Inferring Relations in Knowledge Graphs with Tensor Decompositions

Ankur Padia, Konstantinos Kalpakis and Tim Finin
Computer Science and Electrical Engineering
University of Maryland, Baltimore County
Baltimore, Maryland, USA
{*ankurpadia, kalpakis, finin*}*@umbc.edu*

***Abstract*—Multi-relational data, like knowledge graphs, are generated from multiple data sources by extracting entities and their relationships. We often want to include inferred, implicit or likely relationships that are not explicitly stated, which can be viewed as link-prediction in a graph. Tensor decomposition models have been shown to produce state-of-the-art results in link-prediction tasks. We describe a simple but novel extension to an existing tensor decomposition model to predict missing links using similarity among tensor slices, as opposed to an existing tensor decomposition models which assumes each slice to contribute equally in predicting links. Our extended model performs better than the original tensor decomposition and the non-negative tensor decomposition variant of it in an evaluation on several datasets.**

***Keywords*-Multi-relational Data, Link Prediction**

I. Introduction

Multi-relational datasets, like the one shown in Figure 1, are graphs in which multiple relationships can hold between a pair of entities. Such graphs are gaining importance as they help improve the accuracy of complex tasks such as question answering. The sources from which they are constructed, often text collections [1], may not make explicit some of relationships between entities which can lessen the performance of applications using the knowledge graphs. Hence, developing mechanisms to infer implicit relationships becomes essential.

Tensors help when represent information in multiple dimensions and can represent multi-relational data naturally. Such tensors are then factorized to obtain latent representations for the entities and their relationships. Different factorization techniques are applied to obtain ranking and link prediction [2], [3]. However, each of the tensor decomposition methods assumes that all relation participate equally to predict link. We describe a simple, yet novel, extension to the existing tensor decomposition model for link prediction using similarity among the slices, as not all relation contribute equally to predict links.

II. Notation

First, we introduce the notation used in this paper. A calligraphic letter with underline $\underline{\mathcal{X}}$ denotes a tensor, $\mathbf{X}_k$ denotes k^{th} frontal slice of the tensor $\underline{\mathcal{X}}$, bold faced letter $\mathbf{A}$ denotes a matrix, $\otimes$ denotes Kronecker product of two matrices, bold small letter $\mathbf{a}$ denotes a vector, and italic small letter a denotes a scalar. Bold faced $\mathbf{I}$ denotes an identity matrix, $vec(\mathbf{A})$ denotes the vector form of a matrix $\mathbf{A}$, $||\mathbf{A}||_F$ denotes the Frobenius norm of a matrix $\mathbf{A}$, $||\mathbf{a}||_1$ denotes l_1 norm of a vector $\mathbf{a}$. Moreover, $\underline{\mathcal{X}}$ is a tensor of dimension $N_e \times N_e \times N_r$, where N_e is the number of entities and N_r is the number of relations.

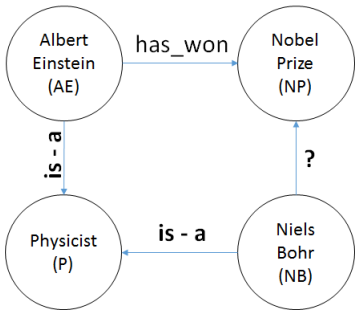


Figure 1. This simple example of a knowledge graph as multi-relation data represents entities as nodes and relations as labeled edges. A pair of entities and a relation form a triple, e.g.,"Albert Einstein has_won a Nobel prize".

III. Related Work

Link prediction with tensor decomposition can be categorized into plain tensor factorization, where no constraint is imposed on factors, and non-negative tensor factorization. Plain decomposition methods like RESCAL [4] achieve state-of-the-art results on multi-relational data and can be scaled to work on large datasets, like YAGO [5], making them good candidates for extension. However, non-negative tensor decompositions [3] introduce additional constraints, resulting in sparser factors that require more time to update, which introduces a scalability issue. Our work includes a simple, but novel, extension to the existing tensor RESCAL decomposition method to predict links using relational similarity, as [4] and [3] assume that all relation participate equally. Additional we preserve the scalability of RESCAL as there are no constraints on the obtained factors.

IV. Model Detail

In this section we describe our extension to the existing model to predict missing links in multi-relation data by

Ankur Padia, Kostantinos Kalpakis and Tim Finin, Inferring Relations in Knowledge Graphs with Tensor Decompositions, IEEE Int. Conf. on Big Data, Dec. 2016. http://ebiq.org/p/766

decomposition a tensor $\underline{\mathcal{X}}$. We solve the regularized minimization problem presented below.

$$\min_{\mathbf{A},\mathbf{R}} \sum_k \quad f(\mathbf{A}, \mathbf{R}_k) + g(\mathbf{A}, \mathbf{R}_k) + f_{sim}(\mathbf{C}, \mathbf{R}) \tag{1}$$

where,

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

$$g(\mathbf{A}, \mathbf{R}_k) = \frac{1}{2}\left(\lambda_A||\mathbf{A}||_F^2 + \lambda_r \sum_k ||\mathbf{R}_k||_F^2\right) \tag{3}$$

$$f_{sim}(\mathbf{C}, \mathbf{R}_k) = \frac{1}{2}\lambda_{sim} \sum_i \mathbf{C}_{k,i} \cdot ||\mathbf{R}_k - \mathbf{R}_i||_F^2 \tag{4}$$
$$\forall 1 \leq i \leq N_r, 1 \leq k \leq N_r$$

Here $\mathbf{A}$ is a shared $N_e \times p$ matrix where p is the dimension of latent representation of the corresponding entity. The frontal slice $\mathbf{R}_k$ is a $p \times p$ matrix that represents the interaction of all entities with respect to the kth relationship. $\mathbf{C}$ is a $N_r \times N_r$ similarity matrix where each element of the matrix is a similarity score between two slices of the tensor. As in [4], the objective of our model is to factor a given tensor $\underline{\mathcal{X}}$ into a shared matrix $\mathbf{A}$, and a tensor of relatively lower dimension, $\underline{\mathcal{R}}$, using similarity values present in the matrix $\mathbf{C}$ and the data tensor.

In the objective function above, the first term $f(\mathbf{A}, \mathbf{R}_k)$ forces the reconstruction to be similar to the original tensor $\underline{\mathcal{X}}$. The second term, $g(\mathbf{A}, \mathbf{R}_k)$, is a regularization term whose function is to avoid overfitting. The third term, $f_{sim}(\mathbf{C}, \mathbf{R})$, depending on the similarity value denoted by $C_{i,j}$, forces slices of the relational tensor to decrease their differences between one another. This simple extension is supported by the intuition that all slices of the tensor need not contribute equally in the reconstruction of $\underline{\mathcal{X}}$.

A. Slice Similarity Matrix : C

Each element of the $N_r \times N_r$ matrix $\mathbf{C}$ represents the similarity between a pair of tensor slices and is computed using Equation 5 as given below.

$$C(i,j) = \frac{|S(X_i) \cap S(X_j)|}{max(|S(X_i)|, |S(X_j)|)} \tag{5}$$
$$\forall 1 \leq i, j \leq N_r$$

Here $C(i,j)$ represents the similarity score between frontal slice $\boldsymbol{X}_i$ and $\boldsymbol{X}_j$. For binary valued tensors, $S(\boldsymbol{X}_i)$ is the union of the row and column indices with non-zero values in the frontal slice $\boldsymbol{X}_i$ and $|S(\boldsymbol{X}_i)|$ gives the cardinality of the set. The numerator, $|S(\boldsymbol{X}_i) \cap S(\boldsymbol{X}_j)|$, computes the common entities present across a pair of frontal slices. The denominator, $max(|S(X_i)|, |S(X_j)|)$, is used normalize the score between zero and one. The maximum function helps to avoid cases when set of indices of one slice is subset of the indices of another.

B. Computing Factor Matrices : $\mathbf{A}$ and $\mathbf{R}_k$

The factors of the tensor decomposition are computed using the Alternate Least Squares (ALS) method [6], which provides faster updates than other updating algorithms. In the ALS method, an unknown is differentiated while treating other unknowns as constants, hence the matrix $\mathbf{A}$ is updated while the frontal slices of $\underline{\mathcal{R}}$ are treated as constant. Taking the gradient of Equation 1 with respect to $\mathbf{A}$ and setting it equal to zero, we obtain the update rule for $\mathbf{A}$.

$$\mathbf{A} \leftarrow \left[\sum_{k=1}^{N_r} \mathbf{X}_k\mathbf{A}\mathbf{R}_k^T + \mathbf{X}_k^T\mathbf{A}\mathbf{R}_k\right]$$
$$\left[\sum_{k=1}^{N_r} \mathbf{R}_k\mathbf{A}^T\mathbf{A}\mathbf{R}_k^T + \mathbf{R}_k^T\mathbf{A}^T\mathbf{A}\mathbf{R}_k + \lambda_A\mathbf{I}\right]^{-1} \tag{6}$$

Adapting the ASALAN method [6], the unknown matrix $\mathbf{R}_k$ can be found by solving following variant of Equation 1, which is a ridge regression problem shown below.

$$\min_{\mathbf{R}_k} ||vec(\mathbf{X}_k) - (\mathbf{A} \otimes \mathbf{A})vec(\mathbf{R}_k)|| + \lambda_r||vec(\mathbf{R}_k)||$$
$$+\lambda_{sim} \sum_i ||vec(\mathbf{R}_k - \mathbf{R}_i)|| \tag{7}$$

Once solved for $\mathbf{R}_k$, the update rule for $\mathbf{R}_k$ is

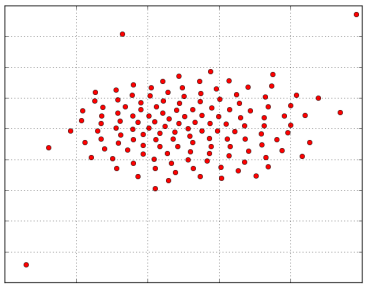
$$\mathbf{R}_k \leftarrow \left((\mathbf{A} \otimes \mathbf{A})^T(\mathbf{A} \otimes \mathbf{A}) + \lambda_r\mathbf{I} + (\lambda_{sim} \sum_i^{N_r} \mathbf{C}(k,i))\mathbf{I}\right)^{-1}$$
$$((\mathbf{A} \otimes \mathbf{A})vec(\mathbf{X}_k)) \tag{8}$$

As the updates are iterative there is no assurance of convergence, but we found empirically that the convergence of our model takes time that is similar to that of RESCAL, typically requiring 40 to 50 iterations.

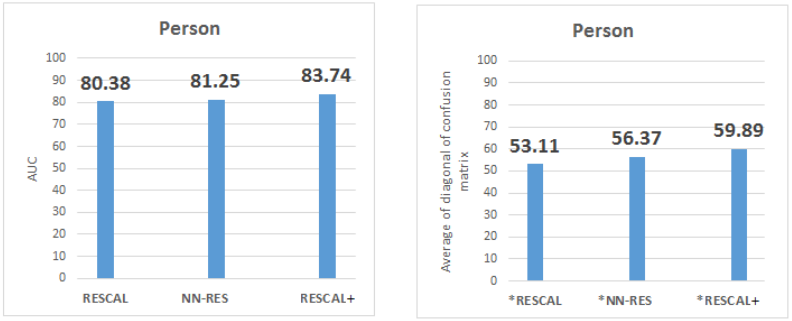
V. Experimental Evaluation

We compared the performance on link-prediction of our RESCAL+ approach with the original RESCAL model [4] and its non-negative variant NN-RES [3] on a real-world dataset, DBpedia-Person [7] using five-fold cross validation. In each fold, we randomly selected ten entries from each tensor slice to generate a test set of positive and negative examples. In order to measure performance, we used two measures: a precision-recall curve and an average of the diagonal entries in the confusion matrix as dataset was imbalanced.

Dataset : We used a real-world dataset, DBpedia-Person, which is a subset of relations for DBpedia entities of type Person. It contains about 10,000 triples and has 140 relations and 4397 entities. We replaced object values with corresponding fixed tag for attribute relations, as such relations can take any arbitrary literal value including strings, dates and numbers. For example, *"Albert Einstein birthDate 1885-10-07 (xsd:date)"* is processed to produce *"Albert Einstein birthDate date"*. If a relation has an object that is an entity, it is left unchanged. After processing, a tensor of size $4397 \times 4397 \times 140$ (approx 2.7B entries) was created. Due to limited space we omit analysis on other



(a) DBpedia-Person



(b) Performance of models on DBpedia-Person

Figure 2. The scatter plot on the left visualizes the slice similarity matrix $\mathbf{C}$ for the DBpedia-Person. Closer data points imply higher similarity between the tensor slices. The graphs show the performance of the models on the DBpedia-Person dataset. Each model is evaluated with two accuracy measures, a precision-recall curve (middle) and average of the diagonal of the confusion matrix (right) to absolutely measure the improvement among the models. Measures using confusion matrix are identified with a prefix *. For example RESCAL+ denotes performance measured using precision-recall curve and *RESCAL+ denotes performance measured using the confusion matrix.

datasets. We used grid search to fix hyperparameters and set λ_r = 0.2, λ_{sim} = 0.2, and λ_A = 10 for our experiments.

Results: We set the dimension, p, of the frontal slices of the unknown core relational tensor $\mathbf{R}_k$ equal to the number of relations present in the dataset. We used Equation 5 to compute the matrix $\mathbf{C}$. We selected a threshold η and rounded down to zero any $c_{ij} \leq \eta||C_{i*}||_1$, where C_{i*} denotes the i^{th} row of $\mathbf{C}$. Relationships that are similar to each other form clusters and help determine the overall similarity of relations across the dataset. In our experiments, we set $\eta = 0.5$ and extended original RESCAL code [1] and used original non-negative variant of RESCAL[2].

Analysis: As shown in Figure 2(b)(left), the AUC results of our RESCAL+ model on the Person dataset is approximately 5% higher than RESCAL and nearly 3% higher than NN-RES. RESCAL+ achieves AUC of 83.74 followed by NN-RES with AUC result of 81.25 and RESCAL with AUC of 80.38. The models performs nearly equally well because they can easily predict values close to zero because of higher number of zero entries in the tensor. Hence, by default, predicting zeros will help achieve relatively similar score using AUC across models. We used the average of the diagonal of the confusion matrix to measure performance in addition to the AUC metric. The confusion matrix helps by showing the impact of false positives and false negatives, which can highlight the improvement achieved by the models to predict ones. As seen in Figure 2(right), performance of RESCAL+ to predict 1 is nearly 10% higher than RESCAL and nearly 5% higher than a NN-RES. The reason for RESCAL+'s better performance can easily be seen in Figure 2(a), as the similarity among the slices is higher for the DBpedia-Person dataset.

[1]https://github.com/mnick/rescal.py

[2]https://gitlab.com/dekromp/non-negative-rescal

VI. Conclusions & Future Work

We described a novel approach using similarities across slices to improve link-prediction in multi-relational data. Our RESCAL+ approach extends the original RESCAL by using a slice similarity matrix $\mathbf{C}$. Preliminary experiments demonstrate that RESCAL+ performs nearly as well on standard data and significantly better on a real-world dataset when compared to RESCAL and its non-negative variant.

VII. Acknowledgment

This work was supported by NSF grant 1228673 and a gift from IBM.

References

[1] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka Jr, and T. M. Mitchell, "Toward an architecture for never-ending language learning." in *Proc. 25th AAAI*, 2010.

[2] T. Franz, A. Schultz, S. Sizov, and S. Staab, "Triplerank: Ranking semantic web data by tensor decomposition," in *Int. Semantic Web Conf.* Springer, 2009.

[3] D. Krompaß, M. Nickel, X. Jiang, and V. Tresp, "Non-negative tensor factorization with RESCAL," in *Tensor Methods for Machine Learning, ECML workshop*, 2013.

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

[5] ——, "Factorizing YAGO: scalable machine learning for linked data," in *21st World Wide Web Conf.* ACM, 2012.

[6] B. W. Bader, R. A. Harshman, and T. G. Kolda, "Temporal analysis of semantic graphs using asalsan," in *7th IEEE Inter. Conf. on Data Mining*. IEEE, 2007.

[7] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann, "Dbpedia-a crystallization point for the web of data," *Web Semantics: science, services and agents on the world wide web*, 2009.