.
- Still limited to learn harder relations without background knowledge.

Thank You!

References

For full references list, please check the paper.