Large-Scale Factorization of Type-Constrained Multi-Relational Data

Conference Paper · October 2014

DOI: 10.13140/2.1.4511.7441

3 authors, including:

Denis Krompass
Siemens
21 PUBLICATIONS **1,084** CITATIONS
SEE PROFILE

Volker Tresp
Siemens
290 PUBLICATIONS **7,632** CITATIONS
SEE PROFILE

Large-Scale Factorization of Type-Constrained Multi-Relational Data

Denis Krompaß
Ludwig Maximilian University
Munich, Germany
Denis.Krompass@campus.lmu.de

Maximilian Nickel
Massachusetts Institute of Technology
Cambridge, MA, USA
mnick@mit.edu

Volker Tresp
Siemens AG, Corporate Technology
Munich, Germany
Volker.Tresp@siemens.com

***Abstract*—The statistical modeling of large multi-relational datasets has increasingly gained attention in recent years. Typical applications involve large knowledge bases like DBpedia, Freebase, YAGO and the recently introduced Google Knowledge Graph that contain millions of entities, hundreds and thousands of relations, and billions of relational tuples. Collective factorization methods have been shown to scale up to these large multi-relational datasets, in particular in form of tensor approaches that can exploit the highly scalable alternating least squares (ALS) algorithms for calculating the factors. In this paper we extend the recently proposed state-of-the-art RESCAL tensor factorization to consider relational type-constraints. Relational type-constraints explicitly define the logic of relations by excluding entities from the subject or object role. In addition we will show that in absence of prior knowledge about type-constraints, local closed-world assumptions can be approximated for each relation by ignoring unobserved subject or object entities in a relation. In our experiments on representative large datasets (Cora, DBpedia), that contain up to millions of entities and hundreds of type-constrained relations, we show that the proposed approach is scalable. It further significantly outperforms RESCAL without type-constraints in both, runtime and prediction quality.**

I. Introduction

A number of recently developed knowledge bases (KB) like DBpedia, Freebase or the YAGO ontology all contain millions of different entities and hundreds to thousands of relations. Facts in these knowledge bases are usually represented as binary relations between entities in form of <subject, predicate, object> triples. The set of all triples defines a large multi-graph where entities correspond to nodes and relational tuples correspond to directed links between nodes. For large multi-relational datasets with a multitude of relations, tensor-based methods have recently been proposed where binary relations are represented as square adjacency matrices that are stacked as frontal slices in a three-way adjacency tensor. Learning is then based on the factorization of this adjacency tensor where highly efficient alternating least squares (ALS) algorithms can be exploited, which partially utilize closed-form solutions. In most relational learning settings a closed-world assumption is appropriate and missing triples are treated as negative evidence, and here ALS, which is applicable with complete data, is highly efficient. In contrast, when a closed-world assumption is not appropriate, for example when the goal is to predict movie ratings, other approaches such as stochastic gradient descent can be more efficient, where all unobserved ratings are treated as missing. We argue that by exploiting type-constraints, the amount of "trivial" negative evidence (under a closed-world assumption) can be significantly reduced, what makes ALS-based methods very attractive again for large type-constrained data. To the best of our knowledge, there is no ALS algorithm for factorization that exploits closed-form solutions and considers type-constraints during factorization. In addition, in contrast to factorizations that consider every unobserved triple as missing, type-constraints add prior knowledge about relations to the factorization, such that triples that disagree with the type constraints are excluded during factorization. Type-constraints are present in most datasets of interest and imply that for a given relation, only subject entities that belong to a certain *domain* class and object entities that belong to a certain *range* class can be related. For instance, the relation *marriedTo* is restricted to human beings. In some domains, type-constraints often define complex overlapping classes since any entity can occur as subject or/and object (the domain and range class of a relation can overlap) in multiple relations (domain and range classes of different relations can overlap).

By neglecting type-constraints when using ALS under closed-world assumptions, it is very likely that the factorization will need more degrees of freedom (controlled by the rank of the factorization), since the latent vector representations for the entities and their interactions have to account for a huge amount of meaningless unobserved relations between entities. We argue that, especially for larger datasets, the required increase of model complexity will lead to an avoidable high runtime and memory consumption. By focusing only on relevant tuples, high quality latent vector representations can be constructed with a significant reduction in the required degrees of freedom.

In this paper we propose a general collective factorization approach for binary relations that exploits relational type-constraints in large multi-relational data by using ALS (similar to [1]). In difference to other factorization methods that have also been applied to large scale datasets [2], it also utilizes closed-form solutions during optimization. The proposed method is capable of factorizing hundreds of adjacency matrices of arbitrary shapes, that represent local closed-world assumptions, into both a shared low-rank latent representation for entities and a representation for the relation-specific interactions of those latent entities (Figure 2.b). Additionally we show that in the absence of a-priori-defined type-constraints, useful type-constraints can be approximated based on observed subject or object entities (and potential relations to them). During factorization, this method assures that relations only contribute to the latent representations of entities that agree with their type-constraints.

We will demonstrate the benefits of our approach in the context of typed-constrained multi-relational datasets on experiments with a small (Cora), a medium (DBpedia-Music) and a very large (DBpedia) dataset. We further show that the explicit consideration of relational type-constraints results in an increase of prediction quality at reduced model complexity (rank). As a result of the decreased model complexity, significant improvements in both runtime and memory requirements could be achieved.

This paper is structured as follows. The next section contains related work. In Section III we explain our notation and review the RESCAL tensor factorization model. In Section IV we introduce our approach. In Section V we describe our experiments. We conclude in Section VI.

II. Related Work

[3, 4] provided a general framework for modeling multi-relational data with collective matrix factorization (CMF) and showed that a parallel factorization of auxiliary matrices, that contain additional information about entities (e.g. items or users), have a benefit on the prediction quality. Unfortunately, these extensions are limited to a low number of relations to take contextual information into account in a recommender system [5, 6, 7, 8, 4]. [9] addressed the sparsity issue with non-negative constraints. [10] proposes a completely decoupled CMF approach for simultaneously solving multiple rating problems and [11] links local factors through a globally shared multivariate regression model. In [12] two entity types are not allowed to share more than one relation.

Recently, higher-order tensors factorizations were proposed for multi-relational learning [1] since they offer a more flexible and natural representation when dealing with a multitude of relations. The work of [9] has been extended recently to tensors [13] as well. Tensors have been applied to Web analysis in [14] and to ranking predictions in the Semantic Web in [15]. [16] applied tensor models to rating predictions. The use of factorization approaches to predict ground atoms was pioneered in [17]; [1], [18], [19], and [20] applied tensor models for this task, where [1] introduced the RESCAL model. This model was the basis of many recent published works: [21] introduced non-negative constraints, [22] presented a logistic factorization and [20] explicitly models the 2nd and 3rd order interactions. [23] introduced neural tensor networks that includes a RESCAL factorization and [2] used a neural tensor decomposition for defining graph-based priors during construction of the Google Knowledge Vault. They also exploited local closed-world assumptions (LCWA), a known concept in the semantic web domain [24].

[25] introduced weighted loss functions for RESCAL. Via such loss functions it is possible to introduce relational type-constraints for RESCAL by assigning zero weights to triples that are excluded through type-constraints. Unfortunately, the scalability of RESCAL gets lost when using this type of weighted loss function. The Hadamard (element wise) matrix product with the weighting tensor leads to expensive matrix operations in the gradients, since a materialization of a complete frontal slice reconstructed from the learned latent representation of the data is required (the product AR_kA^T), which is dense. Besides the high computational costs that arise during this materialization in the gradient, the memory requirements for this materialization itself will be extensive which makes this approach impractical. For example, consider the music domain of DBPedia (311,474 entities). The materialization of a frontal slice (AR_kA^T) will need approximately 776 GB of memory. In this work we will demonstrate how to consider relational type-constraints with no loss in scalability. We propose a method that is capable of factorizing type-constrained multi-relational datasets with millions of entities and hundreds of relations into a meaningful low-rank factorization on a commodity hardware.

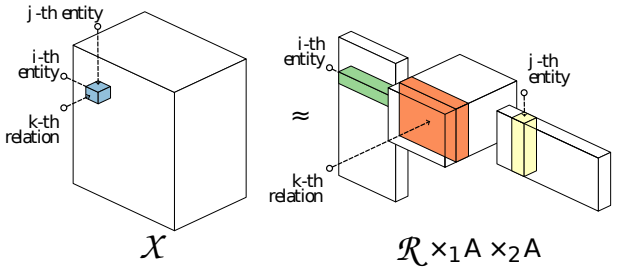


Fig. 1. RESCAL model for binary relations

III. Preliminaries

A. Notation

In the following, calligraphic letters like $\mathcal{X}$ or $\mathcal{R}$ represent tensors. $\hat{\mathcal{X}}$ is our notation for a multi-relational list of matrices with arbitrary shapes. X and R will represent single matrices. X_k and $\hat{X}_k$ represent frontal slices of the tensors or lists, respectively. $X_{i:}$ is the i*th* row in the matrix X and x_{ijk} a single entry in the tensor $\mathcal{X}$. $\circledast$ represent the Hadamard (element-wise) matrix product, where $\otimes$ is the Kronecker product.

The set of subject entities that potentially participates in a binary relation is defined by the *domain* of that relation. The set of object entities is defined by its *range*. We further exploit the concept of *local closed-world semantics*, which assumes for each relation that the data is complete regarding only the observed entities in that relation (links to and from unobserved entities are ignored). We refer to [2] for a comprehensive detailed definition.

B. RESCAL

RESCAL [1] is a three-way-tensor factorization model that has been shown to lead to very good results in various canonical relational learning tasks like link prediction, entity resolution and collective classification [26]. One main feature of RESCAL is that it can exploit a collective learning effect when used on relational data, since an entity has a unique representation over occurrences as a subject or as an object in a relationship and also over all relations in the data. When dealing with semantic web data, multi-relational data is represented as triples that are stored in an adjacency tensor $\mathcal{X}$ of shape $n \times n \times m$, where n is the number of all entities in the data and m is the number of relation types. Each of the m frontal slices X_k of $\mathcal{X}$ represents an adjacency matrix for all entities in the dataset with respect to the k-th relation.

Given an adjacency tensor $\mathcal{X}$, RESCAL computes a factorization of $\mathcal{X}$ where each entity is represented via a *unique* row in the r-dimensional latent factor matrix $A \in \mathbb{R}^{n \times r}$ and each relation is represented via a matrix $R_k \in \mathbb{R}^{r \times r}$ (see Figure 1). For optimization a regularized least squares loss function is minimized.

$$\begin{aligned} \text{loss}(\mathcal{X}, A, \mathcal{R}) &= \sum_k \|X_k - AR_kA^T\|_F^2 \\ &\quad +\lambda_A\|A\|_F^2 + \lambda_R \sum_k \|R_k\|_F^2 \end{aligned} \quad (1)$$

where $\lambda_A \geq 0$ and $\lambda_R \geq 0$ are hyperparameters and $\|\cdot\|_F$ is the Frobenius norm. The cost function can be optimized via very efficient alternating least squares (ALS) that effectively exploite data sparsity [1]. During factorization, RESCAL finds a unique latent representation for each entity that is shared between all relations in the dataset. The relation-specific, often asymmetric, interaction between subject and object entities are captured by the frontal slices of the core tensor $\mathcal{R}$.

IV. Collective Factorization of Type-Constrained Relations

We now describe the novel collective factorization method for type-constrained multi-relational datasets. As already discussed in the introduction, neglecting local closed-world assumptions defined through type-constraints in large type-constrained multi-relational data can lead to higher requirements in degrees of freedom of the factorization model. This is mostly due to the fact that these factorization approaches ignore that most entities only play a role in some relations but are naturally excluded in others. As a consequence, these models have to account for a huge amount of meaningless relationships between entities.

In case of the highly scalable RESCAL tensor factorization [1], where the multi-relational data is organized in a large (but sparse) adjacency tensor (Figure 2.a), every unobserved relation between entities is treated as negative evidence because it ignores type-constraints. Considering given type-constraints for each relation (colored blocks in the tensor in Figure 2.a), what we actually want is to achieve a factorization that ignores all relations between entities that disagree with the type-constraints. Technically speaking, a collective factorization of the local sub-matrices (which can have arbitrary shapes) defined within the adjacency tensor by the type-constraints is desired (Figure 2.b). The joined factorization of these sub-matrices guarantees that the latent representation computed by the factorization will be unaffected by meaningless relations between entities and reduces the required model complexity (controlled by the rank). To achieve this, we directly introduce the type-constraints into the RESCAL least-squares cost function (Equation 1):

$$\begin{aligned} \text{loss}(\hat{\mathcal{X}}, A, \mathcal{R}, \hat{\mathcal{S}}, \hat{\mathcal{O}}) &= \sum_k \|\hat{X}_k - \hat{S}_kAR_kA^T\hat{O}_k^T\|_F^2 \\ &\quad +\lambda_A\|A\|_F^2 + \lambda_R \sum_k \|R_k\|_F^2 \end{aligned} \quad (2)$$

where $\hat{\mathcal{S}}$ and $\hat{\mathcal{O}}$ are lists of binary matrices that represent the type-constraints for each relation. More precise, they select latent-vector representations of the entities that agree with the domain ($\hat{\mathcal{S}}$) and range ($\hat{\mathcal{O}}$) constraints of any particular relation. $\hat{X}_k$ represents the data under local closed-world assumption for relation k that could have been (but not necessarily) extracted from the frontal slice X_k of the large adjacency tensor $\mathcal{X}$ ($\hat{X}_k = \hat{S}_kX_k\hat{O}_k$). Note that if relation $\hat{X}_k$ has the shape $n_k \times m_k$, then $\hat{S}_k$ is of shape $n_k \times z$ and has exactly n_k entries (the remaining ones are zero) that selects rows from the factor matrix A and stacks them in a new type-constrained factor matrix $A_{\hat{S}_k}$, where z is the total number of entities in the dataset and n_k is the amount of entities that belong to the domain of relation k. ($\hat{O}_k$ performs the same function for the range of the relation k and stacks them into $A_{\hat{O}_k}$ of shape $m_k \times z$, respectively).

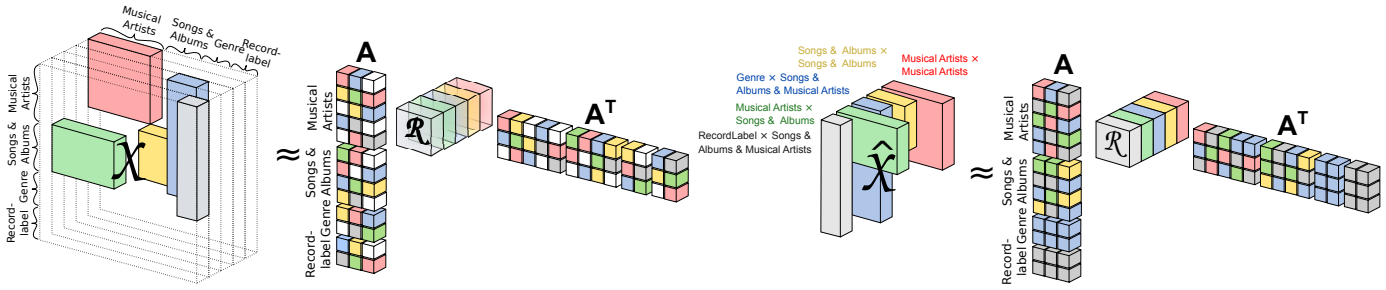
Even though the cost function defined in Equation 2 looks very similar to the cost function of RESCAL (Equation 1) it contains important differences, since we are actually factorizing a list of matrices of arbitrary shapes (indicated by the hat symbol) instead of uniformly shaped frontal tensor slices.

However, through the similar formulation of the problem we are able to exploit the nice scalability properties of RESCAL in our model. For that we adapted the RESCAL update functions such that they consider type-constraints and allow a factorization a list of local closed-world adjacency matrices. Fortunately we are still able to utilize an efficient closed-form solution for the update of the latent interactions matrices R_k.

$$\begin{aligned} A_{i:} &= \left[\sum_k \hat{S}_k^T\hat{X}_kA_{\hat{O}_k}R_k^T + \hat{O}_k^T\hat{X}_k^TA_{\hat{S}_k}R_k\right]_{i:} \\ &\quad \left[\sum_k \hat{s}_{iik}E_k + \hat{o}_{iik}F_k\right]^{-1} \\ \textit{with } E_k &= R_kA_{\hat{O}_k}^TA_{\hat{O}_k}R_k^T \\ \textit{and } F_k &= R_k^TA_{\hat{S}_k}^TA_{\hat{S}_k}R_k \end{aligned} \quad (3)$$

Note that the second part of the equation (the inverse) is dependent on the domain and range constraints of each relation, and therefore the sum over the relations in the inverse is dependent on the regarded entity. The term $\hat{s}_{iik}$ represents the i-*th* diagonal entry of the diagonal matrix $\hat{S}_k^T\hat{S}_k$ and is one, if the entity i is part of the domain of relation k and zero otherwise. The term $\hat{o}_{iik}$ represents the i-*th* diagonal entry of the diagonal matrix $\hat{O}_k^T\hat{O}_k$ and is one if the entity i is part of the range of relation k, zero otherwise. Note that whereas RESCAL only requires the calculation of one inverse here we need to calculate z inverses. This can be very expensive, especially when a high rank is used (the inverse of a $r \times r$ has to be computed). We will discuss in Section IV-A how the entities can be grouped to reduce the amount of computed inverses. On the other hand, the first part of Equation 3 can be calculated very efficiently, especially when dealing with large type-constrained datasets, since the much smaller matrices $\hat{X}_k^T$, $A_{\hat{S}_k}$ and $A_{\hat{O}_k}$ are involved. The similar RESCAL update would have to use the complete frontal slices X_k of the partially observed tensor $\mathcal{X}$ and the whole factor matrix A in this case (which can be multitudes larger than $A_{\hat{S}_k}$ and $A_{\hat{O}_k}$).

For the latent interaction tensor $\mathcal{R}$ we have to consider, that each frontal slice is updated with respect to two different



(a) A partially observed tensor that is factorized by RESCAL

(b) Factorizing with respect to domain and range constraints of the relations

Fig. 2. Schematic factorization of type-constrained multi-relational data using RESCAL (a)[1] (ignores type-constraints), and TypeConCF (b), the proposed model that considers type-constraints during factorization. For better understanding, we used example data from DBpedia (music domain) in the figures. The colored boxes on the left side of the equations represent the regions defined by relational type-constraints. The white (or transparent) regions correspond to data that disagree with the type-constraints. On the right sites of the approximation signs, the coloring illustrates which parts of the data (left side) influence the latent factors during factorization. In both approaches (a,b), the data is factorized into the same latent factor structure, a shared latent representation for the entities (factor matrix A) and an interaction core tensor $\mathcal{R}$ (where each frontal slice is specific for each relation). When using RESCAL (a) which imposes a closed-world assumption on the whole tensor, then the latent representations of the entities (A) and their interactions ($\mathcal{R}$) are influenced by data that disagrees with the type-constraints (white boxes). We propose a collective factorization method (TypeConCF) that factorizes the same data, but under local closed-world assumptions. Thereby a list of local sub-matrices of arbitrary shapes extracted from the partially observed tensor through type-constraints is directly factorized leading to clean (only colored boxes) latent representations.
For example, the second relation (yellow) is constrained to relate only albums and songs, therefore this relation only influences the latent factors that represent the song and album entities. The latent representation of genre entities is only determined by the third relation (blue). The fourth relation on the other hand (green) provides information on the latent representation of musical artists, songs and albums.

sets of latent factors contained in A (namely $A_{\hat{S}_k}$ and $A_{\hat{O}_k}$). For each frontal slice of $\mathcal{R}$ we get the following closed-form solution for the update:

$$\begin{aligned} R_k &= ((C \otimes B)^T (C \otimes B) + \lambda \mathbf{I})^{-1} (C \otimes B)^T vec(\hat{X}_k) \\ &with \quad B = A_{\hat{S}_k} \; and \; C = A_{\hat{O}_k}. \end{aligned}$$

As in [27] we can exploit the following property of the singular value decomposition (*SVD*) regarding the Kronecker product of two matrices [28]

$$C \otimes B = (U_C \otimes U_B)(\Sigma_C \otimes \Sigma_B)(V_C^T \otimes V_B^T)$$

leading to the much more efficient update for R_k

$$R_k = V_B (P \circledast U_B^T \hat{X}_k U_C) V_C^T \tag{4}$$

where P is defined as follows: Let Q be defined as a diagonal matrix and where

$$\begin{aligned} Q_{ii} &= \frac{D_{B_{ii}} D_{C_{ii}}}{D_{B_{ii}}^2 D_{C_{ii}}^2 + \lambda} \\ \text{with } D_B &= \Sigma_B \otimes \Sigma_B \text{ and } D_C = \Sigma_C \otimes \Sigma_C. \end{aligned}$$

Then P can be constructed by a columnwise reshape of the diagonal in Q.

Note, that we are generally dealing with different domain and range constraints for each relation, and $A_{\hat{S}_k}$ and $A_{\hat{O}_k}$ will differ for each update R_k. In contrast to the updates in RESCAL we cannot use the result of one *SVD* on the factor matrix A for all frontal slice updates for $\mathcal{R}$. In our case we have to perform two *SVD*s for each update R_k. This might seem inefficient at first moment, but the *SVD*s have to be only calculated for the smaller factor matrices $A_{\hat{S}_k}$ and $A_{\hat{O}_k}$. Additionally, the matrix products $U_B^T \hat{X}_k U_C$ in Equation 4 become much more efficient, since they involve smaller matrices, when compared with the RESCAL updates, where U would be as large as A (which has $|z|$ rows).

A. Entity Grouping

The updates for the latent representations of the entities (factor matrix A) can be further optimized. We can accelerate these updates by grouping entities that are occurring or are absent in the same relations. For entities of the same group, the second part of Equation 3 (the inverse) is the same and needs only to be calculated once. Consider the following example: If we look at the example dataset from Figure 2 (DBpedia-Music) which contains at the time of this writing 311,474 different entities. If we could assume that all musical artists always occur together in each relation (as subjects and as objects), then we could put them into one group. The same could be applied to songs, genres, record labels and albums. Instead of calculating 311,474 different inverses we only calculate one for each group (only five in this example) and decrease the computation time for the inverses in the A-update dramatically. To include this kind of domain knowledge, we would only have to provide the type-constraints in form of the factor selection matrices $\hat{\mathcal{S}}$ and $\hat{\mathcal{O}}$.

However, in absence of prior knowledge about the real type-constraints, the type-constraints (and the corresponding closed-world assumptions) can be approximated by excluding every unobserved subject or object in a relation.[1] The derived lists of type-constraint matrices $\hat{\mathcal{S}}$ and $\hat{\mathcal{O}}$ can then be used for grouping. In Algorithm 1 our complete approach in absence of prior knowledge on type-constraints is shown. Please note, that we also provide a partially observed adjacency tensor as input to the algorithm in order to exploit a non-random initialization of the parameters as proposed by [29].

B. Relation to Other Factorizations

The proposed collective factorization will transform in other well known third-order tensor factorization methods, if

[1]During factorization the (relation specific) excluded subject and object entities are then ignored

Algorithm 1 Type-Constrained Collective Factorization

Require: $\mathcal{X}$ (A partially observed adjacency tensor (Figure 2.a))

function TYPECONCF($\mathcal{X}, r, \lambda_A, \lambda_R$) ▷ r: rank of factorization, λ_A, λ_R: regularization parameters
  $\hat{\mathcal{S}}$, $\hat{\mathcal{O}}$=*approximateTypeConstraints*($\mathcal{X}$) ▷ Section IV-A
  A,$\mathcal{R} = initialize(\mathcal{X},r)$ ▷ based on eigendecomposition of $\sum_k (X_k + X_k^T)$ [29]
  $\hat{\mathcal{X}} = shrink(\mathcal{X},\hat{\mathcal{S}},\hat{\mathcal{O}})$ ▷ apply closed-world assumptions to the data: $\hat{X}_k = \hat{S}_k X_k \hat{O}_k$ (Figure 2.b)
  repeat
    $A \leftarrow updateA(\hat{\mathcal{X}}, A, \mathcal{R}, \hat{\mathcal{S}}, \hat{\mathcal{O}}, \lambda_A)$ ▷ Equation 3
    $\mathcal{R} \leftarrow updateR(\hat{\mathcal{X}}, A, \hat{\mathcal{S}}, \hat{\mathcal{O}}, \lambda_R)$ ▷ Equation 4
  until convergence or max iteration reached
  return A,$\mathcal{R}$
end function

TABLE I. STATISTICS OF THE DATASETS USED IN THE EXPERIMENTS.

Dataset	Entities	Relations	Facts
Cora	2 497	10	47 547
DBpedia-Music	311 474	7	1 006 283
DBpedia[2]	2 255 018	511	16 945 046

the relational type-constraints have a certain structure. In the absence of type-constraints, $\hat{\mathcal{S}}$ and $\hat{\mathcal{O}}$ become a collection of square identity matrices and drop out of the equation. The result is the RESCAL tensor factorization [1]. If for each relation the domain and range classes are disjunct, we get a Tucker2 [30] like decomposition. If we have equal range constraints for all relations that are disjunct from the various domain constraints of all relations (which are also disjunct from each other), we get a similar decomposition as PARAFAC2 [31] (but R_k is not diagonal).

V. EXPERIMENTS

In the following we report experimental results on three different datasets, the Cora[3] dataset and two datasets extracted from DBpedia[4] [32]. Details of these datasets can be inferred from Table I. We evaluated our approach using simple link prediction tasks. For all datasets we used the same evaluation routine as follows: First, we constructed a partially observed tensor as shown in Figure 2.a (left) from the data, where the frontal slices correspond to the entity adjacency matrices for each relation. From the complete tensor, we defined type-constraints either based on prior knowledge (Cora, DBpedia-Music) or we approximated them (Section IV-A). For steps during evaluation we always sampled entries (ones and zeros) that obey the domain and range constraints of each relation. We conducted 10-fold cross-validation, where we sampled ten disjunct sets of entries without replacement from the tensor. For the Cora dataset, we used the complete data (under local closed-world assumption) for that, in case of the DBpedia datasets this was not feasible due to the high amount of possible entries. In that cases, the sets contained one tenth of the observed relations between entities (ones in the adjacency tensor) and ten times as many unobserved relations (zeros). In each iteration we removed all links present in one set (test set) and factorized the data based on the remaining data (training set). The quality of the prediction (ranking) is measured in Area Under The Precision Recall Curve (AUPRC) using the test set. Reported are always the mean AUPRC scores with corresponding error bars (they are quite small). The proposed method (TypeConCF) always used approximated type-constraints from the training set. Additionally we measure and compare the runtime in seconds. In case of proposed method, the grouping of entities is included in those measurements. All experiments are conducted with a Intel(R) Xeon(R) CPU W3520 @2.67GHz and the models were exclusively implemented in Python using the numpy/scipy library for scientific computing (OpenBLAS, 4 Threads). For all experiments and models, 20 iterations of ALS were performed. For the implementation of RESCAL, we used the python code provided by the author[5].

A. Cora

The dataset links authors, titles and venues through citation records. In addition, the dataset contains relations that link different spellings of entities. The results on the Cora dataset is shown in Figure 3.a and d.

Both models, RESCAL and TypeConCF, perform comparable well on this dataset (Figure 3.a). Both achieve a maximum score of approx. 0.96 in AUPRC (Random: 0.013). It can be observed that in contrast to RESCAL, the prediction quality of TypeConCF increases much faster with the rank of the factorization. At a rank of one, both methods have a similar low score (0.11). For a rank of 25, TypeConCF reaches already 0.8 where RESCAL achieves only 0.66 and at rank 100, TypeConCF reaches almost its maximum score, where RESCAL achieves a significantly lower score of 0.89. With higher ranks, RESCAL starts to catch up and achieves a comparable performance at a rank of approximately 400.

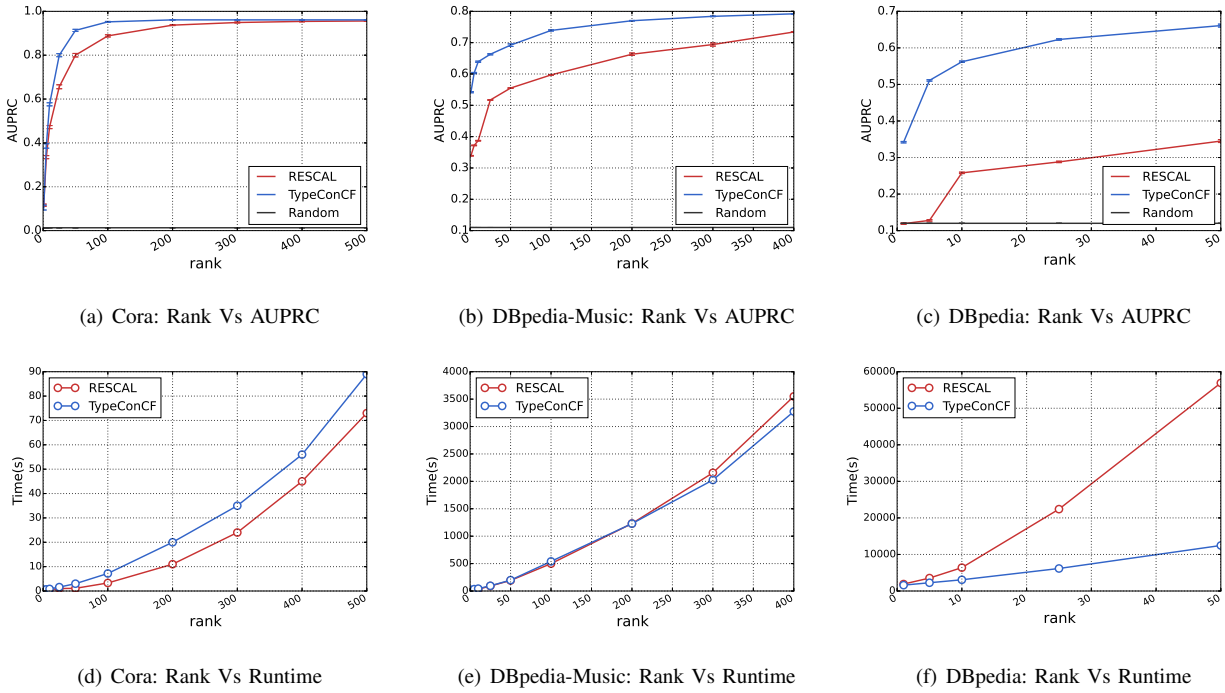
Regarding the runtime of both methods (Figure 3.d), the RESCAL factorization is clearly faster than TypeConCF when using the same rank for the factorization. However, if the prediction quality is taken into account, TypeConCF is achieves a good factorization significantly sooner than RESCAL. For the maximum prediction quality, RESCAL needs 73 seconds (since it needs a rank of 500), TypeConCF on the other hand needs only 20 seconds (rank 200).

[2]We extracted relations that relate at least 1000 entity pairs and excluded all entities that are related less than 5 times throughout all relations

[3]http://alchemy.cs.washington.edu/data/cora/

[4]http://wiki.dbpedia.org/Downloads39?v=pb8; Dataset: Mapping-based Properties (Cleaned)

[5]https://github.com/mnick/scikit-tensor



(a) Cora: Rank Vs AUPRC (b) DBpedia-Music: Rank Vs AUPRC (c) DBpedia: Rank Vs AUPRC

(d) Cora: Rank Vs Runtime (e) DBpedia-Music: Rank Vs Runtime (f) DBpedia: Rank Vs Runtime

Fig. 3. Results on the Cora, DBpedia-Music and DBpedia datasets. (a,b,c) Plotted are the rank used for the factorization against the performance in Area Under Precision Recall Curve (AUPRC). Red: The performance of the RESCAL tensor factorization on the partial observed tensor constructed from the various datasets as illustrated in Figure 2.a. Blue: The performance of the proposed collective factorization for type-constrained multi-relational data (TypeConCF) using the automatic grouping of entities with respect to active entities in each relation (Section IV-A). Grey: Random ranking. (d,e,f) Shown are the same two methods, but instead of AUPRC the runtime in seconds (including time for training and prediction) is measured. In case of TypeConCF, the time for the entity grouping is also included in the measurement.

B. DBPedia-Music-Domain

For the DBpedia-Music dataset, the partially observed tensor is of shape 311,474 $\times$ 311,474 $\times$ 7. The relations are $\{$*Genre, RecordLabel, associatedMusicalArtist, associatedBand, musicalArtist, musicalBand, album*$\}$.

In Figure 3.b the prediction quality with increasing rank used in RESCAL and TypeConCF are shown. Clearly, the proposed method performs much better than RESCAL, especially at ranks below 200 (RESCAL: 0.663, TypeConCF: 0.770). As observed with the Cora dataset, RESCAL's prediction quality starts to catch up at higher ranks and it is expected to hit a comparable performance at some very high rank. RESCAL reaches a score of 0.734 at a rank of 400 (TypeConCF: 0.792). Just the factor matrices require approximately 1 GB of memory at this rank. On the other hand, the factorization with TypeConCF reaches the same score approximately at a rank of 100 (consuming about 4 times less memory).

When comparing the runtime of both methods (Figure 3.e), it can be observed that in contrast to the observations for the Cora dataset, the runtime of TypeConCF is comparable to RESCAL, and even slightly starts to outperform it at ranks higher than 300. As already discussed in Section IV, the benefits on calculating the updates based on the much smaller, relation specific factor matrices $A_{\hat{S}_k}$ and $A_{\hat{O}_k}$ become more prominent. With increasing rank, these smaller matrices grow much slower in their absolute size than the complete factor matrix A and therefore the runtime suffers less from an increase in rank (and even compensates for the more complex inverse calculation in the A-updates).

Note, that TypeConCF clearly outperforms RESCAL when additionally considering the prediction quality in the runtime comparison. RESCAL reaches an AUPRC score of 0.734 using a rank of 400 after approximately one hour, where TypeConCf needed only 9 minutes (rank 100).

C. DBPedia

Especially when dealing with a dataset of this size (the partially observed tensor is of shape 2,255,018 $\times$ 2,255,018 $\times$ 511), is important to use models which achieve high prediction quality with low rank, since every additional dimension in the factorization consumes approximately 18 MB of additional memory. Note, that we are now dealing with a high amount of relations (511) where entities can be linked by multiple relations. Most collective matrix factorization approaches would not be applicable in this case.

In Figure 3.c it can be observed, that due to the sparsity of the tensor (6.52×10^{-9} %), RESCAL has problems to capturing data structure and constructs a factorization of low quality at a very low rank (0.345 AUPRC, rank 50). At a rank of 5 the performance is barely above a random ranking (RESCAL: 0.119, Random: 0.112)[6], where TypeConCF results already in a factorization that clearly captures structure in the data (AUPRC: 0.511). Furthermore, at a rank of 50, the

[6]We observed that the coefficients in the factor matrices A and the core tensor $\mathcal{R}$ for RESCAL were all very close to zero, meaning that RESCAL indeed tried to fit the overwhelming amount of zeros in the data.

factorization starts to become of reasonable quality (0.667[7])) where RESCAL clearly fails (0.345).

On a multi-relational dataset of this size, the calculations in the updates based on the very much smaller factor matrices become very efficient, and with increasing rank the gap in runtime is expected to increase further, since the absolute size of the smaller factor matrices $A_{\hat{S}_k}$ and $A_{\hat{O}_k}$ increases slower when compared to the complete factor matrix A. This can be exactly observed in Figure 3.f. At a rank of 50, RESCAL needs almost 16 hours for the factorization (Figure 3.f) and results in a low quality factorization (AUPRC:0.345), where TypeConCF needs only 3.5 hours and constructs a meaningful factorization of the data (AUPRC: 0.667) at a very low-rank.

VI. Conclusion

We have proposed a general collective factorization approach for large type-constrained multi-relational data that is able to exploit relational type-constraints during factorization. Our experiments showed that in addition to faster convergence we obtained better prediction accuracy and the approach uses less memory, if compared to RESCAL. More precisely, we demonstrated on all datasets that the link prediction quality of the proposed model increased significantly faster with the rank of the factorization than with RESCAL, which generally needed a significantly higher rank to achieve similar AUPRC scores for the link prediction tasks. In case of the very large DBpedia datasets with millions of entities and hundreds of relations, our method was able to make meaningful predictions on a very low-rank of 50, a rank where RESCAL failed.

In addition, we showed through our experiments that useful type-constraints can be approximated by defining them based on actual observed subject and object entities in each relation in absence of prior knowledge.

References

[1] M. Nickel, V. Tresp, and H.-P. Kriegel, "A three-way model for collective learning on multi-relational data," in *ICML*, 2011, pp. 809–816.

[2] X. L. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang, "Knowledge vault: A web-scale approach to probabilisitic knowledge fusion," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 601–610.

[3] A. P. Singh and G. J. Gordon, "Relational learning via collective matrix factorization," in *KDD*, 2008, pp. 650–658.

[4] C. Lippert, S. H. Weber, Y. Huang, V. Tresp, M. Schubert, and H.-P. Kriegel, "Relation-Prediction in Multi-Relational Domains using Matrix-Factorization," in *NIPS 2008 Workshop: Structured Input - Structured Output*, 2008.

[5] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King, "Recommender systems with social regularization," in *WSDM*, 2011, pp. 287–296.

[6] E. Acar, T. G. Kolda, and D. M. Dunlavy, "All-at-once optimization for coupled matrix and tensor factorizations," *CoRR*, vol. abs/1105.3422, 2011.

[7] R. P. Adams, G. E. Dahl, and I. Murray, "Incorporating side information in probabilistic matrix factorization with gaussian processes," in *UAI*, 2010, pp. 1–9.

[8] S. Zhu, K. Yu, Y. Chi, and Y. Gong, "Combining content and link for classification using matrix factorization," in *SIGIR*, 2007, pp. 487–494.

[9] K. Takeuchi, K. Ishiguro, A. Kimura, and H. Sawada, "Non-negative multiple matrix factorization," in *IJCAI*, 2013.

[7]The corresponding Area under ROC curve is 0.84

[10] Y. Zhang, B. Cao, and D.-Y. Yeung, "Multi-domain collaborative filtering," in *UAI*, 2010, pp. 725–732.

[11] D. Agarwal, B.-C. Chen, and B. Long, "Localized factor models for multi-context recommendation," in *KDD*, 2011, pp. 609–617.

[12] G. Bouchard, D. Yin, and S. Guo, "Convex collective matrix factorization," in *AISTATS*, 2013, pp. 144–152.

[13] K. Takeuchi, R. Tomioka, K. Ishiguro, A. Kimura, and H. Sawada, "Non-negative multiple tensor factorization," in *ICDM*, 2013, pp. 1199–1204.

[14] T. G. Kolda, B. W. Bader, and J. P. Kenny, "Higher-order web link analysis using multilinear algebra," in *ICDM*, 2005, pp. 242–249.

[15] T. Franz, A. Schultz, S. Sizov, and S. Staab, "Triplerank: Ranking semantic web data by tensor decomposition," in *International Semantic Web Conference*, 2009, pp. 213–228.

[16] S. Rendle, L. B. Marinho, A. Nanopoulos, and L. Schmidt-Thieme, "Learning optimal ranking with tensor factorization for tag recommendation," in *KDD*, 2009, pp. 727–736.

[17] V. Tresp, Y. Huang, M. Bundschus, and A. Rettinger, "Materializing and querying learned knowledge," in *First ESWC Workshop on Inductive Reasoning and Machine Learning on the Semantic Web (IRMLeS 2009)*, 2009.

[18] H. Wermser, A. Rettinger, and V. Tresp, "Modeling and learning context-aware recommendation scenarios using tensor decomposition," in *ASONAM*, 2011, pp. 137–144.

[19] A. Bordes, J. Weston, R. Collobert, and Y. Bengio, "Learning structured embeddings of knowledge bases," in *AAAI*, 2011.

[20] R. Jenatton, N. L. Roux, A. Bordes, and G. Obozinski, "A latent factor model for highly multi-relational data," in *NIPS*, 2012.

[21] D. Krompaß, M. Nickel, X. Jiang, and V. Tresp, "Non-negative tensor factorization with rescal," in *ECML/PKDD 2013 Workshop on Tensor Methods for Machine Learning*, 2013.

[22] M. Nickel and V. Tresp, "Logistic tensor factorization for multi-relational data," in *Structured Learning: Inferring Graphs from Structured and Unstructured Inputs (ICML WS)*, 2013.

[23] R. Socher, D. Chen, C. D. Manning, and A. Y. Ng, "Reasoning with neural tensor networks for knowledge base completion," in *NIPS*, 2013, pp. 926–934.

[24] K. Sengupta, A. A. Krisnadhi, and P. Hitzler, "Local closed world semantics: Grounded circumscription for owl," in *Proceedings of the 10th International Conference on The Semantic Web - Volume Part I*, ser. ISWC'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 617–632. [Online]. Available: http://dl.acm.org/citation.cfm?id=2063016.2063056

[25] B. London, T. Rekatsinas, B. Huang, and L. Getoor, "Multi-relational learning using weighted tensor decomposition with modular loss," *CoRR*, vol. abs/1303.1733, 2013.

[26] M. Nickel, V. Tresp, and H.-P. Kriegel, "Factorizing yago: scalable machine learning for linked data," in *WWW*, 2012, pp. 271–280.

[27] M. Nickel, "Tensor factorization for relational learning," PhDThesis, Ludwig-Maximilian-University of Munich, Aug. 2013. [Online]. Available: http://edoc.ub.uni-muenchen.de/16056/

[28] A. J. Laub, *Matrix analysis - for scientists and engineers*. SIAM, 2005.

[29] B. W. Bader, R. A. Harshman, and T. G. Kolda, "Temporal analysis of semantic graphs using asalsan," *2013 IEEE 13th International Conference on Data Mining*, vol. 0, pp. 33–42, 2007.

[30] L. R. Tucker, "Some mathematical notes on three-mode factor analysis," *Psychometrika*, vol. 31, pp. 279–311, 1966.

[31] R. A. Harshman., "Parafac2: Mathematical and technical notes," *UCLA working papers in phonetics*, vol. 22, pp. 30–47, 1972.

[32] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer, "DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia," *Semantic Web Journal*, 2014.