Embedding Mapping Approaches for Tensor Factorization and Knowledge Graph Modelling

Yinchong Yang[2], Cristóbal Esteban[12], and Volker Tresp[12]

[1] Siemens AG, Corporate Technology, Munich, Germany
[2] Ludwig-Maximilians-Universität München, Munich, Germany

Abstract. Latent embedding models are the basis of state-of-the art statistical solutions for modelling Knowledge Graphs and Recommender Systems. However, to be able to perform predictions for new entities and relation types, such models have to be retrained completely to derive the new latent embeddings. This could be a potential limitation when fast predictions for new entities and relation types are required. In this paper we propose approaches that can map new entities and new relation types into the existing latent embedding space without the need for retraining. Our proposed models are based on the observable —even incomplete— features of a new entity, e.g. a subset of observed links to other known entities. We show that these mapping approaches are efficient and are applicable to a wide variety of existing factorization models, including nonlinear models. We report performance results on multiple real-world datasets and evaluate the performances from different aspects.

1 Introduction

Latent embedding models, aka *factorization models*, have proven to be powerful approaches for modelling Knowledge Graphs (KG) as described in [18] and [17]. A special case is Collaborative Filtering (CF) where latent embedding models have shown state-of-the-art performance [16]. The common key aspect of these models is that an observed link between multiple entities can be modelled as the interaction between their latent embedding vectors. Multi-linear models such as CP/PARAFAC [14] and Tucker [22] as well as RESCAL [19] are typical examples of models that use latent embeddings. Nonlinear Neural Network-based embedding models are derived in [8] and [20]. For a more detailed review of these works please see [18].

The latent embedding vectors can be used in several ways. For example, it has been shown that distances between entities in the latent space are more compact and meaningful than in the original observable feature space. Also, in entity resolution, entities close to each other in the latent space can sometimes be interpreted as duplicates [7]. Finally, it has been shown that unknown links between known entities can be predicted based on interactions of their latent embeddings [18].

A drawback of latent embedding models is that they need to be retrained when new entities are appended to the database. For large-scale databases and/or

situations where the system is expected to perform immediate operations, such as entity resolution or link prediction on the new entities, this would be very costly and factorization models would find only limited applications.

In this paper, we propose a new class of approaches to handle new entities and new relation types by mapping them into the latent space learned by the factorization model. We emphasize that such mapping models can be learned in conjunction with the training of the factorization model. To map a new entity into the latent space we only require the observable features of the entity. In a KG, for instance, such observable features form a binary vector or matrix, representing the existence of links between this a entity and a subset of known entities in the database.

The rest of the paper is organized as follows: In Section 2 we give a brief review of selected embedding-based factorization models and illustrate the concept of an embedding mapping. We show that for certain specific factorization models there exist embedding mappings in closed form. In Section 3, we propose a general framework that describes a variety of factorization models on a more abstract level and derive a framework defining the mapping models and elaborate three options for training. In Section 4 we present experimental results on real-world datasets. Section 5 discusses related work and Section 6 contains conclusions and an outlook for further works.

Notations: A matrix $\boldsymbol{A}$ is represented as a bold capital letter and a multidimensional tensor $\boldsymbol{\mathcal{X}}$ by a calligraphic bold capital letter. By default we assume a 3-dimensional tensor. In some applications the dimensions correspond to entities and relation types, which we sometimes treat as generalized entities. A matrix with indexing superscript as $\boldsymbol{A}^{(l)}$ denotes the latent embedding matrix for entities of the l-th dimension of a matrix or tensor. The matrix derived by unfolding a tensor w.r.t. dimension l is noted using subscripts as $\boldsymbol{X}_{(l)}$. Note that unfolding a matrix w.r.t. first and second dimension is equivalent to the matrix itself and its transpose, respectively. $\boldsymbol{X}^{\dagger}$ stands for the Moore–Penrose pseudoinverse. A vector is denoted with bold small letters such as $\boldsymbol{x}_{i,\bullet}$ and refers to the i-th row in a corresponding matrix $\boldsymbol{X}$. We refer to a set using either a simple capital Greek letter such as Θ or —if we focus on the elements— using curly brackets as $\{\boldsymbol{A}^{(l)}\}_{l=1}^{L}$. The concatenation operation is noted with squared brackets $[\bullet, ..., \bullet]_{+}$.

2 Factorization Models with Closed-Form Mappings

In this section we review a few well-studied factorization models that are based on latent embeddings and motivate our problem setting of mapping new entities into the latent embedding space.

Matrix Cases: First we review the Singular Value Decomposition(SVD) as a latent embedding model: For an SVD in form of $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^{T}$ we interpret the matrix $\boldsymbol{U}$ to consist of latent embedding vectors in rows, for each entity represented in the first dimension of $\boldsymbol{X}$. The matrix $\boldsymbol{X}$ is constructed by a linear combination of the embeddings $\boldsymbol{U}$ with weights defined as rows of $(\boldsymbol{D}\boldsymbol{V}^{T})^{T}$.

Then we consider $\boldsymbol{U} = \boldsymbol{X}(\boldsymbol{D}\boldsymbol{V}^T)^\dagger$, which is an inverse-relation, to be a mapping function from $\boldsymbol{X}$ to the latent embedding in $\boldsymbol{U}$. It is generally assumed that this mapping relation also holds for a new observation which is not present in $\boldsymbol{X}$, i.e.

$$\boldsymbol{u}_{new}^T = \boldsymbol{x}_{new}^T(\boldsymbol{D}\boldsymbol{V}^T)^\dagger \tag{1}$$

given that $\boldsymbol{D}$ and $\boldsymbol{V}$ are regarded as constant. We can generalize these relationships to Matrix Factorization(MF) $\boldsymbol{X} = \boldsymbol{A}\boldsymbol{B}^T$ as used in [16]. The latent embeddings are now rows of $\boldsymbol{A}$ and the weights as rows of $\boldsymbol{B}$. The mapping function now is

$$\boldsymbol{a}_{new}^T = \boldsymbol{x}_{new}^T(\boldsymbol{B}^T)^\dagger. \tag{2}$$

In both cases (SVD and MF), instead of a complete recalculation of the factorization to derive the corresponding latent embedding vector, we simply need to apply a linear map to $\boldsymbol{x}_{new}$, where the map is derived from the pseudo-inverse operation.

Tensor Cases: Following the notation in [15], we describe the CP/PARAFAC model [14] as well as its more general form, the Tucker decomposition [22], as $\boldsymbol{\mathcal{X}} \approx \boldsymbol{\mathcal{G}} \times_1 \boldsymbol{A} \times_2 \boldsymbol{B} \times_3 \boldsymbol{C}$. A row in each of the three matrices, i.e., $\boldsymbol{a}_{i,\bullet}, \boldsymbol{b}_{j,\bullet}, \boldsymbol{c}_{k,\bullet}$, stores the latent embedding of the i-, j- and k-th entity, respectively, in the corresponding dimensions of $\boldsymbol{\mathcal{X}}$; and the core tensor $\boldsymbol{\mathcal{G}}$ specifies the linear interaction between each triple of embedding vectors to derive the entry $x_{i,j,k}$. In the special case of CP, $\boldsymbol{\mathcal{G}}$ takes the form of a hyper-diagonal tensor. By rewriting the model as $\boldsymbol{X}_{(1)} = \boldsymbol{A}\boldsymbol{G}_{(1)}(\boldsymbol{C} \otimes \boldsymbol{B})^T$ we could also interpret $\boldsymbol{A}$ as a latent embedding matrix and $\boldsymbol{G}_{(1)}(\boldsymbol{C} \otimes \boldsymbol{B})^T$ as the linear weights. Inverting this linear relation we can obtain a mapping of the form

$$\boldsymbol{a}_{new}^T = \boldsymbol{x}_{new}^T(\boldsymbol{G}_{(1)}(\boldsymbol{C} \otimes \boldsymbol{B})^T)^\dagger. \tag{3}$$

Such closed-form mappings cannot be derived in at least two cases: Firstly, for non-linear factorization models such as Multiway Neural Network (mwNN) [8], Neural Tensor Networks [20] and TransE [3]; secondly for models with shared embeddings such as RESCAL [19]. We derive solutions for those two cases in the next section. In the experimental part of this paper, we implement and test (logistic) MF to model data in matrix form and CP, RESCAL and mwNN for tensor data. A brief summary of the model architectures can be found in Table 1. To obtain proper probabilistic models, we introduce a natural parameter η for each entry x in the matrix or tensor.

3 General Models and Training Algorithms

In this section we introduce a generic framework describing factorization and **Em**bedding **Ma**pping (abbreviately termed 'Emma'**).** We also propose three novel approaches to train the mapping models. For the sake of generality we shall refer to matrices also as tensors.

Model	Distr. Assumption	Natural Parameter
MF	$x_{i,j} \sim \mathcal{N}(\eta_{i,j}, \sigma)$	$\hat{\eta}_{i,j} = \sum_{r=1}^{R} a^{(1)}_{i,r} \cdot a^{(2)}_{j,r}$
logistic MF	$x_{i,j} \sim \mathcal{B}(sig(\eta_{i,j}))$	
CP	$x_{i,j,k} \sim \mathcal{N}(\eta_{i,j,k}, \sigma)$	$\hat{\eta}_{i,j,k} = \sum_{r=1}^{R} a^{(1)}_{i,r} \cdot a^{(2)}_{j,r} \cdot a^{(3)}_{k,r}$
RESCAL	$x_{i,j,k} \sim \mathcal{N}(\eta_{i,j,k}, \sigma)$	$\hat{\eta}_{i,j,k} = \sum_{p}^{P} \sum_{p'}^{P} w_{p,p',k} \cdot a^{(1)}_{i,p} \cdot a^{(1)}_{j,p'}$
mwNN	$x_{i,j,k} \sim \mathcal{B}(sig(\eta_{i,j}))$	$\hat{\eta}_{i,j,k} = MLP([\boldsymbol{a}^{(1)}_{i,\bullet}, \boldsymbol{a}^{(2)}_{j,\bullet}, \boldsymbol{a}^{(3)}_{k,\bullet}]_+)$

Table 1. A summary of selected factorization models within the scope of this paper. *sig* denotes the sigmoid or logistic function $sig(x) = \frac{1}{1+\exp(-x)}$; $\mathcal{N}$ and $\mathcal{B}$ denote Gaussian and Bernoulli distributions, respectively. We denote with MLP a standard three layered perceptron.

3.1 General Models

The Factorization Model defines the interaction between latent embeddings to construct the tensor as

$$\begin{aligned} \widehat{\boldsymbol{\mathcal{H}}} &= f\left(\{\boldsymbol{A}^{(l)}\}_{l=1}^{L};\ \Theta\right) \quad \text{with} \\ \widehat{\boldsymbol{\mathcal{H}}} &\in \mathbb{R}^{n_1 \times \ldots \times n_L}, \boldsymbol{A}^{(l)} \in \mathbb{R}^{n_l \times p_l}. \end{aligned} \tag{4}$$

Here, the L-dimensional tensor $\boldsymbol{\mathcal{H}}$ contains the natural parameters η. The matrices in set $\{\boldsymbol{A}^{(l)}\}_{l=1}^{L}$ store the latent embeddings in their rows. The tensor is reconstructed with operations defined by a parameterized function $f(\bullet;\ \Theta)$.

For instance, in case of CP factorization, the function f specifies the linear combination of rows in each embedding matrix without additional parameters ($\Theta = \emptyset$); and for mwNN, Θ consists of the weights in an NN model whose architecture is defined as part of f.

The Factorization Cost Function: We define the factorization cost function c_F and its regularized version c_F^p as

$$c_F = d_F(\boldsymbol{\mathcal{X}}, \widehat{\boldsymbol{\mathcal{H}}}) = d_F(\boldsymbol{\mathcal{X}}, f(\{\boldsymbol{A}^{(l)}\}_{l=1}^{L};\ \Theta)) \tag{5}$$

$$c_F^p = c_F + \sum_{l=1}^{L} \gamma(\boldsymbol{A}^{(l)}) + \rho(\Theta). \tag{6}$$

During training the cost function is optimized w.r.t. the parameters in Θ and embeddings in the $\boldsymbol{A}^{(l)}$'s. An example for the differentiable distance measure d_F is the Frobenius Norm. For binary tensors, the cross-entropy loss is more suitable. In Equation (6) we included a regularization function $\gamma(\bullet)$ for the latent embeddings and a second one $\rho(\bullet)$ for the model parameters.

The Mapping Model defines a function $g(\bullet;\ \Psi_l)$ that maps each row in the tensor unfolding $\boldsymbol{X}_{(l)}$ to the corresponding row in the learned embedding matrix $\boldsymbol{A}^{(l)}$ as

$$\begin{aligned} \widehat{\boldsymbol{A}}^{(l)} &= g\left(\boldsymbol{X}_{(l)};\ \Psi_l\right) \quad \forall l \in [1, ..., L] \quad \text{with} \\ \boldsymbol{X}_{(l)} &\in \mathbb{R}^{n_l \times \prod_{l' \neq l} n_{l'}}. \end{aligned} \tag{7}$$

Note that in the input of the mapping function, each arbitrary row i in $\boldsymbol{X}_{(l)}$, is identical to the vectorized i-th hyper-slice of the tensor and consists of all available information about the i-th entity. For a KG, for instance, this could be the vector indicating the existence of relations between entity i and all other entities for all relation types. The function $g(\bullet)$ defines the architecture of the mapping model and Ψ_l consists of all parameters.

The Mapping Cost Function: We define the mapping cost function as

$$c_M = \sum_{l=1}^{L} d_M(\boldsymbol{A}^{(l)}, \widehat{\boldsymbol{A}}^{(l)}) = \sum_{l=1}^{L} d_M(\boldsymbol{A}^{(l)}, g(\boldsymbol{X}_{(l)};\ \Psi_l)) \tag{8}$$

$$c_M^p = c_M + \sum_{l=1}^{L} \rho(\Psi_l). \tag{9}$$

Optimizing the mapping cost function involves adjusting Ψ_l for each l with a given $g(\bullet)$ so that the distance between the learned embedding $\boldsymbol{A}^{(l)}$ from factorization and mapped embedding $\widehat{\boldsymbol{A}}^{(l)}$ from the corresponding tensor unfoldings is minimized.

The Compact Model: Since the latent embeddings, e.g., the $\boldsymbol{A}^{(l)}$'s, are also adjustable parameters, we could write a more compact model by plugging Equation (7) into Equation (4) and obtain

$$\widehat{\boldsymbol{\mathcal{H}}} = f\left(\left\{g\left(\boldsymbol{X}_{(l)};\ \Psi_l\right)\right\}_{l=1}^{L};\ \Theta\right). \tag{10}$$

Analogously, combining cost functions of the factorization model —Equations (5), (6)— and those of the mapping model — Equations (8), (9)— we obtain:

The Compact Cost Function as:

$$c = d_F(\boldsymbol{\mathcal{X}}, \widehat{\boldsymbol{\mathcal{H}}}) = d_F\left(\boldsymbol{\mathcal{X}}, f\left(\left\{g\left(\boldsymbol{X}_{(l)};\ \Psi_l\right)\right\}_{l=1}^{L};\ \Theta\right)\right) \tag{11}$$

$$c^p = c + \rho(\Theta) + \sum_{l=1}^{L} \rho(\Psi_l), \tag{12}$$

where $\boldsymbol{A}^{(l)}$'s are not explicitly defined but could be derived from $g(\boldsymbol{X}_{(l)};\ \Psi_l)$.

It should be noted that the tensor $\boldsymbol{\mathcal{X}}$ occurs both at the output and the input (as unfoldings) of the cost function. More specifically, for a tensor $\boldsymbol{\mathcal{X}}$ of three dimensions, the factorization and mapping models as a whole actually model each entry $x_{i,j,k}$ based on the i-th, j-th and k-th slices of 1st, 2nd and the 3rd dimension of the tensor, respectively, whereby the $x_{i,j,k}$ is the intersection of all these three slices. This aspect is illustrated in Figure 1.

The factorization problem defined in Equation (4) could be solved using a variety of well studied factorization models that optimize Equation (5). In the following we focus on solving the mapping problem in Equation (7) and the

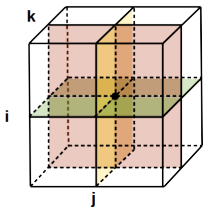

Fig. 1. Illustration of the Compact Model in case of a tensor: We model the each entry $x_{i,j,k}$ over the natural parameter $\eta_{i,j,k}$ based on the three slices of the tensor indexed by i, j and k, respectively.

compact modelling problem in Equation (10). Within the scope of this paper we specifically assume that the function $g(\bullet;\ \Psi_l)$ represents a linear relation between $\boldsymbol{A}^{(l)}$ and $\boldsymbol{X}_{(l)}$ that can be modelled by a matrix $\Psi_l = \{\boldsymbol{M}_l\}$, i.e.

$$\widehat{\boldsymbol{A}}^{(l)} = \boldsymbol{X}_{(l)}\boldsymbol{M}_l \ \forall l \in [1, ..., L]. \tag{13}$$

3.2 Training Approaches

Post Embedding Mapping: The most intuitive way to perform an Emma is to solve Equations (4) and (7) sequentially: Given that a certain factorization model has already derived the latent embeddings $\boldsymbol{A}^{(l)}$, we could consider the mapping from $\boldsymbol{X}_{(l)}$ to the embedding to be a linear system of n_l equations as suggested in Equation (7). Since one is interested in small dimensions for the latent embeddings, i.e. $p_l < \prod_{l' \neq l} n_{l'}$, the system is overdetermined and can be approximately solved using Least-Square (LS) methods. Specifically: $\widehat{\boldsymbol{M}}_l = (\boldsymbol{X}_{(l)}^T\boldsymbol{X}_{(l)})^\dagger \boldsymbol{X}_{(l)}^T\boldsymbol{A}^{(l)}$.

It is easy to see that the inverting methods of Equations (1), (2) and (3) introduced in Section 2 are special cases of this LS estimation. For instance, with MF model $\boldsymbol{X} = \boldsymbol{A}\boldsymbol{B}^T$, the general LS estimation could be described as $\widehat{\boldsymbol{A}} = \boldsymbol{X}\boldsymbol{M}_A$ with $\boldsymbol{M}_A = (\boldsymbol{X}^T\boldsymbol{X})^\dagger\boldsymbol{X}^T\boldsymbol{A}$. Plugging in the information of the model definition, we obtain $\boldsymbol{M}_A = (\boldsymbol{B}\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{B}^T)^\dagger\boldsymbol{A}\boldsymbol{B}^T\boldsymbol{A} = (\boldsymbol{B}^T)^\dagger$, which is the same as the inverting operation in Equation (2). An apparently desirable feature of this Post Mapping approach is its simplicity. It is applicable for any known factorization model and does not affect the factorizing process, since this approach assumes the learned embeddings as fixed. In the following we shall refer to this approach as Emma-Post.

Embedding Mapping Learning with Hatting Algorithm: Alternatively, one could also integrate the Emma learning into the factorization learning process: Instead of solving the LS problem after the factorization learning is completed, we suggest to fit the LS solution against the current latent embedding after each epoch of the factorization learning. Specifically, after each epoch of the factorization learning, we solve the LS problem based on the current embeddings and replace these with their LS estimates to satisfy the criterion of Equation (8). In terms of notation, we replace $\boldsymbol{A}^{(l)}$ with $\widehat{\boldsymbol{A}}^{(l)} = \boldsymbol{H}^{(l)}\boldsymbol{A}^{(l)}$ i.e. we 'hat' the embedding matrix with the Hat-Matrix as in linear regression [13]. In the next epoch, the factorization algorithm proceeds from the LS estimates of the embeddings. In addition, to avoid collinearity and overfitting, it is also advisable to add a ridge regularization term to the Hat Matrix. We formulate this

Algorithm 1. Hatting Algorithm Framework

for all $l \in [1, ..., L]$ **do**:
  $\boldsymbol{U}^{(l)} \leftarrow (\boldsymbol{X}_{(l)}^T \boldsymbol{X}_{(l)} - \lambda \boldsymbol{I})^\dagger \boldsymbol{X}_{(l)}^T$
  $\boldsymbol{H}^{(l)} \leftarrow \boldsymbol{X}_{(l)} \boldsymbol{U}^{(l)}$
end for
for each epoch t in learning factorization **do**:
  $\{\boldsymbol{A}^{(l)}\}_{l=1}^L, \Theta \leftarrow$ update w.r.t. c_F^p

Absolute Updating:	*or*	*Stochastic Updating with Late-Starting:*
		if $t > \tau$ **then**:
for $l \in [1, ..., L]$ **do**:		**for** $l \in [1, ..., L]$ **do**:
		$\pi^{(l)} = 1 - \max(0, R^2(\boldsymbol{A}^{(l)}, \boldsymbol{H}^{(l)}\boldsymbol{A}^{(l)}))$
$\boldsymbol{A}^{(l)} \leftarrow \boldsymbol{H}^{(l)}\boldsymbol{A}^{(l)}$		$\boldsymbol{A}^{(l)} \leftarrow \boldsymbol{H}^{(l)}\boldsymbol{A}^{(l)}$ with probability $\pi^{(l)}$
end for		**end for**
		end if

end for
return:
  $\{\boldsymbol{A}^{(l)}\}_{l=1}^L$ as latent embeddings;
  $\{\boldsymbol{M}^{(l)} = \boldsymbol{U}^{(l)}\boldsymbol{A}^{(l)}\}_{l=1}^L$ as Emma matrices.

integrated Emma as an algorithmic framework in Algorithm 1 with the left-sided option termed *Absolute Updating*.

There are two major advantages of this algorithmic framework: Firstly, the LS updates are efficient to calculate since the Hat Matrix is only calculated once prior to the iterative factorization learning, during which one only needs to perform in each epoch one matrix multiplication for each dimension for the sake of LS-updating. Secondly, any factorization algorithm can be easily extended with this LS update as long as it is performed in a iterated manner, such as when using ALS or gradient based approaches.

Based on our experiments we also propose following practical adjustments of the algorithm:

i. Late Starting Strategy: Since the embeddings are usually initialized randomly by many algorithms, it is not necessary in such cases to perform LS updates during early epochs. It is advisable to start LS updates, for instance, when the embeddings are updated in smaller magnitudes i.e. where the cost function is locally flat. One could measure the gradient changes in each epoch or simply define a $\tau > 1$ to be the first iteration where the LS update commences.

ii. Stochastic Update: We suggest monitoring the quality of the linear mapping in terms of R^2, the Coefficient of Determination. As long as R^2 is small, it is always assumed to be necessary to further perform LS update. But as the training proceeds, R^2 often tends to get larger and converges to 1. In such cases it may again be unnecessary to perform an LS update in each iteration and one could save some runtime. Since the R^2 typically lies within $(0, 1)$ but could also be negative based on our definition, we define a coefficient $\pi = 1 - \max(0, R^2)$ to be interpreted as the probability or necessity to perform an LS update. Both

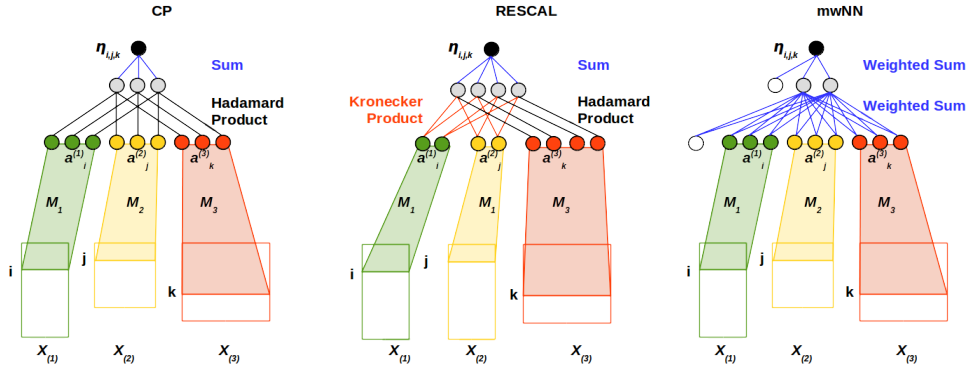

Fig. 2. Illustrating the Compact Model as NNs in 3-D case. Here the rows indexed by i, j and k in the tensor unfoldings $\boldsymbol{X}_{(1)}$, $\boldsymbol{X}_{(2)}$ and $\boldsymbol{X}_{(3)}$ correspond to the three coloured slices in Figure 1, respectively. Note that for RESCAL there are only two mapping matrices instead of three since the entity embeddings are shared between subject and object.

improvements are taken into account in the right-sided option *Stochastic Updating with Late-Starting* of Algorithm 1. In the following we shall refer to this approach as Emma-Hatting.

Embedding Mapping Learning with Back-Propagation: It is easy to see that, in combination with Multi-way Neural Networks, the linear mapping between tensor unfoldings and the latent embeddings could also be considered as one more linear activated layer of the network. To this end the Hatting Algorithm could be replaced with the usual Error Back-Propagation. This aspect also applies to other factorization models as long as such a model can be formulated as an NN. In such cases the latent embeddings become the first hidden layer and the tensor unfoldings become the input layer. For such models the latent embeddings are not explicitly learned but are derived from the product of the input vector and the mapping matrices. This aspect corresponds to the compact model described in Equation (10) and is illustrated in Figure 2. Similar illustrations could also be found in [18] for RESCAL and the mwNN. Note that for MF and RESCAL there are no parameters other than the mapping matrices to be learned; while the mwNN also learns the weights following the embedding vectors. In the following we shall refer to this approach as Emma-BP.

In summary, the Emma-Post approach optimizes the cost functions in Equations (5) and (8) consecutively and separately. Therefore the two error terms of $d_F(\boldsymbol{\mathcal{X}}, \widehat{\boldsymbol{\mathcal{H}}})$ and $d_M(\boldsymbol{A}^{(l)}, \widehat{\boldsymbol{A}}^{(l)})$ are —though minimized to a certain extent— always present. The Emma-Hatting approach also considers these two cost functions. But the LS estimates are calculated more than once and the LS error term —in the long run— is expected to be smaller than that derived from Emma-Post. However, because of this LS update within the factorization algorithm, the

gradient approach may become unpredictable with respect to whether the optima identified with LS correction are better than those without the LS update. The stochastic Hatting Algorithm, from this point of view, could be considered as a compromise between the post mapping and the absolute hatting approach. On the one hand it still regulates the factorization algorithm to satisfy the cost function Equation (8); on the other hand it allows the factorization algorithm to minimize the cost function of Equation (5) continuously as long as the error of Equation (8) is relatively small. In other words, this approach enables better factorization quality by tolerating some acceptable mapping error. By omitting the explicit mapping error, the Emma-BP approach models the factorization and mapping as a whole only in terms of the factorization error of Equation (11).

4 Experiments

In this section we present experiments on three real-world datasets and evaluate the results from two different aspects. First we evaluate our models on a user-item matrix and a KG dataset (both binary) in terms of prediction quality. Then, with another tensor dataset of real values, we focus on the interpretability of the mapped embeddings. The models were implemented in Keras [6] and its backend Theano [1].

4.1 Movielens Data

Data: The Movielens-100K [12] dataset is a user-item matrix containing 100000 ratings from 943 users on 1682 movies. The fact that a rating was performed on an item is encoded as a 1, otherwise we use a 0. Such binary-item matrix could be considered as a special case of a KG with two types of entities and one type of relation.

Task: The major task of a conventional recommender system is to predict the probability of a known user being interested in an known item, as long as event has not been observed in the past. (In terms of a KG this is equivalent to link prediction for one relation between two entities.) In contrast, we intend to predict the probabilities for a *new user* being interested in known items; or vice versa: for a new item to be of interest for known users.

Settings: First, we sample 20% of users to hold out for test and train our models using the remaining 80% with embedding sizes of 20 and 50. In the test phase, we mask a sequence of proportions $[0, 0.1, 0.2, 0.3, 0.4, 0.5]$ of all watched movies of each test user and predict a distribution over all known movies, especially the masked ones. We measure the quality of each prediction in terms of NDCG@k[5]. Further we transpose the user-movie matrix and conduct the same experiment, i.e., we add new movies to the database and try to recommend each movie to the most likely user. Other than [11], we test a logistic MF combined with Emma-Post, Emma-Hatting and Emma-BP. As baseline model we perform a most-popular prediction: we calculate the frequencies of each movie for all users in the training set and interpret these as constant recommendation scores

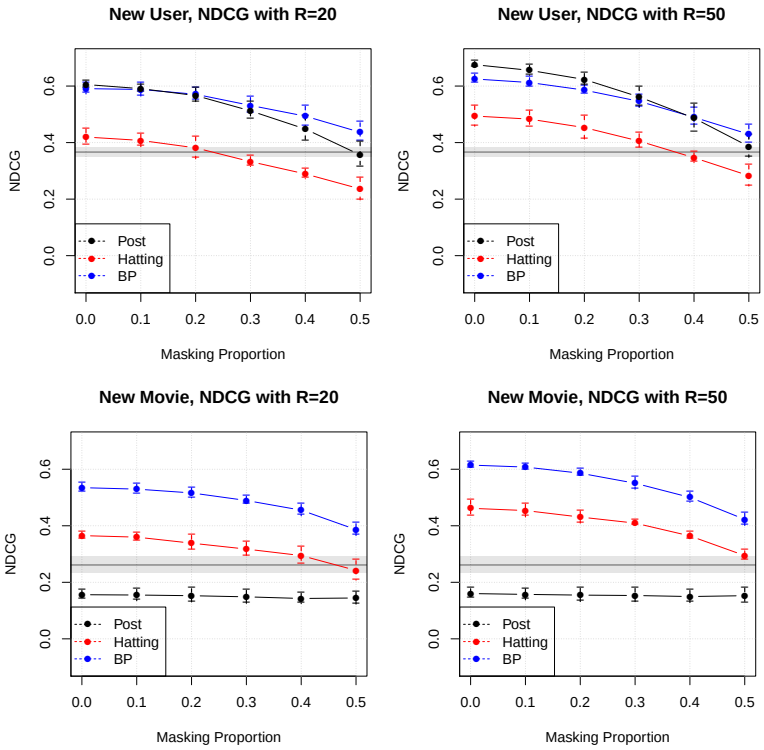

Fig. 3. Evaluation of Prediction for New Users and Movies. The horizontal line and grey bar represent the baseline recommendation and its standard deviation.

for a new user and a new movie, respectively. We repeat this process 5 times with different and mutually exclusive training and test samples in order to derive prediction stability in terms of mean and standard deviations.

Results: The results of the experiments are presented in Figure 4.1. The mean and standard deviation of the NDCG@k scores of the three mapping techniques are visualized in corresponding colors. The horizontal axis represents the proportion of masked entries in the test set; the mean and standard deviation of baseline predictions are demonstrated as the horizontal line and the gray bars.

The performances of the models suggest different trends for new-user and new-movie cases. For new users, the predictions made by logistic MF with Emma-Hatting are suboptimal and may even drop below the baseline for larger masking proportions. Emma-Post and Emma-BP offer better and comparable prediction qualities, though the latter remains advantageous even as the masking proportion becomes quite large. Emma-Post, however, cannot even beat the baseline for new appended movies, while Emma-BP perform noticeably well for all possible masking proportions.

4.2 Knowledge Graph Data

Data: In following experiments we test our models on the Freebase dataset [2] as prepared in [17]. We sample entities that contain at least 500 known relations forming a binary tensor of shape $39 \times 115 \times 115$.

Task: Similar to recommender systems, the conventional link prediction in KGs is performed for each triple of known entities and relations [21]. With our Emma models we intend to predict the existence of links between known and new entities, given some observed but incomplete information about this new entity. More specifically, for an existing KG modelled with a binary tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I \times I \times K}$ we assume a further binary tensor $\boldsymbol{\mathcal{Z}} \in \mathbb{R}^{I' \times I \times K}$ storing a subset of true links between I' new entities and the I known ones. Our task here shall be to predict the unobserved links in $\boldsymbol{\mathcal{Z}}$ based on factorization and mapping models trained only on $\boldsymbol{\mathcal{X}}$.

Settings: In order to also estimate the model's stability we perform a $20\% - 80\%$ Cross-Validation on the data by splitting the entity set into 5 mutually exclusive groups. In each test set we mask and try to recover 20% of known links for each entity with two approaches: 1) We map the entities with masked links into the latent space obtained by the Emma-Post, Emma-Hatting and Emma-BP models that have been trained with the corresponding training set and predict the masked links with the same models; 2) we train a RESCAL and a mwNN model on training *and* test sets, simulating a retraining scenario, and predict the masked links in the conventional way such as in [17]. In both cases we generate negative samples according to the Local-Closed-World-Assumption [17].

Results: In Table 2 we report the prediction quality of AUROC and AUPRC using models of RESCAL and mwNN in combination with all three Emma approaches as well as from retraining. In general, mwNN outperforms RESCAL in terms of larger means and smaller standard deviations in almost all cases, which could also be supported by the results reported in [17]. In predicting for new entities, mwNN combined with Emma-BP yields the best mean values. Especially in terms of AUPRC the advantage could be as large as 33% compared to second best result produced by RESCAL+Emma-Hatting for $R = 10$ and 44% compared with RESCAL+Emma-BP for $R = 30$. The minimal standard deviations are achieved in $5/8$ cases by Emma-Post, though it almost always produces worst mean values in combination with any factorization models. As expected, retraining always offers better predictions than Emma approaches. But do note that a prediction with an Emma model does not cost any run time; while a retraining process for one or multiple entities would demand a comparable amount of time as training an Emma model from scratch. Interpreting the retraining predictions as upper bound, it should also be noted that the combination of mwNN and Emma-BP achieves in most cases results relatively close to those of retraining. We speculate that such canonical model-algorithm combination might enjoy numerical advantage.

Fac.	Mapping	R = 10		R = 30	
		AUROC	AUPRC	AUROC	AUPRC
RESCAL	Retraining	0.901 ± 0.039	0.820 ± 0.059	0.788 ± 0.054	0.616 ± 0.100
	Emma-Post	0.759 ± 0.096	0.600 ± 0.145	$0.693 \pm \mathbf{0.065}$	$0.432 \pm \mathbf{0.106}$
	Emma-Hatting	$\mathbf{0.778} \pm 0.112$	$\mathbf{0.605} \pm 0.152$	0.700 ± 0.091	0.481 ± 0.120
	Emma-BP	$0.700 \pm \mathbf{0.060}$	$0.485 \pm \mathbf{0.092}$	$\mathbf{0.740} \pm 0.090$	$\mathbf{0.509} \pm 0.134$
mwNN	Retraining	0.964 ± 0.008	0.886 ± 0.060	0.970 ± 0.010	0.923 ± 0.017
	Emma-Post	$0.844 \pm \mathbf{0.009}$	0.390 ± 0.101	$0.826 \pm \mathbf{0.035}$	$0.382 \pm \mathbf{0.063}$
	Emma-Hatting	0.847 ± 0.042	0.423 ± 0.118	0.843 ± 0.038	0.394 ± 0.081
	Emma-BP	$\mathbf{0.949} \pm 0.022$	$\mathbf{0.805} \pm \mathbf{0.080}$	$\mathbf{0.931} \pm 0.036$	$\mathbf{0.735} \pm 0.101$

Table 2. Prediction Qualities of RESCAL and mwNN in combination with all three mapping approaches on a FreeBase dataset.

4.3 Amino Acid Data

With previous experiments we have shown that Emma models are able to predict links between every known entity and a newly appended entity based on incomplete information. With the following experiment we also show that Emma models can map a new entity into the latent space with high interpretability — here in terms of correlation coefficients.

Data: The Amino Acid Dataset [4] contains a three-way tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{5\times51\times201}$ and a matrix $\boldsymbol{Y} \in \mathbb{R}^{5\times3}$. The latter one describes the proportion of 3 types of amino acid mixed according to 5 different recipes. The corresponding 5 samples are then measured by fluorescence with excitation 250-300 nm, emission 250-450 nm on a spectrofluorometer and the measurements are recorded in the tensor $\boldsymbol{\mathcal{X}}$. With a CP factorization producing matrices of dimensions $\boldsymbol{A}^{(1)} \in \mathbb{R}^{5\times3}$, $\boldsymbol{A}^{(2)} \in \mathbb{R}^{201\times3}$ and $\boldsymbol{A}^{(3)} \in \mathbb{R}^{61\times3}$ it is expected that each column in $\boldsymbol{A}^{(1)}$ would strongly correlate with one column in the recipe matrix $\boldsymbol{Y}$. Note that the order of the columns in $\boldsymbol{A}^{(1)}$ is arbitrary and may not correspond to the column in $\boldsymbol{Y}$ at the same position. For more details please refer to [4].

Task: The latent embeddings learned from this data set are expected to be interpretable in terms of correlations with known recipes. If a new entity is correctly mapped into the latent space, its *mapped* embedding(s) along with other *learned* embeddings would also correlate with the corresponding column in the recipe matrix. Here we assume the information observed on the new entity to be complete and do not perform any masking.

Settings: We remove each slice $\boldsymbol{X}_{i,\bullet,\bullet}, i \in [1,5]$ (corresponding to a certain recipe) from the tensor and calculate CP factorizations with Emma-Post and Emma-Hatting based on the rest of the data. The slice held out is then mapped to the 3-D latent space with mapping matrix and appended to the learned embeddings of the other slices. We then calculate the mean of its column-wise Pearson correlation coefficients with $\boldsymbol{Y}$. Further we conducted the same experiment with two slices removed at a time.

Results: With the first leaving-one-out experiment setting we derive accordingly 5 averaged correlations and for the second leaving-two-out setting we have $\frac{5!}{3!\cdot 2!} = 10$ values. We report that the means and standard deviations of the correlation coefficients derived from Emma-Post and Emma-Hatting to be both 0.999 ± 0.001. As for leaving-two-out experiments, Emma-Post achieves 0.991 ± 0.007 and Emma-Hatting yields the same mean value but a larger standard deviation of 0.015. To this end we conclude that both mapping approaches can map one or two new slices into the latent space in a way that its embedding(s) —along with other embeddings learned from factorization— correlates column-wise with the recipe matrix $\boldsymbol{Y}$ with a high correlation score. As expected, in leaving-two-out experiments the correlation coefficients decrease slightly due to the smaller training set. Note that once again Emma-Post seems to produce smaller standard deviations just as the KG experiments.

5 Related Works

[10] introduced a method to map user attributes (referred to as 'content information' in the context of content filtering) into the latent embedding space to solve cold-start problem for new entities. Despite the fact that we are not considering the cold-start problem and do not require content information, this model still shares a few aspects with ours: 1) In both approaches, the codomain of the mapping function is the latent space learned by factorization models with the purpose of finding latent embeddings for new entities; 2) both approaches use modular learning that can be combined with a variety of existing factorization models. An important difference is the domain of the mapping function. In our case it is the observable feature space of an entity, while in [10], it is the user attribute space. It would be interesting, though, to combine our models with such content information: In the first step, one could perform content filtering to produce some first recommendations. Secondly, one could interpret these recommendations as incomplete and enrich them using Emma models by mapping them into the latent space and perform link predictions.

Our proposed Emma-BP model shares the idea of learning the factorization jointly with an implicit latent embeddings with the Temporal Latent Embedding (TLEM) Model [9], where a concatenated NN is trained with observable features as inputs, which are mapped to some implicit latent features. In TLEM one intends to model the *consecutive* effect of a sequence of events on the next one; while we are interested in the *collaborative* effect among entities.

6 Conclusions

The major contribution of this paper is to propose three approaches for mapping new entities into the latent embedding space learned by a wide variety of factorization models.

Our approaches are not based on retraining while obtaining comparable quality to model retraining. Our framework describes factorization and mapping

problems on an abstract level, which could inspire the development of further mapping approaches. During our experiments we also realized the model's restrictions in practice: Due to the unfolding operations in the mapping model, the dimensions of the model inputs increase quadratically with the dimensions of the tensor. For instance, in our KG experiment in Section 4 with a tensor of shape $39 \times 115 \times 115$, the mapping model requires inputs of dimensions 4485, 4485 and 13225, respectively. As part of future work, we will explore different approaches for dimensionality reduction. In addition, we plan to study non-linear extensions to Emma as well as the application to recurrent NNs.

References

[1] Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., Bengio, Y.: Theano: a CPU and GPU math expression compiler. In: Proceedings of SciPy (2010)

[2] Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database. In: ACM SIGMOD (2008)

[3] Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: NIPS (2013)

[4] Bro, R.: Multi-way analysis in the food industry: models, algorithms, and applications. Ph.D. thesis, Københavns Universitet (1998)

[5] Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., Hullender, G.: Learning to rank using gradient descent. In: Proceedings of the 22nd international conference on Machine learning. pp. 89–96. ACM (2005)

[6] Chollet, F.: Keras: Deep learning library for theano and tensorflow. https://github.com/fchollet/keras (2015)

[7] Culotta, A., McCallum, A.: Joint deduplication of multiple record types in relational data. In: ACM information and knowledge management (2005)

[8] Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmann, T., Sun, S., Zhang, W.: Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In: ACM SIGKDD (2014)

[9] Esteban, C., Schmidt, D., Krompaß, D., Tresp, V.: Predicting sequences of clinical events by using a personalized temporal latent embedding model. In: ICHI (2015)

[10] Gantner, Z., Drumond, L., Freudenthaler, C., Rendle, S., Schmidt-Thieme, L.: Learning attribute-to-feature mappings for cold-start recommendations. In: ICDM (2010)

[11] Gantner, Z., Rendle, S., Freudenthaler, C., Schmidt-Thieme, L.: Mymedialite: A free recommender system library. In: ACM conf. on Recommender systems (2011)

[12] Herlocker, J.L., Konstan, J.A., Borchers, A., Riedl, J.: An algorithmic framework for performing collaborative filtering. In: ACM SIGIR (1999)

[13] Hoaglin, D.C., Welsch, R.E.: The hat matrix in regression and anova. The American Statistician 32(1), 17–22 (1978)

[14] Kiers, H.A.: Towards a standardized notation and terminology in multiway analysis. Journal of chemometrics 14(3) (2000)

[15] Kolda, T.G., Bader, B.W.: Tensor Decompositions and Applications. SIAM Review (2009)

[16] Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer 42(8), 30–37 (2009)

[17] Krompaß, D., Baier, S., Tresp, V.: Type-constrained representation learning in knowledge graphs. In: ISWC (2015)

[18] Nickel, M., Murphy, K., Tresp, V., Gabrilovich, E.: A review of relational machine learning for knowledge graphs. Proceedings of the IEEE 104(1), 11–33 (2016)
[19] Nickel, M., Tresp, V., Kriegel, H.P.: A three-way model for collective learning on multi-relational data. In: ICML (2011)
[20] Socher, R., Chen, D., Manning, C.D., Ng, A.: Reasoning with neural tensor networks for knowledge base completion. In: NIPS (2013)
[21] Taskar, B., Wong, M.F., Abbeel, P., Koller, D.: Link prediction in relational data. In: NIPS (2003)
[22] Tucker, L.R.: Some mathematical notes on three-mode factor analysis. Psychometrika 31(3) (1966)