A Three-Way Model for Collective Learning on Multi-Relational Data

Maximilian Nickel NICKEL@CIP.IFI.LMU.DE
Ludwig-Maximilians-Universität, Munich, Germany

Volker Tresp VOLKER.TRESP@SIEMENS.COM
Siemens AG, Corporate Technology, Munich, Germany

Hans-Peter Kriegel KRIEGEL@DBS.IFI.LMU.DE
Ludwig-Maximilians-Universität, Munich, Germany

Abstract

Relational learning is becoming increasingly important in many areas of application. Here, we present a novel approach to relational learning based on the factorization of a three-way tensor. We show that unlike other tensor approaches, our method is able to perform collective learning via the latent components of the model and provide an efficient algorithm to compute the factorization. We substantiate our theoretical considerations regarding the collective learning capabilities of our model by the means of experiments on both a new dataset and a dataset commonly used in entity resolution. Furthermore, we show on common benchmark datasets that our approach achieves better or on-par results, if compared to current state-of-the-art relational learning solutions, while it is significantly faster to compute.

1. Introduction

With the growing relevance of relational data and network data in such diverse areas as social network modeling, the semantic web, bioinformatics and artificial intelligence, the field of relational learning is increasing in importance. This paper is concerned with the application of tensors to relational learning. Tensors and their decompositions are widely used in fields like psycho- or chemometrics and more recently have also been applied to data mining and machine learning problems, e.g. for modelling temporal effects in social networks. In relational learning, tensors have just emerged to be used as an alternative to more common approaches like graphical models. The motivation to use tensors in this domain is manifold. From a modelling perspective tensors provide simplicity, as multiple relations of any order can be expressed straightforwardly as a higher-order tensor. Moreover, no a priori knowledge about the structure of the problem needs to be known or needs to be inferred from the data, as it is necessary e.g. for graphical models like Bayesian Networks or Markov Logic Networks (MLN). A reason to employ tensor decompositions from a learning perspective is that relational domains are usually high-dimensional and sparse, a setting where factorization methods have shown very good results.

An important property of relational data is that correlations can be produced between multiple interconnected nodes. These correlations can be captured by including the attributes, relations or classes of related entities in a learning task. However, well-known tensor factorization approaches such as CANDECOMP/PARAFAC (CP) (Harshman & Lundy, 1994) or Tucker (Tucker, 1966) cannot model this collective learning effect sufficiently. The DEDICOM decomposition (Harshman, 1978) is capable of detecting this type of correlations, but unfortunately, it puts constraints on the model that are not reasonable for relational learning in general and thus leads to suboptimal results. In this paper we propose the relational learning approach RESCAL which is based on a tensor factorization that is related to DEDICOM but does not exhibit the same constraints. In doing so we are able to derive models of higher quality and with significant runtime improvements. We will present an efficient algorithm for computing the factorization and evaluate our approach on both a new collective learning dataset and relational learning benchmark datasets. We will show that significant improvements can be achieved by using our model compared to more commonly used tensor models like CP. Furthermore, we will also show that our approach gives better or similar results if compared to current state-of-the-art approaches in relational learning on these datasets, while only taking a fraction of the time to

Appearing in *Proceedings of the 28^{th} International Conference on Machine Learning*, Bellevue, WA, USA, 2011.

compute.

2. Modelling and Notation

The modelling approach of relational domains in this paper is the following. To represent dyadic relational data we make use of the semantic web's RDF formalism where relations are modeled as triples of the form `(subject, predicate, object)` and where a predicate either models the relationship between two entities or between an entity and an attribute value. In order to model dyadic relational data as a tensor, we employ a three-way tensor $\mathcal{X}$, where two modes are identically formed by the concatenated entities of the domain and the third mode holds the relations.[1] Figure 1 provides an illustration of this modelling method. A tensor entry $\mathcal{X}_{ijk} = 1$ denotes the fact that there exists a relation `(i-th entity, k-th predicate, j-th entity)`. Otherwise, for non-existing and unknown relations, the entry is set to zero.

这只是一种KG的表示方法

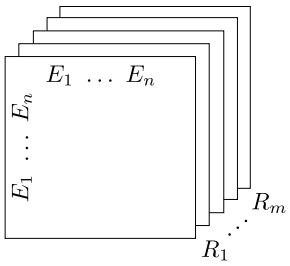


Figure 1: Tensor model for relational data. $E_1 \cdots E_n$ denote the entities, while $R_1 \cdots R_m$ denote the relations in the domain

In the remainder of this paper we will use the following notation: Tensors are represented by calligraphic letters $\mathcal{X}$, while $\mathcal{X}_k$ refers to the k-th frontal slice of the tensor $\mathcal{X}$. $X_{(n)}$ denotes the unfolding of the tensor $\mathcal{X}$ in mode n. $A \otimes B$ refers to the Kronecker product of the matrices A and B. $\mathbf{I}_k$ denotes the identity matrix of size k and $\mathbf{vec}(X)$ is the vectorization of the matrix X. Vectors are represented by bold lowercase letters, e.g. $\boldsymbol{a}$. Furthermore, it is assumed that the data is given as a $n \times n \times m$ tensor $\mathcal{X}$, where n is the number of entities and m the number of relations.

3. Related Work

The literature on statistical relational learning is vast, thus we will only give a very brief overview. A common approach to relational learning is to employ graphical models such as Bayesian Networks or Markov Logic networks (Friedman et al., 1999; Richardson & Domingos, 2006). Moreover, IHRM (Xu et al., 2006) and IRM (Kemp et al., 2006) are nonparametric Bayesian approaches to relational learning, while (Singh & Gordon, 2008) employ collective matrix factorizations for relational learning. (Getoor & Taskar, 2007) presents further approaches and gives a detailed introduction into the area of relational learning. Tensor decompositions have been widely applied to fields like psycho- and chemometrics. An extensive review of tensor decompositions and their applications can be found in (Kolda & Bader, 2009). Recently, (Sutskever et al., 2009) introduced the Bayesian Clustered Tensor Factorization (BCTF) and applied it to relational data. (Sun et al., 2006) presents methods for dynamic and streaming tensor analysis. These methods are used to analyse network traffic and bibliographic data. (Franz et al., 2009) use CP to rank relational data that is given in form of RDF triples.

[1] Please note that we don't assume homogeneous domains, thus the entities of one mode can be instances of multiple classes like persons, items, places, etc.

4. Methods and Theoretical Aspects

Relational learning is concerned with domains where entities are interconnected by multiple relations. Hence, correlations can not only occur directly between entities or relations, but also across these interconnections of different entities and relations. Depending on the learning task at hand, it is known that it can be of great benefit when the learning algorithm is able to detect these relational learning specific correlations reliably. For instance, consider the task of predicting the party membership of a president of the United States of America. Without any additional information, this can be done quite accurately when the party of the president's vice president is known, since both persons have mostly been members of the same party. To include information such as attributes, classes or relations of connected entities to support a classification task is commonly referred to as collective classification. However, this procedure can not only be useful in classification problems, but also in entity resolution, link prediction or any other learning task on relational data. We will refer to the mechanism of exploiting the information provided by related entities regardless of the particular learning task at hand as *collective learning*.

4.1. A Model for Multi-Relational Data

对三维张量进行分解

To perform collective learning on multi-relational data, we propose RESCAL, an approach which uses a tensor factorization model that takes the inherent structure of relational data into account. More precisely, we employ the following rank-r factorization, where each slice $\mathcal{X}_k$ is factorized as

n × r 的矩阵，表示每个实体的潜在表示(向量表示 向量嵌入为r)

$$\mathcal{X}_k \approx A R_k A^T, \text{ for } k = 1, \dots, m \qquad (1)$$

r × r 的非对称矩阵，建模第 k 个关系中的实体的潜在交互

Here, A is a $n \times r$ matrix that contains the latent-component representation of the entities in the domain and R_k is an asymmetric $r \times r$ matrix that models the interactions of the

latent components in the k-th predicate.

The factor matrices A and R_k can be computed by solving the regularized minimization problem

$$\min_{A,R_k} f(A,R_k) + g(A,R_k) \tag{2}$$

where

$$f(A,R_k) = \frac{1}{2}\left(\sum_k \|\mathcal{X}_k - AR_kA^T\|_F^2\right) \tag{3}$$

and g is the following regularization term

$$g(A,R_k) = \frac{1}{2}\lambda\left(\|A\|_F^2 + \sum_k \|R_k\|_F^2\right) \tag{4}$$

which is included to prevent overfitting of the model.

An important aspect of (1) for collective learning and what distinguishes it from other tensor factorizations like CP or even BCTF is that the entities of the domain have a unique latent-component representation, regardless of their occurrence as subjects or objects in a relation, as they are represented both times by the matrix A. The effect of this modelling becomes more apparent by looking at the element-wise formulation of (3), namely

$$f(A,R_k) = \frac{1}{2}\sum_{i,j,k}\left(\mathcal{X}_{ijk} - \boldsymbol{a}_i^T R_k \boldsymbol{a}_j\right)^2$$

Here, $\boldsymbol{a}_i$ and $\boldsymbol{a}_j$ denote the i-th and j-th row of A and thus are the latent-component representations of the i-th and j-th entity. By holding $\boldsymbol{a}_j$ and R_k fixed, it is clear that the latent-component representation $\boldsymbol{a}_i$ depends on $\boldsymbol{a}_j$ as well as the existence of the triple `(i-th entity, k-th predicate, j-th entity)` represented by $\mathcal{X}_{ijk}$. Moreover, since the entities have a unique latent-component representation, $\boldsymbol{a}_j$ holds also the information which entities are related to the j-th entity as subjects and objects. Consequently, all direct and indirect relations have a determining influence on the calculation of $\boldsymbol{a}_i$. Just as the entities are represented by A, each relation is represented by the matrix R_k, which models how the latent components interact in the respective relation and where the asymmetry of R_k takes into account whether a latent component occurs as a subject or an object.

For a short illustration of this mechanism, consider the example shown in Figure 2. The latent-component representations of `Al` and `Lyndon` will be similar to each other in this example, as both representations reflect that their corresponding entities are related to the object `Party X`. Because of this, `Bill` and `John` will also have similar latent-component representations. Consequently, the product $\boldsymbol{a}_{\texttt{Bill}}^T R_{\texttt{party}} \boldsymbol{a}_{\texttt{PartyX}}$ will yield a similar value to

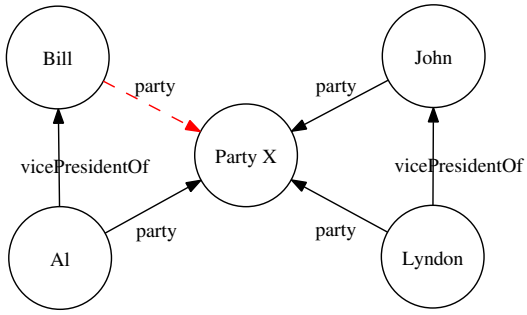


Figure 2: Visualization of a subgraph of the relational graph for the US presidents example. The relation marked red is unknown.

$\boldsymbol{a}_{\texttt{John}}^T R_{\texttt{party}} \boldsymbol{a}_{\texttt{PartyX}}$ and as such the missing relation can be predicted correctly. Please note that this information propagation mechanism through the latent components would break if `Bill` and `John` would have different representations as subjects and objects.

4.2. Connections to other Tensor Factorizations

The model specified in (1) can be considered a relaxed version of DEDICOM or equivalently, an asymmetric extension of IDIOSCAL. The rank-r DEDICOM decomposition of a three-way tensor $\mathcal{X}$ is given as: $\mathcal{X}_k \approx AD_kRD_kA^T$, for $k = 1,\ldots,m$. Here, A is a $n \times r$ matrix that contains the latent components and R is an asymmetric $r \times r$ matrix that models the *global* interactions of the latent components. The diagonal $r \times r$ matrix D_k models the participation of the latent components in the k-th predicate. Thus, DEDICOM is suitable when there is one global interaction model for the latent components and its variation across the third mode can be described by diagonal factors. Examples where this is a reasonable assumption include international trade or communication patterns across time, as presented in (Bader et al., 2007). However, for multi-relational data this assumption is often too restricted.

Furthermore, the model (1) can be regarded as a restricted Tucker3 model. Let $X_{(n)} = AG_{(n)}(C \otimes B)^T$ be the matricized form of the Tucker3 decomposition of $\mathcal{X}$. (1) is then equivalent to a Tucker3 model, where the factors B and C are constrained to $B = A$ and $C = \mathbf{I}_k$, while G is holding the slices R_k.

4.3. Computing the Factorization

In order to compute the factor matrices for (1), equation (2) could be solved directly with any nonlinear optimization algorithm. However, to improve computational efficiency we take an alternating least-squares (ALS) approach and exploit the connection of (1) to DEDICOM, by using the very efficient ASALSAN (Bader et al., 2007) algorithm as a starting point and adapting it to our model. In particular,

we employ the following algorithm:

Update A: To compute the update step of A we use the same approach as ASALSAN and solve the problem approximately for left and right A simultaneously for all frontal slices of $\mathcal{X}$. The data is stacked side by side for this, such that

$$\bar{X} = A\bar{R}(\mathbf{I}_{2m} \otimes A^T) \tag{5}$$

where

$$\bar{X} = (X_1\ X_1^T \dots X_m\ X_m^T)$$
$$\bar{R} = (R_1\ R_1^T \dots R_m\ R_m^T)$$

We approximate this nonlinear problem by solving only for the left A while holding the right A constant. As (Bader et al., 2007) points out, this approach seems to be viable because of the construction of $\bar{X}$. Let $\bar{A}$ denote the constant right hand A. Then, the gradient of (5) is given by

$$\frac{\partial f}{\partial A} = \bar{R}[(\mathbf{I} \otimes \bar{A}^T\bar{A})\bar{R}^T A^T - (\mathbf{I} \otimes \bar{A}^T)\bar{X}^T] + \lambda A^T$$

By using the method of normal equations and setting this gradient to zero, the update of A can be computed by

$$A \leftarrow \left[\sum_{k=1}^{m} X_k A R_k^T + X_k^T A R_k\right] \left[\sum_{k=1}^{m} B_k + C_k + \lambda\mathbf{I}\right]^{-1}$$

where

$$B_k = R_k A^T A R_k^T, \quad C_k = R_k^T A^T A R_k$$

Update R_k: By holding A fixed and vectorizing X_k as well as R_k, the minimization problem (3) can be cast as

$$f(R_k) = \|\mathbf{vec}(X_k) - (A \otimes A)\mathbf{vec}(R_k)\| + \lambda\|\mathbf{vec}(R_k)\|$$

what is a regularized linear regression problem and can be solved accordingly, i.e. by

$$R_k \leftarrow \left(Z^T Z + \lambda\mathbf{I}\right)^{-1} Z\mathbf{vec}(X_k)$$

where $Z = A \otimes A$.

The starting points for A and R_k can either be random matrices or A is initialized from the eigendecomposition of $\sum_k (X_k + X_k^T)$. To compute the factor matrices, the algorithm performs alternating updates of A and all R_k until $\frac{f(A,R_k)}{\|\mathcal{X}\|_F^2}$ converges to some small threshold ϵ or a maximum number of iterations is exceeded.

One benefit of RESCAL is that it can be computed efficiently. Similar to the ASALSAN algorithm, it is possible to use the QR decomposition of A and X_k for updating R_k such that A and X_k are only $r \times r$ matrices. In doing so, updating R_k becomes independent from the number of entities and is only dependent on the complexity of the model. Despite its similarities to DEDICOM, there are significant computational advantages of our approach, as a simpler model is computed. Evidently, (1) has no D_k factors, a problem that has to be solved in ASALSAN by Newton's method. Moreover, in RESCAL the term $\left(Z^T Z\right)^{-1} Z$ isn't dependent on k and thus only needs to be computed once before updating all R_k. Computing the inverse of $\left(Z^T Z + \lambda\mathbf{I}\right)$ is the most expensive operation in updating R_k, as Z is the result of a Kronecker product and thus can become very large. However, in case R_k is not regularized, this can be simplified. Since $(A \otimes B)(C \otimes D) = AC \otimes BD$ and $(A \otimes B)^{-1} = (A^{-1} \otimes B^{-1})$ (Horn & Johnson, 1994) it holds that

$$\left(Z^T Z\right)^{-1} = \left(A^T A\right)^{-1} A \otimes \left(A^T A\right)^{-1} A$$

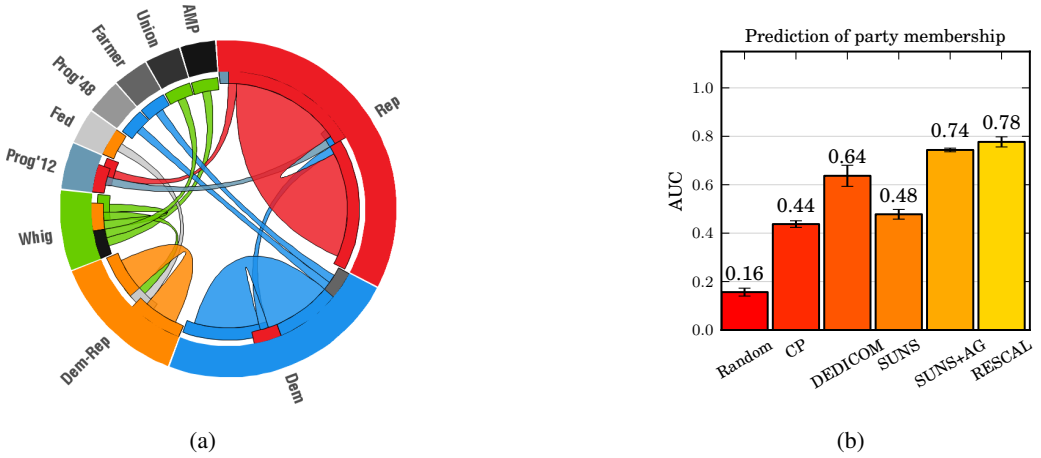
Here, the inverse only has to be computed for the much smaller matrix $A^T A$. Furthermore, since m occurs only as a linear factor, the algorithm will scale linearly with the number of predicates. But the simplicity of the model comes also at a cost. It can be seen from (1) and the representation as a Tucker3 model that no rank reduction is performed on the mode that holds the relations, what potentially can have a negative effect when the relations are noisy and should be aggregated simultaneously with the entities.

4.4. Solving Relational Learning Tasks

Canonical relational learning tasks can be approached with RESCAL in the following way. To predict whether a link exists between two entities e_i, e_j for the k-th predicate, it is sufficient to look at the rank-reduced reconstruction $\hat{\mathcal{X}}_k = AR_kA^T$ of the appropriate slice $\mathcal{X}_k$. In doing so, *link prediction* can be done by comparing $\hat{\mathcal{X}}_{ijk} > \theta$ to some given threshold θ or by ranking the entries according to their likelihood that the link in question exists.

Collective classification can be cast as a subtask of link prediction, as the class of an entity can be modelled by introducing a class relation and including the classes as entities in the tensor. Thus, the classification problem can also be solved by reconstructing the appropriate slice of the class relation. The collectivity of the classification, as for all other learning tasks, is given automatically by the structure of the model.

Furthermore, the matrix A can be regarded as the entity-latent-component space that is created by the decomposition. *Link-based clustering* can be performed by clustering the entities in this entity-latent-component space. In doing so, the similarity between entities is computed upon their similarity across multiple relations.



(a) (b)

Figure 3: (a) is a visualization of the `presidentOf` relation on the US presidents dataset. The size of the ring segments indicates how many persons in the dataset are members of the respective party. An arc indicates a `presidentOf` relation and the size of an arc indicates how often this relation occurs between the connected segments. (b) shows the results of 10-fold cross-validation on this data.

5. Evaluation

In the following we evaluate RESCAL in comparison to standard tensor factorizations and relational learning algorithms on various datasets. All algorithms have been implemented in Python and evaluated on a Intel Core 2 Duo 2.5GHz with 4GB RAM. For CP we implemented the CP-ALS algorithm of the Tensor Toolbox (Bader & Kolda, 2010) in Python/NumPy.

5.1. Collective Classification

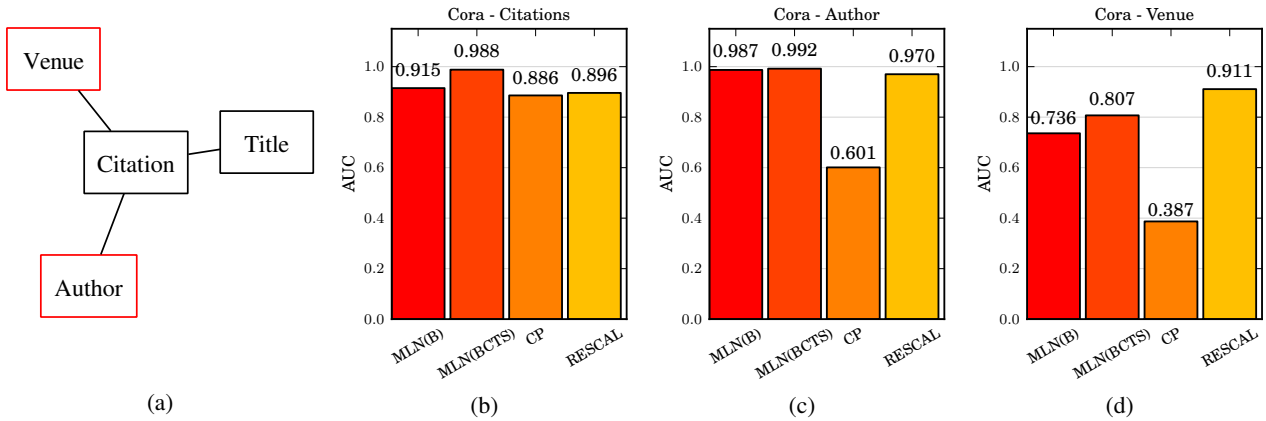
The first experiment in this evaluation is designed to assess, in a controlled setting, the collective learning capabilities of our approach as discussed in Section 4. For this purpose, we created a dataset for the US presidents example, by retrieving the names of all presidents and vice presidents of the United States from DBpedia, in combination with their party membership and the `presidentOf`/`vicePresidentOf` information.[2] The objective of the experiment is to predict the party membership for each person in a collective classification setting. It can be seen from Figure 3(a), that a president and his/her vice president have mostly been members of the same party. Hence, the ability to perform collective learning by including the party of a related person should greatly improve the classification results, especially as the dataset contains no further information other than the three relations that could help in predicting the correct party. From the raw RDF data we constructed a $93 \times 93 \times 3$ tensor, where the entity modes correspond to the joint set of persons and parties and the third mode is formed by the three relations. In addition to RESCAL we evaluated the tensor factorizations CP and DEDICOM. Moreover, we included the SUNS algorithm (Huang et al., 2010) in the evaluation, a relational learning approach for large scale data. Despite SUNS' ability to handle large data sizes, its capabilities for collective learning are limited. Therefore, SUNS is a good indicator what improvement can be gained in collective learning. For the same reason we included Aggregated SUNS (SUNS+AG), which mimics collective learning and thus is the counterpart of SUNS without aggregation.[3] To evaluate the algorithms we've conducted cross-validation by partitioning all persons into ten folds and deleting the party membership information of the persons in the test fold. In the case of RESCAL, CP and DEDICOM we computed a rank-reduced factorization, ranked all parties by their entries in the reconstructed party-membership-slice and recorded the area under the precision-recall curve. It can be seen from Figure 3(b) that aggregation improves the SUNS model significantly. The results of RESCAL and DEDICOM outperform both CP and SUNS and show clearly the usefulness of our approach for domains where collective learning is an important feature. It can also be seen that CP isn't capable of collective learning, as it yields similar results to SUNS. There is also a significant difference between the results of RESCAL and DEDICOM, what indicates that, even on this small dataset with highly correlated predicates, the constraints imposed by DEDICOM are to restrictive.

5.2. Collective Entity Resolution

The US presidents example demonstrated the capabilities of our approach on a small, specially created dataset. In

[2] Dataset available at `http://www.cip.ifi.lmu.de/~nickel`

[3] Here, aggregation means that the party membership information of the related person has been added manually as a new relation to each statistical unit



(a) (b) (c) (d)

Figure 4: (a) shows the structural representation of the Cora dataset. (b)-(d) show the results for the area under the precision-recall curve for 5-fold entity resolution runs on this dataset

turn, the Cora dataset is a larger real-world dataset which is commonly used to assess the performance of relational learning approaches in tasks like collective classification or entity resolution. It is especially interesting as it is known that applying collective learning to this dataset can improve the quality of the model significantly (Sen et al., 2008). Cora exists in different versions, here we use the dataset and experimental setup as described in (Singla & Domingos, 2007) and also compare our results to those reported in that publication. From the raw data we constructed a tensor of size $2497 \times 2497 \times 7$.

The objective of this experiment is to perform entity resolution, as described in the following: While MLNs have to treat entity resolution as a link prediction problem, i.e. by predicting the probability of the triple `(x, isEqual, y)`, tensor factorizations allow us take a different approach. We derive the likelihood that two entities are identical from their similarity in the entity-latent-component space A. More specifically, we normalize the rows of A, such that each row represents the normalized participation of the corresponding entity in the latent components. Based on this representation we compute the similarity between two entities x, y by using the heat kernel $k(x, y) = e^{-\|x-y\|^2/\delta}$, where δ is a user-given constant and use this similarity score as a measure for the likelihood that x and y refer to the same entity. This is a relative ad hoc approach to entity resolution, but the focus of this experiment is again rather on assessing the collective learning capabilities of our approach than conducting a full entity resolution experiment. We are mainly interested whether our approach can produce results roughly similar to MLNs and how they compare to CP. Figure 4 shows the results of our evaluation, which are especially interesting in combination with the structural representation of the data. Since the Citation class is the central node in Figure 4 and the other classes aren't interconnected, it is sufficient to look only at directly connected entities in order to perform entity resolution for citations. But the citations are noisy themselves. Therefore, when entity resolution has to be done for an author or a venue, it is very helpful for an algorithm when it can include the entities that are connected to an author's or venue's citation in its evaluation of the entity. This circumstance is reflected in the evaluation results in Figure 4. For citations there is no significant difference between RESCAL and CP. For authors and venues however, RESCAL gives far better results than CP, as it can perform collective learning. In the case of venues, it shows even better results than the much more complex MLN.

5.3. Kinships, Nations and UMLS

In order to evaluate how well RESCAL is performing compared to current state-of-the-art relational learning solutions, we applied it to the Kinships, Nations and UMLS datasets used in (Kemp et al., 2006) and compared the area under the precision-recall curve (AUC) to the results of BCTF, IRM and MRC published in (Sutskever et al., 2009) and (Kok & Domingos, 2007). Due to space constraints we refer to (Kemp et al., 2006) for a detailed description of the datasets. In order to get comparable results to BCTF, IRM and MRC we followed their experimental protocol and performed 10-fold cross-validation by using `(subject, predicate, object)` triples as the statistical unit. For a comparison to standard tensor decompositions we've included CP and DEDICOM in the evaluation. The task for all three datasets is link prediction. Figure 5 shows the results of our evaluation. It can be seen that RESCAL gives comparable or better results than BCTF, IRM and MRC on all datasets.

At last we want to briefly demonstrate the link-based clustering capabilities of RESCAL. Therefore, we computed a rank-20 decomposition of the Nations dataset and applied

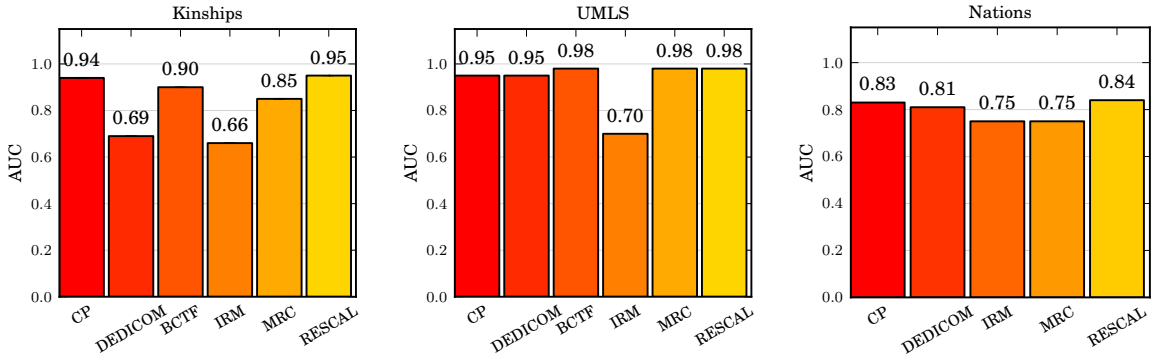


Figure 5: Link prediction results for the Kinships, Nations and UMLS datasets

k-means with $k = 6$ to the matrix A. It can be seen from Figure 6 that in doing so, similar clusters as in (Kemp et al., 2006) are obtained. The countries are partitioned into one group containing countries from the communist bloc, two groups from the western bloc, where Brazil is separated from the rest, and three groups for the neutral bloc. The six relations shown in Figure 6 indicate that this is a reasonable partitioning of the data.

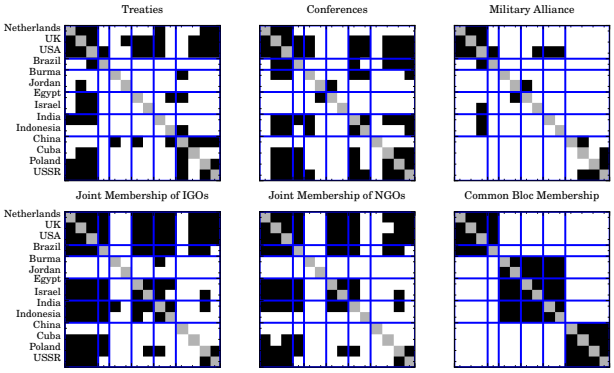


Figure 6: A clustering of countries in the Nations dataset. Black squares indicate an existing relation between the countries. Gray squares indicate missing values.

5.4. Runtime Performance and Technical Considerations

Relational datasets can become large very quickly, hence runtime if often a critical issue in real world problems. We recorded the runtime of CP, DEDICOM and regularized RESCAL on various datasets for different ranks. Table 1 shows the results of this evaluation. All algorithms have been started multiple times from random initializations and their convergence tolerance was set to $\epsilon = 10^{-5}$. Despite the efficiency of the ASALSAN algorithm, it is evident that RESCAL often gives a huge improvement over DEDICOM in terms of runtime performance. This is mainly due to two reasons. The simpler model of RESCAL allows significant improvements with regard to the computational complexity, as discussed in Section 4. But interestingly, RESCAL shows even runtime improvements in comparison to CP, although CP-ALS is theoretically faster to compute. We believe that this is the case because the RESCAL model is more suitable for relational data which is indicated by a faster convergence rate, as for purely random tensors of arbitrary size CP-ALS is consistently faster than RESCAL. However, it should also be noted that RESCAL often scales worse with the regard to the rank than CP-ALS, in particular for regularized problems. In comparison to MRC and IRM it is the case that RESCAL as well as CP-ALS show much faster training times. (Kok & Domingos, 2007) states that IRM and MRC have been run at least ten hours per fold on the IRM datasets, while both tensor decompositions show training times below 2 minutes per fold. Unfortunately there is no information about runtime performance or publicly available code for BCTF.

Table 1: Runtime comparison on various datasets. |E| denotes the number of entities, |R| the number of relations in the data. The symbol - indicates that the algorithm didn't converge.

Dataset	Algorithm	Total Runtime		
		Rank		
		10	20	40
Kinships	CP-ALS	6.4s	25.4s	105.8s
|E|: 104, |R|: 26	ASALSAN	527s	1549s	16851s
	RESCAL	**1.1s**	**3.7s**	**51.2s**
Nations	CP-ALS	16.4s	43.8s	68.3s
|E|: 125, |R|: 57	ASALSAN	830s	4602s	42506s
	RESCAL	**1.7s**	**5.3s**	**54.4s**
UMLS	CP-ALS	5.5s	11.7s	**53.9s**
|E|: 135, |R|: 49	ASALSAN	1706s	4846s	6012s
	RESCAL	**2.6s**	**4.9s**	72.3s
Cora	CP-ALS	369s	934s	3190s
|E|: 2497, |R|: 7	ASALSAN	**132s**	**154s**	-
	RESCAL	364s	348s	**680s**

At last we want to make a case for the ease of use of RESCAL. Its algorithm only involves standard matrix operations and has been implemented in Python/NumPy without any additional software in less than 120 lines of code. An efficient implementation of CP-ALS on the other hand requires special tensor operations and data structures like the Khatri-Rao product or the Kruskal tensor.

6. Conclusion and Future Work

RESCAL is a tensor factorization approach to relational learning which is designed to account for the inherent structure of dyadic relational data. In doing so, our approach is able to perform collective learning via the latent components of the factorization. The results on various datasets as well as the runtime performance are very competitive and show that tensors in general and RESCAL specifically are promising new approaches to relational learning. Currently we intend to investigate different extensions to our approach. In order to obtain highly scalable solutions, we are looking into distributed versions of RESCAL as well as a stochastic gradient descent approach to the optimization problem. Furthermore, to improve both the predictive performance and the runtime behaviour of RESCAL, we also plan to exploit constraints like typed relations while computing the factorization.

Acknowledgements

We thank the reviewers for their helpful comments. We acknowledge funding by the German Federal Ministry of Economy and Technology (BMWi) under the THESEUS project and by the EU FP 7 Large-Scale Integrating Project LarKC.

References

Bader, Brett W. and Kolda, Tamara G. MATLAB tensor toolbox version 2.4, March 2010. URL http://csmr.ca.sandia.gov/~tgkolda/TensorToolbox/.

Bader, Brett W., Harshman, Richard A., and Kolda, Tamara G. Temporal analysis of semantic graphs using ASALSAN. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pp. 33–42, Omaha, NE, USA, 2007. doi: 10.1109/ICDM.2007.54. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4470227.

Franz, T., Schultz, A., Sizov, S., and Staab, S. Triplerank: Ranking semantic web data by tensor decomposition. *The Semantic Web-ISWC 2009*, pp. 213–228, 2009.

Friedman, N., Getoor, L., Koller, D., and Pfeffer, A. Learning probabilistic relational models. In *International Joint Conference on Artificial Intelligence*, volume 16, pp. 1300–1309, 1999.

Getoor, L. and Taskar, B. *Introduction to statistical relational learning*. The MIT Press, 2007. ISBN 0262072882.

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 for Mathematical Psychology, McMaster University, Hamilton, Ontario, August*, 1978.

Harshman, R. A and Lundy, M. E. PARAFAC: parallel factor analysis. *Computational Statistics & Data Analysis*, 18(1): 39–72, 1994. ISSN 0167-9473.

Horn, Roger A. and Johnson, Charles R. *Topics in matrix analysis*. Cambridge University Press, June 1994. ISBN 9780521467131.

Huang, Yi, Tresp, Volker, Bundschus, Markus, and Rettinger, Achim. Multivariate structured prediction for learning on semantic web. In *Proceedings of the 20th International Conference on Inductive Logic Programming (ILP)*, 2010.

Kemp, C., Tenenbaum, J. B, Griffiths, T. L, Yamada, T., and Ueda, N. Learning systems of concepts with an infinite relational model. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, pp. 381, 2006.

Kok, S. and Domingos, P. Statistical predicate invention. In *Proceedings of the 24th international conference on Machine learning*, pp. 433–440, 2007.

Kolda, Tamara G. and Bader, Brett W. Tensor decompositions and applications. *SIAM Review*, 51(3): 455—500, 2009. ISSN 00361445. doi: 10.1137/07070111X. URL http://link.aip.org/link/SIREAD/v51/i3/p455/s1&Agg=doi.

Richardson, M. and Domingos, P. Markov logic networks. *Machine Learning*, 62(1):107–136, 2006. ISSN 0885-6125.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. Collective classification in network data. *AI Magazine*, 29(3):93, 2008. ISSN 0738-4602.

Singh, A. P and Gordon, G. J. Relational learning via collective matrix factorization. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 650–658, 2008.

Singla, P. and Domingos, P. Entity resolution with markov logic. In *Data Mining, 2006. ICDM'06. Sixth International Conference on*, pp. 572–582, 2007.

Sun, J., Tao, D., and Faloutsos, C. Beyond streams and graphs: dynamic tensor analysis. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 374–383, 2006. ISBN 1595933395.

Sutskever, I., Salakhutdinov, R., and Tenenbaum, J. B. Modelling relational data using bayesian clustered tensor factorization. *Advances in Neural Information Processing Systems*, 22, 2009.

Tucker, L. R. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966. ISSN 0033-3123.

Xu, Z., Tresp, V., Yu, K., and Kriegel, H. P. Infinite hidden relational models. In *Proceedings of the 22nd International Conference on Uncertainty in Artificial Intelligence*, 2006.