LowFER: Low-rank Bilinear Pooling for Link Prediction

Saadullah Amin [1] **Stalin Varanasi** [1 2] **Katherine Ann Dunfield** [1 2] **Günter Neumann** [1 2]

Abstract

Knowledge graphs are incomplete by nature, with only a limited number of observed facts from the world knowledge being represented as structured relations between entities. To partly address this issue, an important task in statistical relational learning is that of *link prediction* or *knowledge graph completion*. Both linear and non-linear models have been proposed to solve the problem. Bilinear models, while expressive, are prone to overfitting and lead to quadratic growth of parameters in number of relations. Simpler models have become more standard, with certain constraints on bilinear map as relation parameters. In this work, we propose a factorized bilinear pooling model, commonly used in multi-modal learning, for better fusion of entities and relations, leading to an efficient and constraint-free model. We prove that our model is fully expressive, providing bounds on the embedding dimensionality and factorization rank. Our model naturally generalizes Tucker decomposition based TuckER model, which has been shown to generalize other models, as efficient low-rank approximation without substantially compromising the performance. Due to low-rank approximation, the model complexity can be controlled by the factorization rank, avoiding the possible cubic growth of TuckER. Empirically, we evaluate on real-world datasets, reaching on par or state-of-the-art performance. At extreme low-ranks, model preserves the performance while staying parameter efficient.

1. Introduction

Knowledge graphs (KGs) are large collections of structured knowledge, organized as subject and object entities and relations, in the form of fact triples <*sub*, rel, *obj*>. The usefulness of knowledge graphs, however, is affected primarily by their incompleteness. The task of *link prediction* or *knowledge graph completion* (KGC) aims to infer missing facts from existing ones, by essentially *scoring* a relation and entities triple for use in predicting its validity, and thereby avoiding the cost and time of extending knowledge graphs manually. To accomplish this, several models have been proposed, including linear and non-linear models. Bilinear models have additionally been used in multi-modal learning due to their expressive nature, where the fusion of features from different modalities plays a key role towards the performance of a model, with concatenation or element-wise summation being commonly used fusion techniques. The underlying assumption is that the distributions of features across modalities may vary significantly, and the representation capacity of the fused features may be insufficient, therefore limiting the final prediction performance (Yu et al., 2017). In this work, we apply this assumption to knowledge graphs by considering that the *entities* and *relations* come from different multi-modal distributions and good fusion between them can potentially construct a KG.

A major drawback of using bilinear modeling methods is the quadratic growth of parameters, which results in high computational and memory costs and risks overfitting. In multi-modal learning, *factorization* techniques have therefore been researched to address these challenges (Kim et al., 2016; Fukui et al., 2016; Yu et al., 2017; Ben-Younes et al., 2017; Li et al., 2017; Liu et al., 2018), and *constraints* based bilinear maps have become a more prevalent standard in link prediction (Yang et al., 2015; Trouillon et al., 2016; Kazemi & Poole, 2018). Applying constraints can be seen as hard regularization since it allows for incorporating background knowledge (Kazemi & Poole, 2018), but restricts the learning potential of the model due to limited parameter sharing (Balažević et al., 2019a). We focus on a constraint-free approach, using the low-rank factorization of bilinear models, as it offers flexibility and generalizes well, naturally leading to other models under certain conditions. Our work extends the multi-modal factorized bilinear pooling (MFB) model, introduced by Yu et al. (2017), and applies it to the link prediction task.

Our contributions are outlined as follows:

[1]German Research Center for Artificial Intelligence (DFKI), Saarbrücken, Germany [2]Department of Language Science and Technology, Saarland University, Saarbrücken, Germany. Correspondence to: Saadullah Amin <saadullah.amin@dfki.de>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

- We propose a simple and parameter efficient linear model by extending multi-modal factorized bilinear (MFB) pooling (Yu et al., 2017) for link prediction.
- We prove that our model is *fully expressive* and provide bounds on entity and relation embedding dimensions, along with the factorization rank.
- We provide relations to the family of bilinear link prediction models and Tucker decomposition (Tucker, 1966) based TuckER model (Balažević et al., 2019a), generalizing them as special cases. We also show the relation to 1D convolution based HypER model (Balažević et al., 2019b), bridging the gap from bilinear to convolutional link prediction models.
- On real-world datasets the model achieves on par or state-of-the-art performance, where at extreme low-ranks, with limited number of parameters, it outperforms most of the prior arts, including deep learning based models.

2. Related Work

Given a set of entities $\mathcal{E}$ and relations $\mathcal{R}$ in a knowledge graph $\mathcal{KG}$, the task of link prediction is to assign a score s to a triple (e_s, r, e_o):

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

where $e_s \in \mathcal{E}$ is the *subject* entity, $e_o \in \mathcal{E}$ is the *object* entity and $r \in \mathcal{R}$ is the *relation* between them. The scoring function f estimates the general binary tensor $\mathbf{T} \in |\mathcal{E}| \times |\mathcal{R}| \times |\mathcal{E}|$, by assigning a score of 1 to $\mathbf{T}_{ijk}$ if relation r_j exists between entities e_i and e_k, 0 otherwise. The scoring function can be a linear or non-linear model, trained to predict true triples in a $\mathcal{KG}$.

Deep learning based scoring functions such as ConvE (Dettmers et al., 2018) and HypER (Balažević et al., 2019b) use 2D and 1D convolution on subject entity and relation representations respectively. Both perform well in practice and are efficient, but the former lacks direct interpretation, whereas the latter has shown to be related to tensor factorization. Transitional methods (Bordes et al., 2013; Wang et al., 2014; Ji et al., 2015; Lin et al., 2015; Nguyen et al., 2016; Feng et al., 2016) use additive dissimilarity scoring functions, whereby they differ in terms of the constraints applied to the projection matrices. While interpretable, they are theoretically limited as they have shown to be not *fully expressive* (Wang et al., 2018; Kazemi & Poole, 2018). There are several other related works (Nickel et al., 2016; Das et al., 2017; Yang et al., 2017; Shen et al., 2018; Schlichtkrull et al., 2018; Ebisu & Ichise, 2018; Sun et al., 2019), but we will mainly focus on different types of linear models here, as they are more relevant to our work.

All discussed linear models can be seen as a decomposition of the tensor $\mathbf{T}$, using different factorization methods. One way to factorize this tensor is to factorize its slices in the relation dimension with DEDICOMP (Harshman, 1978). RESCAL (Nickel et al., 2011), a relaxed version of DEDICOMP, decomposes using a scoring function that consists of a bilinear product between subject and object entity vectors with a relation specific matrix. RESCAL, however, tends to overfit due to the quadratic growth of parameters in number of relations. Others use Canonical Polyadic decomposition (CPD or simply CP) (Hitchcock, 1927; Harshman & Lundy, 1994) to factorize the binary tensor. In CP, each value in the tensor is obtained as a sum of multiple Hadamard products of three vectors, representing subject, object and relation. DistMult (Yang et al., 2015), equivalent to INDSCAL (Carroll & Chang, 1970), is as such and uses a diagonal relation matrix, unlike RESCAL, to account for overfitting. ComplEx (Trouillon et al., 2016; Trouillon & Nickel, 2017) uses complex valued vectors for entities and relations to explicitly model asymmetric relations. SimplE (Kazemi & Poole, 2018) extends CP by introducing two vectors (*head* and *tail*) for each entity and two for relations (including the inverse). Tucker decomposition (Tucker, 1966) based TuckER (Balažević et al., 2019a) learns a 3D core tensor, which is multiplied with a matrix along each mode to approximate the binary tensor. A key difference between CP based methods and TuckER is that it learns representations not only via embeddings, but also through a shared core tensor.

3. Model

Downstream performance for tasks such as visual question answering strongly depends on the multi-modal fusion of features to leverage the heterogeneous data (Liu et al., 2018). Bilinear models are expressive as they allow for pairwise interactions between the feature dimensions but also introduce huge number of parameters that lead to high computational and memory costs and the risk of overfitting (Fukui et al., 2016). Substantial research has therefore focused on efficiently computing the bilinear product. In multi-modal compact bilinear (MCB) pooling (Gao et al., 2016; Fukui et al., 2016), authors employ a sampling based approximation that uses the property that the tensor sketch projection (Charikar et al., 2004; Pham & Pagh, 2013) of the outer product of two vectors can be represented as their sketches convolution. Multi-modal low-rank bilinear (MLB) pooling (Kim et al., 2016) uses two low-rank projection matrices to transform the features from the original space to a common space, followed by the Hadamard product, which was later generalized by the multi-modal factorized bilinear (MFB) pooling (Yu et al., 2017). Our work is based on the MFB model but can also be seen as related to Liu et al. (2018). In contrast to KGC bilinear models, these bilinear models allow for parameter sharing and generally, are constraint-free.

3.1. Multi-modal Factorized Bilinear Pooling (MFB)

Given two feature vectors $\mathbf{x} \in \mathbb{R}^m$, $\mathbf{y} \in \mathbb{R}^n$ and a bilinear map $\mathbf{W} \in \mathbb{R}^{m \times n}$, the bilinear transformation is defined as $z = \mathbf{x}^T \mathbf{W} \mathbf{y} \in \mathbb{R}$. To obtain a vector in $\mathbb{R}^o$, o such maps have to be learned (e.g. in RESCAL these would be relation specific matrices), resulting in large number of parameters. However, $\mathbf{W}$ can be factorized into two low-rank matrices:

$$z = \mathbf{x}^T \mathbf{U} \mathbf{V}^T \mathbf{y} = \mathbf{1}^T (\mathbf{U}^T \mathbf{x} \circ \mathbf{V}^T \mathbf{y})$$

where $\mathbf{U} \in \mathbb{R}^{m \times k}$, $\mathbf{V} \in \mathbb{R}^{n \times k}$, k is the factorization rank, $\circ$ is the element-wise product of two vectors and $\mathbf{1} \in \mathbb{R}^k$ is vector of all ones. Therefore, to obtain a output feature vector $\mathbf{z} \in \mathbb{R}^o$, two 3D tensors are required, $\mathbf{W}_x = [\mathbf{U}_1, \mathbf{U}_2, ..., \mathbf{U}_o] \xrightarrow{reshape} \mathbf{W}_x^{'}$ and $\mathbf{W}_y = [\mathbf{V}_1, \mathbf{V}_2, ..., \mathbf{V}_o] \xrightarrow{reshape} \mathbf{W}_y^{'}$, where $\mathbf{W}_x \in \mathbb{R}^{m \times k \times o}$, $\mathbf{W}_y \in \mathbb{R}^{n \times k \times o}$ are 3D tensors and $\mathbf{W}_x^{'} \in \mathbb{R}^{m \times ko}$, $\mathbf{W}_y^{'} \in \mathbb{R}^{n \times ko}$ are their reshaped 2D matrices respectively. The final (fused) vector $\mathbf{z}$ is then obtained by summing non-overlapping windows of size k over the Hadamard product of projected vectors using $\mathbf{W}_x^{'}$ and $\mathbf{W}_y^{'}$:

$$\mathbf{z} = \text{SumPool}(\mathbf{W}_x^{'T} \mathbf{x} \circ \mathbf{W}_y^{'T} \mathbf{y}, k) \tag{1}$$

At $k = 1$, MFB reduces to MLB, which converges slowly, and MCB requires very high-dimensional vectors to perform well (Yu et al., 2017). Further, MFB significantly lowers the number of parameters with low-rank factorized matrices and leads to better performance.

3.2. Low-rank Bilinear Pooling for Link Prediction

Consider that *entities* and *relations* are not intrinsically bound and come from two different modalities, such that *good* fusion between them can potentially result in a knowledge graph of fact triples. Entities and relations can be shown to possess certain properties that allow them to function similarly to others within the same modality. For example, the relation *place-of-birth* shares inherent properties with the relation *residence*. As such, similar entity pairs can yield similar relations, given appropriate shared properties. Like in multi-modal auditory-visual fusion, where the sound of a roar may better predict a resulting image within the distribution of animals that roar, a relation such as *place-of-birth*, can better predict an entity pair within a distribution of (*person*, *place*) entity pairs. In link prediction, we assume that the latent decomposition with MFB can help the model capture different aspects of interactions between an entity and a relation, which can lead to better scoring with the missing entity. We therefore, apply the **Low**-rank **F**actorization trick of bilinear maps with k-sized non-overlapping summation pooling (section 3.1) to **E**ntities and **R**elations (LowFER).

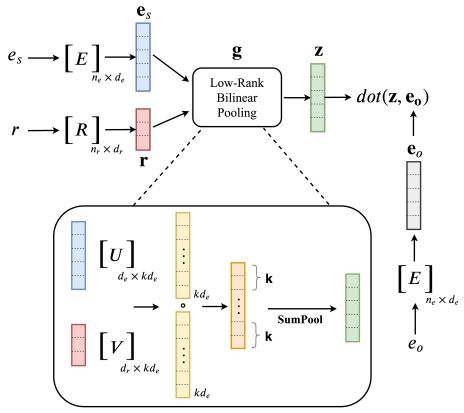

Figure 1. Overview of the LowFER model. For an input tuple (e_s, r) and target entity e_o, we first get entity vectors $\mathbf{e}_s, \mathbf{e}_o \in \mathbb{R}^{d_e}$ from entity embedding matrix $\mathbf{E} \in \mathbb{R}^{n_e \times d_e}$ and relation vector $\mathbf{r} \in \mathbb{R}^{d_r}$ from relation embedding matrix $\mathbf{R} \in \mathbb{R}^{n_r \times d_r}$, where n_e and n_r are number of entities and relations in $\mathcal{KG}$. LowFER projects $\mathbf{e}_s$ and $\mathbf{r}$ into a common space $\mathbb{R}^{kd_e}$ followed by Hadamard product and k-summation pooling, where k is the factorization rank. The output vector $\mathbf{z}$ is then matched against target entity $\mathbf{e}_o$ to give final score.

More formally, for an entity $e \in \mathcal{E}$, we represent its embedding vector $\mathbf{e}$ of d_e dimension as a look-up from entity embedding matrix $\mathbf{E} \in \mathbb{R}^{n_e \times d_e}$, where $n_e = |\mathcal{E}|$. Similarly, for a relation $r \in \mathcal{R}$, we represent its embedding vector $\mathbf{r}$ of d_r dimension as a look-up from relation embedding $\mathbf{R} \in \mathbb{R}^{n_r \times d_r}$, where $n_r = |\mathcal{R}|$. Then, for a given triple (e_s, r, e_o), we define our scoring function as:

$$f(e_s, r, e_o) := \mathbf{g}(e_s, r) \cdot \mathbf{e}_o = \mathbf{g}(e_s, r)^T \mathbf{e}_o \tag{2}$$

where $\mathbf{g}(.,.) \in \mathbb{R}^{d_e}$ is a vector valued function of the subject entity vector $\mathbf{e}_s$ and the relation vector $\mathbf{r}$, defined from Eq. 1 as:

$$\mathbf{g}(e_s, r) := \text{SumPool}(\mathbf{U}^T \mathbf{e}_s \circ \mathbf{V}^T \mathbf{r}, k) \tag{3}$$

where matrices $\mathbf{U} \in \mathbb{R}^{d_e \times kd_e}$ and $\mathbf{V} \in \mathbb{R}^{d_r \times kd_e}$ represent our model parameters. We can re-write the Eq. 3 more compactly as:

$$\mathbf{g}(e_s, r) = \mathbf{S}^k \text{diag}(\mathbf{U}^T \mathbf{e}_s) \mathbf{V}^T \mathbf{r} \tag{4}$$

where $\text{diag}(\mathbf{U}^T \mathbf{e}_s) \in \mathbb{R}^{kd_e \times kd_e}$ and $\mathbf{S}^k \in \mathbb{R}^{d_e \times kde}$ is a constant matrix[1] such that:

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

Using this compact notation in Eq. 2, the final scoring function of LowFER is obtained as:

$$f(e_s, r, e_o) = (\mathbf{S}^k \text{diag}(\mathbf{U}^T \mathbf{e}_s) \mathbf{V}^T \mathbf{r})^T \mathbf{e}_o \tag{5}$$

[1]Note that at $k = 1$, $\mathbf{S}^1 = \mathbf{I}_{d_e}$

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 $k = \min(n_e, n_r)$

3.3. Training LowFER

To train the LowFER model, we follow the setup of Balažević et al. (2019a). First, we apply sigmoid non-linearity after Eq. 5 to get the probability $p(y_{(e_s,r,e_o)}) = \sigma(f(e_s, r, e_o))$ of a triple belonging to a $\mathcal{KG}$. Then, for every triple (e_s, r, e_o) in the dataset, a reciprocal relation is added by generating a synthetic example (e_o, r^{-1}, e_s) (Dettmers et al., 2018; Lacroix et al., 2018) to create the training set $\mathcal{D}$. For faster training, Dettmers et al. (2018) introduced 1-N scoring, where each tuple (e_s, r) and (e_o, r^{-1}) is simultaneously scored against all entities $e \in \mathcal{E}$ to predict 1 if $e = e_o$ or e_s respectively and 0 elsewhere (see Trouillon et al. (2017) and Sun et al. (2019) for other methods to collect negative samples). The model is trained with binary cross-entropy instead of margin based ranking loss (Bordes et al., 2013), which is prone to overfitting for link prediction (Trouillon & Nickel, 2017; Kazemi & Poole, 2018). For a mini-batch $\mathcal{B}$ of size m drawn from $\mathcal{D}$, we minimize:

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

where y_i is a target label for a given entity-relation pair (e, r) for entity e_i, $y_{(e,r,e_i)}$ is the model prediction and Θ represents model parameters. Following Yu et al. (2017), we also apply the power normalization $\mathbf{x} \leftarrow \text{sign}(\mathbf{x})|\mathbf{x}|^{0.5}$ and l_2-normalization $\mathbf{x} \leftarrow \mathbf{x}/||\mathbf{x}||$ before summation pooling to stabilize the training from large output values as a result of Hadamard product in Eq. 3.

4. Theoretical Analysis

4.1. Full Expressibility

A key theoretical property of link prediction models is their ability to be fully expressive, which we define formally as:

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 $r \in \mathcal{R}$.*

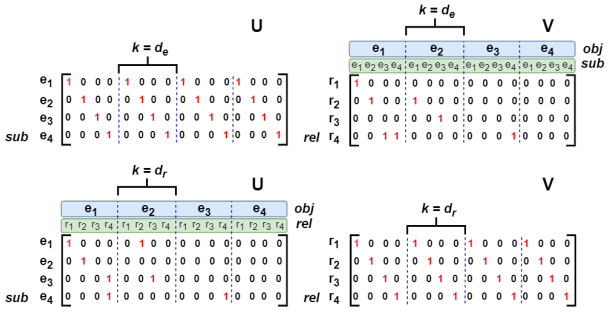

Figure 2. LowFER model parameters for a toy dataset under the settings used in Proposition 1. *Top*: For the case when $k = d_e = n_e$. *Bottom*: For the case when $k = d_r = n_r$.

A *fully expressive* model can represent relations of any type, including *symmetric*, *asymmetric*, *reflexive*, and *transitive* among others. Models such as RESCAL, HolE, ComplEx, SimplE and TuckER have been shown to be *fully expressive* (Trouillon & Nickel, 2017; Wang et al., 2018; Kazemi & Poole, 2018; Balažević et al., 2019a). On the other hand, DistMult is not *fully expressive* as it enforces symmetric relations only. Further, Wang et al. (2018) showed that TransE is not *fully expressive*, which was later expanded by Kazemi & Poole (2018), showing that other translational variants, including, FTransE, STransE, FSTransE, TransR and TransH are likewise not *fully expressive*. By the virtue of universal approximation theorem (Cybenko, 1989; Hornik, 1991), neural networks can be considered *fully expressive* (Kazemi & Poole, 2018). Table 1 summarizes the bounds of linear models that are *fully expressive*. With Proposition 1 (proof in Appendix A.1), we establish that LowFER is *fully expressive* and provide bounds on entity and relation embedding dimensions and the factorization rank k.

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 expressive.*

As a given example, consider a set of entities $\mathcal{E} = \{e_1, e_2, e_3, e_4\}$ and relations $\mathcal{R} = \{r_1, r_2, r_3, r_4\}$ such that r_1 is reflexive, r_2 is symmetric, r_3 is asymmetric, and r_4 is transitive, then for ground truth $\mathcal{T} = \{(e_1, r_1, e_1), (e_1, r_2, e_2), (e_2, r_2, e_1), (e_3, r_3, e_2), (e_4, r_4, e_3), (e_3, r_4, e_1), (e_4, r_4, e_1)\}$ and following the settings in Proposition 1, Figure 2 shows the model parameters $\mathbf{U}$ and $\mathbf{V}$ for this toy example. Now, consider the case $k = d_e = n_e$, then $\mathbf{U}$ copies each entity vector in k-sized slices and $\mathbf{V}$ buckets target entities per relation such that each source entity is distributed into disjoint sets. Note that reshaping $\mathbf{V}$ as 3D tensor of size $n_r \times n_e \times n_e$ and transposing first two dimensions results in binary tensor $\mathbf{T}$.

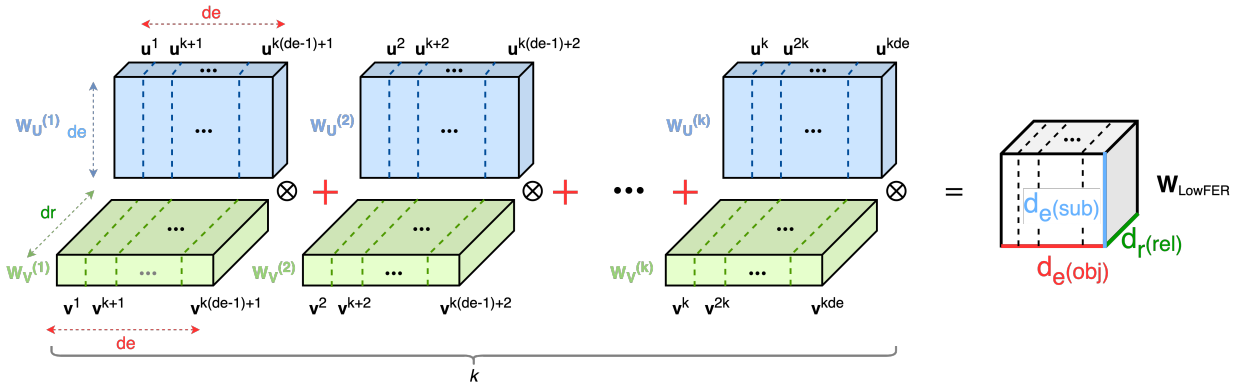

Figure 3. Low-rank approximation of the core tensor $\mathcal{W}$ of TuckER (Balažević et al., 2019a) with LowFER by summing k low-rank 3D tensors, where each tensor is obtained by stacking d_e rank-1 matrices obtained by the outer product of k-apart columns of $\mathbf{U}$ and $\mathbf{V}$.

4.2. Relation with TuckER

Initially, it was shown by Kazemi & Poole (2018) that RESCAL, DistMult, ComplEx and SimplE belong to a *family of bilinear models* with different set of constraints. Later, Balažević et al. (2019a) established that TuckER generalizes all of these models as special cases. In this section, we will formulate relation between our model and TuckER (Balažević et al., 2019a), followed by relations with the *family of bilinear models* in the next section. This provides a unifying view and shows LowFER's ability to generalize.

TuckER's scoring function is defined as follows (Balažević et al., 2019a):

$$\phi_t(e_s, r, e_o) = \mathcal{W} \times_1 \mathbf{e}_s \times_2 \mathbf{r} \times_3 \mathbf{e}_o \tag{6}$$

where $\mathcal{W} \in \mathbb{R}^{d_e \times d_r \times d_e}$ is the core tensor, $\mathbf{e}_s, \mathbf{e}_o \in \mathbb{R}^{d_e}$ and $\mathbf{r} \in \mathbb{R}^{d_r}$ are subject entity, object entity and the relation vectors respectively. $\times_n$ denotes the tensor product along the n-th mode. First, note that Eq. 4 can be expanded as:

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

where $\mathbf{u}_i \in \mathbb{R}^{d_e}$ and $\mathbf{v}_i \in \mathbb{R}^{d_r}$ are column vectors of $\mathbf{U}$ and $\mathbf{V}$ respectively and $\otimes$ represents the outer product of two vectors. To take the vectors $\mathbf{e}_s$ and $\mathbf{r}$ out, we realize the above matrix operations in a different way. We first create k matrices sliced from $\mathbf{U}$ and $\mathbf{V}$ each, such that each matrix is formed by choosing all adjacent column vectors that are k distance apart in $\mathbf{U}$ (and $\mathbf{V}$), i.e., for the l-th slice, we have $\mathbf{W}_U^{(l)} = [\mathbf{u}_l, \mathbf{u}_{k+l}, ..., \mathbf{u}_{k(d_e-1)+l}] \in \mathbb{R}^{d_e \times d_e}$ and $\mathbf{W}_V^{(l)} = [\mathbf{v}_l, \mathbf{v}_{k+l}, ..., \mathbf{v}_{k(d_e-1)+l}] \in \mathbb{R}^{d_r \times d_e}$. Taking the column-wise outer product of these sliced matrices forms a 3D tensor in $\mathbb{R}^{d_e \times d_r \times d_e}$. With slight abuse of notation, we also use $\otimes$ to represent this tensor operation. It can be viewed as transforming the matrix obtained by mode-2 Khatri-Rao product into a 3D tensor (Cichocki et al., 2016). Now consider a 3D tensor $\mathbf{W} \in \mathbb{R}^{d_e \times d_r \times d_e}$ as the sum of these k products:

$$\mathbf{W} = \sum_{i=1}^{k} \mathbf{W}_U^{(i)} \otimes \mathbf{W}_V^{(i)} \tag{7}$$

Figure 3 shows these operations. With this tensor, the scoring function f in Eq. 5 can be re-written as TuckER's scoring function as follows:

$$\hat{\phi}_t(e_s, r, e_o) = \mathbf{W} \times_1 \mathbf{e}_s \times_2 \mathbf{r} \times_3 \mathbf{e}_o \tag{8}$$

It should be noted that $\mathbf{W}$ in Eq. 8 is obtained as a summation of k low-rank 3D tensors, each of which is obtained by stacking rank-1 matrices in contrast to TuckER's core tensor $\mathcal{W}$ in Eq. 6, which can be a full rank 3D tensor. Our model can therefore approximate TuckER and can be viewed as a generalization of TuckER (Balažević et al., 2019a). We further show that we can accurately obtain $\mathcal{W}$ with appropriate $\mathbf{W}_U^{(i)}$'s and $\mathbf{W}_V^{(i)}$'s in Eq. 7 (proof in Appendix A.2).

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 $k <= \min(d_e, d_r)$, entity embedding dimension d_e and relation embedding dimension d_r that accurately represents the former.*

LowFER and TuckER parameters grow linearly in the number of entities and relations as $\mathcal{O}(n_e d_e + n_r d_r)$. However, LowFER's shared parameters complexity can be controlled by decoupled low-rank matrices through the factorization rank, making it more flexible, e.g., consider $d = d_e = d_r$, the core tensor $\mathcal{W}$ of TuckER grows as $\mathcal{O}(d^3)$, whereas LowFER grows only as $\mathcal{O}(kd^2)$. As an example, in Lacroix

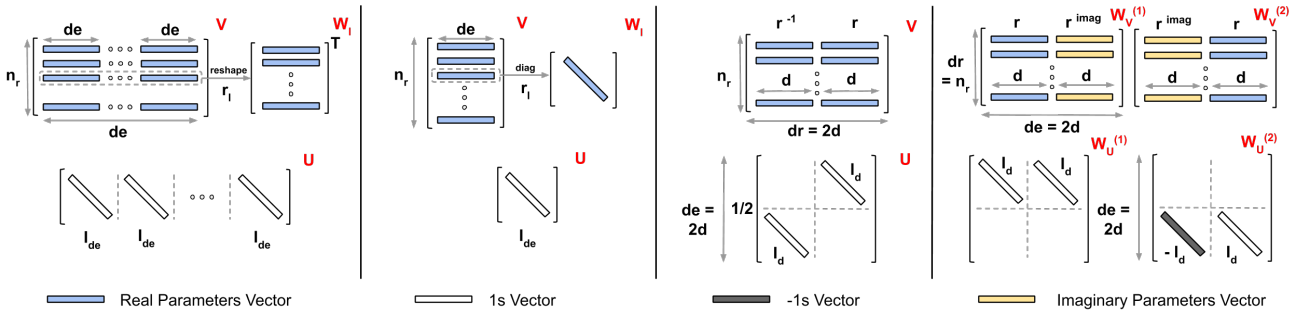

Figure 4. Modeling the family of bilinear models with LowFER, (*from left-to-right*): RESCAL (Nickel et al., 2011), DistMult (Yang et al., 2015), SimplE (Kazemi & Poole, 2018) and ComplEx (Trouillon et al., 2016) (see section 4.3 for details).

et al. (2018) authors used $d_e = d_r = 2000$ which would require more than 8 billion parameters to model with TuckER compared to only $4k$ million for LowFER, with k controlling the growth. More generally, at $k = d_e/2$, LowFER has equal number of parameters as TuckER therefore, we expect similar performance at such rank values. In practice, $k = \{1, 10, 30\}$ performs extremely well (section 5.1).

4.3. Relations with the Family of Bilinear Models

In this section, we will establish relations between LowFER and other bilinear models. For simplicity, we consider the relation embedding to be a constant matrix $\mathbf{R} = \mathbf{I}_{n_r}$ in all the cases and use $\mathbf{V}$ to model relation parameters. However, the conditions presented here can be extended otherwise, with the remark that they are not unique.

RESCAL (Nickel et al., 2011): scoring function is defined as:

$$\phi_r(e_s, r_l, e_o) = \mathbf{e}_s^T \mathbf{W}_l \mathbf{e}_o$$

where $\mathbf{W}_l \in \mathbb{R}^{d_e \times d_e}$ is l-th relation matrix. For LowFER to encode RESCAL with Eq. 5, we set $k = d_e$, $d_r = n_r$ and $\mathbf{U} = [\ \mathbf{I}_{d_e} \mid \mathbf{I}_{d_e} \mid \ldots \mid \mathbf{I}_{d_e}\ ] \in \mathbb{R}^{d_e \times d_e^2}$ (block matrix partitioned as d_e identity matrices of size $d_e \times d_e$). This is effectively taking a row l from $\mathbf{V} \in \mathbb{R}^{n_r \times {d_e}^2}$, reshaping it to $d_e \times d_e$ matrix and then taking the transpose to get the equivalent $\mathbf{W}_l$ in RESCAL's scoring function.

DISTMULT (Yang et al., 2015): scoring function is defined as:

$$\phi_d(e_s, r_l, e_o) = \mathbf{e}_s^T \text{diag}(\mathbf{w}_l) \mathbf{e}_o$$

where $\mathbf{w}_l \in \mathbb{R}^{d_e}$ is the vector for l-th relation. For LowFER to encode DistMult with Eq. 5, we set $k = 1$, $d_r = n_r$ and $\mathbf{U} = \mathbf{I}_{d_e}$. This is effectively taking a row l from $\mathbf{V} \in \mathbb{R}^{n_r \times d_e}$ and creating a diagonal matrix of it to get the equivalent $\text{diag}(\mathbf{w}_l)$ in DistMult's scoring function.

SIMPLE (Kazemi & Poole, 2018): scoring function is defined as:

$$\phi_s(e_s, r_l, e_o) = \frac{1}{2}(\mathbf{h}_{e_s}^T \text{diag}(\mathbf{r}_l) \mathbf{t}_{e_o} + \mathbf{h}_{e_o}^T \text{diag}(\mathbf{r}_l^{-1}) \mathbf{t}_{e_s})$$

where $\mathbf{h}_{e_s}, \mathbf{h}_{e_o} \in \mathbb{R}^d$ are subject, object entities head vectors, $\mathbf{t}_{e_s}, \mathbf{t}_{e_o} \in \mathbb{R}^d$ are subject, object entities tail vectors and $\mathbf{r}_l, \mathbf{r}_l^{-1} \in \mathbb{R}^d$ are relation and inverse relation vectors. Let $\hat{\mathbf{e}}_s = [\mathbf{t}_{e_s}; \mathbf{h}_{e_s}] \in \mathbb{R}^{2d}$, $\mathbf{e}_o = [\mathbf{h}_{e_o}; \mathbf{t}_{e_o}] \in \mathbb{R}^{2d}$ and $\hat{\mathbf{r}}_l = [\mathbf{r}_l^{-1}; \mathbf{r}_l] \in \mathbb{R}^{2d}$ then SimplE scoring is equivalent to $\frac{1}{2}\hat{\mathbf{e}}_s^T \text{diag}(\hat{\mathbf{r}}_l) \mathbf{e}_o$, where $\hat{\mathbf{e}}_s$ and $\hat{\mathbf{r}}_l$ are obtained by swapping the head, tail vectors in $\mathbf{e}_s = [\mathbf{h}_{e_s}; \mathbf{t}_{e_s}]$ and relation, inverse relation vectors in $\mathbf{r}_l = [\mathbf{r}_l; \mathbf{r}_l^{-1}]$ respectively. For LowFER to encode SimplE, $\mathbf{U}$ becomes a permutation matrix (ignoring the $\frac{1}{2}$ scaling factor), swapping the first d-half with the second d-half of a given vector in $\mathbb{R}^{2d}$ and l-th row in $\mathbf{V}$ is $\hat{\mathbf{r}}_l$, more specifically, with Eq. 5, we set $k = 1$, $d_e = 2d$, $d_r = n_r$ and $\mathbf{U} \in \mathbb{R}^{2d \times 2d}$ is a block matrix with four partitions such that, $\mathbf{U}_{12} = \mathbf{U}_{21} = \frac{1}{2}\mathbf{I}_d$ and 0s elsewhere.

COMPLEX (Trouillon et al., 2016) scoring function is defined as:

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

where Re(.) and Im(.) represents the real and imaginary parts of a complex vector. Consider $\hat{\mathbf{e}}_s = [\text{Re}(\mathbf{e}_s); \text{Im}(\mathbf{e}_s)] \in \mathbb{R}^{2d}$ and $\hat{\mathbf{e}}_o = [\text{Re}(\mathbf{e}_o); \text{Im}(\mathbf{e}_o)] \in \mathbb{R}^{2d}$ then the ComplEx scoring function can be obtained as $\hat{\mathbf{e}}_s^T \mathbf{W}_l \hat{\mathbf{e}}_o$, where $\mathbf{W}_l \in \mathbb{R}^{2d \times 2d}$ represents the l-th relation matrix such that its diagonal is $[\text{Re}(\mathbf{r}_l); \text{Re}(\mathbf{r}_l)]$, the d offset diagonal is $\text{Im}(\mathbf{r}_l)$ and $-d$ offset diagonal is $-\text{Im}(\mathbf{r}_l)$. For LowFER to encode ComplEx, similar to SimplE, we will use two permutation matrices to obtain the above four terms. That is, in Eq. 8, we have $k = 2$, $d_e = 2d$, $d_r = n_r$, $\mathbf{U} \in \mathbb{R}^{2d \times 4d}$ is such that $\mathbf{W}_U^{(1)}$ is a block matrix with $\mathbf{W}_{U_{11}}^{(1)} = \mathbf{W}_{U_{12}}^{(1)} = \mathbf{I}_d$ and 0 elsewhere. Further, $\mathbf{W}_U^{(2)}$ is also a block matrix with $\mathbf{W}_{U_{21}}^{(2)} = -\mathbf{I}_d$, $\mathbf{W}_{U_{22}}^{(2)} = \mathbf{I}_d$ and 0 elsewhere. Lastly, $\mathbf{V} \in \mathbb{R}^{n_r \times 4d}$ is such that $\mathbf{W}_V^{(1)}$ row l has $[\text{Re}(\mathbf{r}_l); \text{Im}(\mathbf{r}_l)]$ and $\mathbf{W}_V^{(2)}$ row l has $[\text{Im}(\mathbf{r}_l); \text{Re}(\mathbf{r}_l)]$, i.e.,

Table 2. Link prediction results. Best scores per metric are **boldfaced** and second best underlined.

		WN18RR				FB15k-237				WN18				FB15k			
Linear	Model	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10	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
	Neural LP (Yang et al., 2017)	–	–	–	–	0.250	–	–	0.408	0.940	–	–	0.945	0.760	–	–	0.837
	R-GCN (Schlichtkrull et al., 2018)	–	–	–	–	0.248	0.151	0.264	0.417	0.819	0.697	0.929	**0.964**	0.696	0.601	0.760	0.842
	ConvE (Dettmers et al., 2018)	0.430	0.400	0.440	0.520	0.325	0.237	0.356	0.501	0.943	0.935	0.946	0.956	0.657	0.558	0.723	0.831
	TorusE (Ebisu & Ichise, 2018)	–	–	–	–	–	–	–	–	0.947	0.943	0.950	0.954	0.733	0.674	0.771	0.832
	RotatE (Sun et al., 2019)	–	–	–	–	0.297	0.205	0.328	0.480	–	–	–	–	–	–	–	–
	HypER (Balažević et al., 2019b)	0.465	0.436	0.477	0.522	0.341	0.252	0.376	0.520	0.951	0.947	**0.955**	0.958	0.790	0.734	0.829	0.885
Yes	DistMult (Yang et al., 2015)	0.430	0.390	0.440	0.490	0.241	0.155	0.263	0.419	0.822	0.728	0.914	0.936	0.654	0.546	0.733	0.824
	HolE (Nickel et al., 2016)	–	–	–	–	–	–	–	–	0.938	0.930	0.945	0.949	0.524	0.402	0.613	0.739
	ComplEx (Trouillon et al., 2016)	0.440	0.410	0.460	0.510	0.247	0.158	0.275	0.428	0.941	0.936	0.936	0.947	0.692	0.599	0.759	0.840
	ANALOGY (Liu et al., 2017)	–	–	–	–	–	–	–	–	0.942	0.939	0.944	0.947	0.725	0.646	0.785	0.854
	SimplE (Kazemi & Poole, 2018)	–	–	–	–	–	–	–	–	0.942	0.939	0.944	0.947	0.727	0.660	0.773	0.838
	TuckER (Balažević et al., 2019a)	**0.470**	**0.443**	**0.482**	**0.526**	0.358	**0.266**	0.394	**0.544**	**0.953**	**0.949**	**0.955**	0.958	0.795	0.741	0.833	0.892
	LowFER-1	0.454	0.422	0.470	0.515	0.318	0.233	0.348	0.483	0.949	0.945	0.951	0.956	0.720	0.639	0.774	0.859
	LowFER-10	0.464	0.433	0.477	0.523	0.352	0.261	0.386	0.533	0.950	0.946	0.952	0.958	0.810	0.760	0.843	0.896
	LowFER-k*	0.465	0.434	0.479	**0.526**	**0.359**	**0.266**	**0.396**	**0.544**	0.950	0.946	0.952	0.958	**0.824**	**0.782**	**0.852**	**0.897**

$\mathbf{W}_V^{(2)} = \mathbf{W}_V^{(1)}\mathbf{P}$, where $\mathbf{P} \in \mathbb{R}^{2d \times 2d}$ is the d-half swapping permutation matrix. Figure 4 demonstrates LowFER parameters for the *family of bilinear models* under the conditions discussed in this section.

4.4. Relation to HypER

HypER (Balažević et al., 2019b) is a convolutional model based on *hypernetworks* (Ha et al., 2017), where the relation specific 1D filters are generated by the hypernetwork and convolved with the subject entity vector. Balažević et al. (2019b) showed that it can be understood in terms of tensor factorization up to a non-linearity. With a similar argument, we show that LowFER encodes HypER, bringing it closer to the convolutional approaches as well.

HypER scoring function is defined as (Balažević et al., 2019b):

$$\phi_h(e_s, r, e_o) = h(\text{vec}(\mathbf{e}_s * \mathbf{F}_r)\mathbf{W})\mathbf{e}_o \tag{9}$$

where $\mathbf{F}_r = \text{vec}^{-1}(\mathbf{Hr}) \in \mathbb{R}^{n_f \times l_f}$, $\mathbf{H} \in \mathbb{R}^{n_f l_f \times d_r}$ (hypernetwork), $\mathbf{W} \in \mathbb{R}^{n_f l_m \times d_e}$, vec(.) transforms $n \times m$ matrix to nm-sized vector, $\text{vec}^{-1}(.)$ does the reverse operation, $*$ is the convolution operator, $h(.)$ is ReLU non-linearity and n_f, l_f and $l_m = d_e - l_f + 1$ are *number of filters*, *filter length* and *output length* of convolution. The convolution between a filter and the subject entity embedding can be seen as a matrix multiplication, where the filter is converted to a Toeplitz matrix of size $l_m \times d_e$. With n_f filters, we can realize a 3D tensor of size $n_f \times l_m \times d_e$. Since the filters are generated by the hypernetwork, we have d_r such 3D tensors, resulting in a 4D tensor of size $n_f \times l_m \times d_e \times d_r$ (Balažević et al., 2019b). Without loss of generality, we can view this 4D tensor as a 3D tensor $\mathcal{F} \in \mathbb{R}^{n_f l_m \times d_e \times d_r}$. Taking mode-1 product as $\mathcal{F} \times_1 \mathbf{W}^T$ returns a final tensor $\mathcal{G} \in \mathbb{R}^{d_e \times d_e \times d_r}$. Thus, HypER operations $\text{vec}(\mathbf{e}_s * \mathbf{F}_r)\mathbf{W}$ simplify to $\mathcal{G} \times_3 \mathbf{r} \times_2 \mathbf{e}_s$. At $k = d_e$, with $\mathbf{U} \in \mathbb{R}^{d_e \times d_e^2}$ as block identity matrices (same as in LowFER's relation to RESCAL) and $\mathbf{V} \in \mathbb{R}^{d_r \times de^2}$ set to $\mathbf{G}^T$ ($\mathcal{G}$ viewed as a matrix of size $d_e^2 \times d_r$ and transposed), LowFER's score in Eq. 5 represents HypER, up to the non-linearity.

5. Experiments and Results

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) (see Appendix B for the details, including best hyperparameters and additional experiments).

5.1. Link Prediction

Table 2 shows our main results, where LowFER-1, LowFER-10 and LowFER-k* represent our model for $k = 1$, $k = 10$ and $k =$ best. We choose LowFER-1 and LowFER-10 as baselines. Overall, LowFER reaches competitive performance on all the datasets with state-of-the-art results on FB15k and FB15k-237. On WN18 and WN18RR, TuckER is marginally better than LowFER.

LowFER performs well at low-ranks with significantly less number of parameters compared to other linear models (Table 3). At $k = 1$, it performs better than or on par with both non-linear and linear models (including ComplEx and SimplE) except HypER and TuckER. For FB15k-237, LowFER-1 ($\sim$3M parameters) outperforms R-GCN, RotatE, DistMult and ComplEx by an average of 5.9% on MRR, and it additionally outperforms convolutional models (ConvE, HypER) at $k = 10$ with only +0.8M parameters. On FB15k, the best reported TuckER model is improved upon, with absolute +1.9% increase on toughest Hits@1 metric. This already achieves state-of-the-art with almost half the parameters, $\sim$5.5M in contrast to TuckER's $\sim$11.3M. On WN18RR and WN18, LowFER-1 outperforms all the models excluding TuckER and HypER. With LowFER-k*, we marginally reach state-of-the-art performance on WN18RR and FB15k-237. On FB15k, we reach new state-of-the-art for $\sim$9.51M parameters with +2.9% and +4.1% improvement on MRR and Hits@1.

The empirical gains can be attributed to LowFER's ability to perform *good* fusion between entities and relations while avoiding overfitting through low-rank matrices remaining parameter efficient, with strong performance even at ex-

Table 3. Comparison between the number of parameters in millions (M) of strong linear models. For LowFER-k*, the k values are 10, 100, 30 and 50 for WN18, FB15k-237, WN18RR and FB15k respectively.

Model	WN18	FB15k-237	WN18RR	FB15k
ComplEx	16.4	6.0	16.4	6.5
SimplE	16.4	-	16.4	6.5
TuckER	9.4	11.0	9.4	11.3
LowFER-1	8.2	3.0	8.2	4.6
LowFER-10	8.6	3.8	8.6	5.5
LowFER-k*	8.6	11.3	9.6	9.5

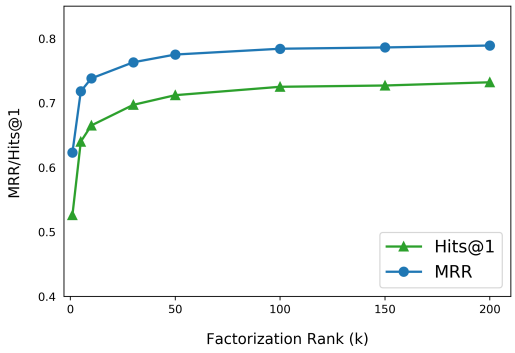

Figure 5. Influence of increasing the factorization rank on MRR and Hits@1 scores for FB15k.

treme low-ranks. Further, like TuckER, it allows for parameter sharing through the $\mathbf{U}$ and $\mathbf{V}$ matrices, unlike ComplEx and SimplE which rely only on embedding matrices.

5.2. Effect of Factorization Rank

From link prediction results, we observe that rank plays an important role depending on the entities-to-relations ratio in the dataset. For $d_e = 200$ and $d_r = 30$, we vary k from $\{1, 5, 10, 30, 50, 100, 150, 200\}$ on FB15k and plot the MRR and Hits@1 scores (Figure 5). From $k = 1$ to $k = 5$, the MRR score increases from 0.62 to 0.72 and Hits@1 increases from 0.53 to 0.64. For higher ranks (after 50), the change is minimal. Empirically, the effect of k diminishes as the number of the entities per relation becomes larger, e.g., it is ~ 3722 for WN18RR in contrast to ~ 11 for FB15k. We suspect that this could be due to the fact that as $n_e \gg d_e$, most of the knowledge is learned through embedding matrices rather than the model parameters $\mathbf{U}$ and $\mathbf{V}$. To test this, we took a trained LowFER model, on WN18 dataset, and added zero mean Gaussian noise with variance in $\{1.0, 1.25, 1.5, 1.75, 2.0\}$ to $\mathbf{U}$ and $\mathbf{V}$ and evaluated on the test set. The MRR score changed from 0.95 to $\{0.92, 0.84, 0.65, 0.42, 0.24\}$ for each level of noise. This shows that in cases as such, the embeddings have potential to capture more knowledge than the shared parameters.

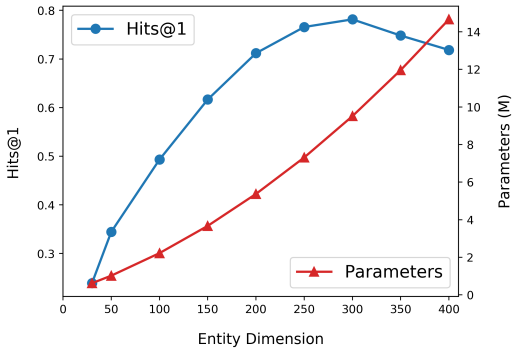

Figure 6. Influence of changing the entity embedding dimension d_e on Hits@1 metric and growth of parameters in million (M).

Table 4. Link prediction results on FB15k with $d_e = d_r = 200$.

k	Params (M)	MRR	Hits@1	Hits@3	Hits@10
1	3.60	0.634	0.538	0.695	0.803
5	3.92	0.720	0.641	0.776	0.860
10	4.33	0.742	0.667	0.790	0.871
30	5.93	0.774	0.709	0.817	0.885
50	7.53	0.776	0.713	0.818	0.886
100	11.53	0.779	0.717	0.821	0.887

Empirically, we found when $d_e = d_r$, taking $k = d_e/2$ performs nearly the same as TuckER (Balažević et al., 2019a). This can be observed in LowFER-k* for FB15k-237 ($d_e = d_r = 200$, $k = 100$), where our results are almost indistinguishable from TuckER's. This can be expected as the number of parameters in both models are almost the same ($\sim$11M). It should be noted that in practice when we train LowFER, we initialize with two i.i.d matrices, which are not shared, compared to TuckER's core tensor (Eq. 6), allowing us to reach almost the same performance despite less parameter sharing.

5.3. Effect of Embedding Dimension

The size of entity embedding dimension d_e accounts for the significant number of parameters in LowFER, growing linearly with number of entities n_e. To study the effect, we trained our models on FB15k, with $d_r = 30$, $k = 50$ constant, and varying de in $\{30, 50, 100, 150, 200, 250, 300, 350, 400\}$. As can be seen in Figure 6, increasing the entity embedding dimension significantly increases the Hits@1 metric, for almost linear growth in number of parameters. However, it only improves till 300 and starts overfitting afterwards.

In Balažević et al. (2019a), authors reported $d_e = d_r = 200$ as best choice of dimensions for TuckER on FB15k, however, we found using $de = 300$ and $d_r = 30$ better with lesser number of parameters for LowFER. For fair compari-

Table 5. Link prediction results on FB15k with $d_e = 200$, $d_r = 50$, $k = 150$ and l_2-regularization 0.0005.

Model	Params (M)	MRR	Hits@1	Hits@3	Hits@10
TuckER	11.3	0.795	0.741	0.833	0.892
LowFER-k*	10.6	0.795	0.739	0.831	0.891
LowFER-k* + Reg	10.6	0.802	0.749	0.837	0.892

son, we also provide the results for $d_e = d_r = 200$ for k in $\{1, 5, 10, 30, 50, 100\}$ in Table 4. As k is increased, we see an improvement over all the metrics. At $k = 100$, where we expected LowFER to match TuckER's performance (MRR=0.795, Hits@1=0.741, ∼11 million parameters), it was lower (-1.6% on MRR and -2.4% on Hits@1). In comparison, our model with $d_e = 300$, $d_r = 30$ and $k = 10$ with ∼5.6 million parameters only, gives better results than this setting and TuckER. Therefore, at $d_e = d_r = 200$, our model is most likely overfitting.

As noted above that it could be that LowFER is overfitting therefore, we did coarse grid search over relation embedding dimension in $\{30, 50, 100, 150, 200\}$ and k in $\{1, 5, 10, 30, 50, 100, 150, 200\}$ while keeping $d_e = 200$ fixed. We found $d_r = 50$ at $k = 150$ reaches almost the same performance as TuckER with ∼10.6M parameters compared to TuckER's ∼11.3M parameters. We also experimented with l_2-regularization (Reg) and noted minor improvements, with regularization strength 0.0005. Table 5 summarizes these results. Note that all the experiments reported in main results (Table 2) were without any regularization. In general, we only noticed slight improvements in FB15k with l_2-regularization.

5.4. Analysis of Relation Results

Link prediction models that can discover relation types automatically without prior knowledge indicate *better* generalization. As shown, and discussed in section 4, LowFER, among other models (Table 1), can learn to capture all relation types without additional constraints. However, in practice, these bounds are loose and require very large dimensions, raising an inspection into their performance on different relation types. In Kazemi & Poole (2018), it was identified that WN18 contains redundant relations, i.e., $\forall e_i, e_j \in \mathcal{E} : (e_i, r_1, e_j) \in \mathcal{T} \Leftrightarrow (e_j, r_2, e_i) \in \mathcal{T}$, such as <*hyponym*, *hypernym*>, <*meronym*, *holonym*> etc. To alleviate this, Dettmers et al. (2018) proposed WN18RR with such relations removed, since knowledge about one can help infer the knowledge about the other. Table 6 shows the per relation results of LowFER and TuckER on WN18 and WN18RR. We see that performance drops for 7 relations, with an average performance decrease of -70.6% and -69.3% for LowFER and TuckER respectively (with highest decrease on *member_of_domain_usage* for both). For symmetric relations (such as *derivationally_related_form*), the performance is approximately the same where we observe severe limitation to model asymmetry. We believe this is because LowFER (also TuckER) is constraint-free and adding certain constraints based on background knowledge is *necessary* to improve the model's accuracy. SimplE is the only *fully expressive* model that has formally shown to address these limitations (cf. Proposition 3, 4 and 5 in Kazemi & Poole (2018)). Since LowFER subsumes SimplE therefore, such rules can be studied for extending LowFER to incorporate the background knowledge.

Table 6. Relation specific test set results on WN18 and WN18RR with LowFER-k* and best reported TuckER model (Balažević et al., 2019a).

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

6. Conclusion

This work proposes a simple and parameter efficient *fully expressive* linear model that is theoretically well sound and performs on par or state-of-the-art in practice. We showed that LowFER generalizes to other linear models in KGC, providing a unified theoretical view. It offers a strong baseline to the deep learning based models and raises further interest into the study of linear models. We also highlighted some limitations with respect to gains on harder relations, which still pose a challenge. We conclude that the constraint-free and parameter efficient linear models, which allow for parameter sharing, are better from a modeling perspective, but are still similarly limited in learning difficult relations. Therefore, studying the trade-off between parameters sharing and constraints becomes an important future work.

Acknowledgements

The authors would like to thank the anonymous reviewers for helpful feedback and gratefully acknowledge the use of code released by Balažević et al. (2019a). The work was partially funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No. 777107 through the project Precise4Q and by the German Federal Ministry of Education and Research (BMBF) through the project DEEPLEE (01IW17001).

References

Balažević, I., Allen, C., and Hospedales, T. TuckER: Tensor Factorization for Knowledge Graph Completion. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 5188–5197, 2019a.

Balažević, I., Allen, C., and Hospedales, T. M. Hypernetwork knowledge graph embeddings. In *International Conference on Artificial Neural Networks*, pp. 553–565. Springer, 2019b.

Ben-Younes, H., Cadene, R., Cord, M., and Thome, N. MUTAN: Multimodal Tucker Fusion for Visual Question Answering. In *International Conference on Computer Vision*, 2017.

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.

Carroll, J. D. and Chang, J.-J. Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition. *Psychometrika*, 35(3):283–319, 1970.

Charikar, M., Chen, K., and Farach-Colton, M. Finding frequent items in data streams. *Theoretical Computer Science*, 312(1):3–15, 2004.

Cichocki, A., Lee, N., Oseledets, I., Phan, A.-H., Zhao, Q., Mandic, D. P., et al. Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions. *Foundations and Trends® in Machine Learning*, 9(4-5):249–429, 2016.

Cybenko, G. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

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. *arXiv preprint arXiv:1711.05851*, 2017.

De Lathauwer, L., De Moor, B., and Vandewalle, J. A multilinear singular value decomposition. *SIAM journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.

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 *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Feng, J., Huang, M., Wang, M., Zhou, M., Hao, Y., and Zhu, X. Knowledge graph embedding by flexible translation. In *Fifteenth International Conference on the Principles of Knowledge Representation and Reasoning*, 2016.

Fukui, A., Park, D. H., Yang, D., Rohrbach, A., Darrell, T., and Rohrbach, M. Multimodal Compact Bilinear Pooling for Visual Question Answering and Visual Grounding. In *Conference on Empirical Methods in Natural Language Processing*, pp. 457–468. ACL, 2016.

Gao, Y., Beijbom, O., Zhang, N., and Darrell, T. Compact bilinear pooling. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 317–326, 2016.

Ha, D., Dai, A., and Le, Q. V. Hypernetworks. In *International Conference on Learning Representations*, 2017.

Harshman, R. A. Models for analysis of asymmetrical relationships among N objects or stimuli. In *First Joint Meeting of the Psychometric Society and the Society of Mathematical Psychology, Hamilton, Ontario, 1978*, 1978.

Harshman, R. A. and Lundy, M. E. PARAFAC: Parallel factor analysis. *Computational Statistics & Data Analysis*, 18(1):39–72, 1994.

Hayashi, K. and Shimbo, M. On the equivalence of holographic and complex embeddings for link prediction. *arXiv preprint arXiv:1702.05563*, 2017.

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.

Hornik, K. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

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

Ji, G., He, S., Xu, L., Liu, K., and Zhao, J. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 687–696, 2015.

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

Kim, J.-H., On, K.-W., Lim, W., Kim, J., Ha, J.-W., and Zhang, B.-T. Hadamard product for low-rank bilinear pooling. *arXiv preprint arXiv:1610.04325*, 2016.

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

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

Li, Y., Wang, N., Liu, J., and Hou, X. Factorized bilinear models for image recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2079–2087, 2017.

Lin, Y., Liu, Z., Luan, H., Sun, M., Rao, S., and Liu, S. Modeling relation paths for representation learning of knowledge bases. *arXiv preprint arXiv:1506.00379*, 2015.

Liu, H., Wu, Y., and Yang, Y. Analogical inference for multi-relational embeddings. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2168–2178. JMLR. org, 2017.

Liu, Z., Shen, Y., Lakshminarasimhan, V. B., Liang, P. P., Zadeh, A., and Morency, L.-P. Efficient low-rank multimodal fusion with modality-specific factors. *arXiv preprint arXiv:1806.00064*, 2018.

Mahdisoltani, F., Biega, J., and Suchanek, F. M. Yago3: A knowledge base from multilingual wikipedias. 2013.

Miettinen, P. Boolean tensor factorizations. In *2011 IEEE 11th International Conference on Data Mining*, pp. 447–456. IEEE, 2011.

Nguyen, D. Q., Sirts, K., Qu, L., and Johnson, M. Stranse: a novel embedding model of entities and relationships in knowledge bases. *arXiv preprint arXiv:1606.08140*, 2016.

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.

Nickel, M., Rosasco, L., and Poggio, T. Holographic embeddings of knowledge graphs. In *Thirtieth AAAI conference on artificial intelligence*, 2016.

Pereyra, G., Tucker, G., Chorowski, J., Kaiser, Ł., and Hinton, G. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*, 2017.

Pham, N. and Pagh, R. Fast and scalable polynomial kernels via explicit feature maps. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 239–247, 2013.

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*, pp. 593–607. Springer, 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*, pp. 6786–6797, 2018.

Sun, Z., Deng, Z.-H., Nie, J.-Y., and Tang, J. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*, 2019.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

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

Trouillon, T. and Nickel, M. Complex and holographic embeddings of knowledge graphs: a comparison. *arXiv preprint arXiv:1707.01475*, 2017.

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

Trouillon, T., Dance, C. R., Gaussier, É., Welbl, J., Riedel, S., and Bouchard, G. Knowledge graph completion via complex tensor factorization. *The Journal of Machine Learning Research*, 18(1):4735–4772, 2017.

Tucker, L. R. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

Wang, Y., Gemulla, R., and Li, H. On multi-relational link prediction with bilinear models. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Wang, Z., Zhang, J., Feng, J., and Chen, Z. Knowledge graph embedding by translating on hyperplanes. In *Twenty-Eighth AAAI conference on artificial intelligence*, 2014.

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*, pp. 2319–2328, 2017.

Yu, Z., Yu, J., Fan, J., and Tao, D. Multi-modal Factorized Bilinear Pooling with Co-attention Learning for Visual Question Answering. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 1839–1848. IEEE, 2017.