Reflections on: Knowledge Graph Fact Prediction via Knowledge-Enriched Tensor Factorization

Ankur Padia, Konstantinos Kalpakis, Francis Ferraro, and Tim Finin

University of Maryland, Baltimore County, Baltimore, MD, USA
{ankurpadia, kalpakis, ferraro, finin}@umbc.edu

Abstract. We present a family of four novel methods for embedding knowledge graphs into real-valued tensors that capture the ordered relations found in RDF. Unlike many previous models, these can easily use prior background knowledge from users or existing knowledge graphs. We demonstrate our models on the task of predicting new facts on eight different knowledge graphs, achieving a 5% to 50% improvement over existing systems. Through experiments, we derived recommendations for selecting the best model based on knowledge graph characteristics. We also give a provably-convergent, linear tensor factorization algorithm.

1 Introduction

Knowledge graphs are increasingly important due to their effectiveness in supporting a wide range of applications. While immensely useful in their current state, much work remains to be done to detect and correct errors they contain and add missing relations. Representation learning [2] provides a way to augment or reduce reliance on manually constructed ontology axioms and rules by using knowledge graph instances to discover common patterns and then use them to suggest changes to the graph. One popular approach is based on learning how to embed the entities and relations in a graph into a real-valued vector space, allowing the entities and relations to be represented by dense, real-valued vectors. There is considerable interest in understanding how graph embeddings can be used to augment knowledge graphs [11][9].

Current state-of-the-art systems compute embeddings to support a specific task, which might be link ranking (aka link recommendation), or fact prediction (Table 1). Link ranking tries to augment the graph by recommending relations that could hold between a subject–object pair assuming (1) at least one relation exists between the pair and (2) we can determine a good threshold for choosing links that hold. The fact prediction task identifies potential facts and classifies them as correct or incorrect, learning a model that can find relations that are likely to hold. This task is more specific than link ranking and more directly solves an important problem. Since we are only interested in extending a knowledge graph with relations likely to be true, our approach is designed to solve it directly. Fact prediction models can also be used to filter triples produced by information extraction or inference system.

Embedding entities and relations into a vector space has been shown to achieve good results. Tensor–based approaches like RESCAL [14] jointly learn the latent representation of entities and relations by factorizing the tensor representation of the knowledge graph. This can be further improved by imposing constraints on the factors, such
as non-negativity, to achieve better performance on sparse graphs. RESCAL and its variants \cite{21,9,10} have achieved state-of-the-art results in predicting missing relations on real-world knowledge graphs. However, their extensions require additional schema information, which may be unknown or require significant effort to provide. Neural network based approaches, like TransE \cite{4} learn an embedding of a knowledge graph by minimizing the ranking loss to rank likely links higher than unlikely ones.

However, these models do not exploit the similarity among relations when computing embeddings nor have they studied the role that relation similarities have on regularizing and constraining the relation embeddings and the subsequent effect on fact prediction. We addresses these deficiencies and make four contributions: we (1) develop a new graph embedding framework exploiting prior knowledge of relation-similarity; (2) provide four new embedding models; (3) evaluate the models and previous systems on eight real-world knowledge graphs for fact prediction; and (4) prove convergence for a factorization algorithm matching or outperforming baselines. In this extended abstract, we describe one of the model, Quadratic+Constraint (QC) model in detail and refer interested readers to the full paper \cite{18}.

## 2 Similarity-driven knowledge graph embedding

Our approach for similarity-driven knowledge graph embedding relies on learning entity and relation embedding such that when “combined” (multiplied), the result is a close approximation of the original facts and relation occurrences observed in the graph. We augment the learning process with a relational similarity matrix that provides a holistic judgment of how similar pairs of relations are, adding additional constraints to the learned embeddings.

We represent a multi-relational knowledge graph of $N_r$ binary relations among $N_e$ entities by the order-3 tensor $X$ of dimension

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Alternate terminology</th>
<th>Definition</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link ranking</td>
<td>Link recommendation</td>
<td>Given relation $r$ and entity $e_i$, $(e_i, r, ?)$</td>
<td>Input: Where is Statue of Liberty located?</td>
</tr>
<tr>
<td></td>
<td>Link prediction</td>
<td>Rank possible entities $e_j$</td>
<td>Output: (1) Germany (2) United States (3) New York (city) (4) New York (state) (5) Brazil</td>
</tr>
<tr>
<td>Fact prediction</td>
<td>Fact classification</td>
<td>Given entity pair $e_i$ and $e_j$, $(e_i, r, e_j)$</td>
<td>Input: Is the Statue of Liberty located in Germany?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rank possible relations, $r$</td>
<td>Output: 0 (No)</td>
</tr>
</tbody>
</table>

Table 1. We focus on a binary classification fact prediction task rather than link ranking.

Fig. 1. Heatmap for WIN18RR’s similarity matrix using transitivity (relation names abbreviated)
$N_e \times N_e \times N_r$ where each $N_e \times N_e$ is an adjacency matrix for relation $k$ (called slice) has values of 1 when the $i^{th}$ entity ($e_i$) is connected to the $j^{th}$ entity ($e_j$) with relation $k$, else 0. This binary tensor is often very large and sparse. Our goal is to construct dense, informative $p$-dimensional embedding for entities and relations. We represent the collection of $p$-dimensional entity embeddings by $A$ of size $N_e \times p$, a compact order-3 tensor $R$ of size $p \times p \times N_r$, and the similarity matrix by $C$ of size $N_r \times N_r$ (symbols are same as described in [18]).

Our objective is to reconstruct each of the $k$ relation slices of $X$ as the product

$$X_k \approx A_\alpha R_k A_\beta^T.$$  

Here both $A_\alpha$ and $A_\beta$ are matrices: each row is the embedding of an entity. By changing the exact form of $A$—that is, the number of different entity matrices, or the different ways to index $A$—we can then arrive at different models, each encapsulating both mathematical and philosophical differences. In this extended abstract, we describe just one of the four models, with full details on all available at [18]. We examine the case of having only a single entity embedding matrix, represented as $A$—that is, $A_\alpha = A_\beta = A$. This results in a quadratic reconstruction problem, as we approximate $X_k \approx A R_k A^T$. We also examine the case of having two separate entity embedding matrices, represented as $A_1$ and $A_2$. This results in a reconstruction problem that is linear in the entity embeddings, as we approximate $X_k \approx A_1 R_k A_2^T$.

We learn $A$, and $R$ by minimizing the augmented reconstruction loss

$$\min_{A, R} f(A, R) + g(A, R) + f_s(A, R, C).$$  

The first term of (2) reflects each of the $k$ relational criteria given by (1). The second employs standard numerical regularization of the embeddings, such as Frobenius minimization, that enhances the algorithm’s numerical stability and supports the interpretability of the resulting embeddings. The third term uses our relational similarity matrix $C$ to enrich the learning process with our extra knowledge.

**Computing relational similarity matrix $C$.** We can view a knowledge graph’s nodes and edges as representing a flow of information, with subjects and objects acting as information producers and consumers, respectively. Tensor factorization captures this interaction [14]. Relations that occur on more of the same entities are more likely to have some notion of being similar. Each element of the $N_r \times N_r$ matrix $C$ represents the similarity between a pair of relations, i.e., slices $X_i$ and $X_j$, and is computed using the following equation for all pairs of relations ($1 \leq i, j \leq N_r$):

$$C_{i,j} = \frac{|(S(X_i) \cup O(X_i)) \cap (S(X_j) \cup O(X_j))|}{|(S(X_i) \cup O(X_i)) \cup (S(X_j) \cup O(X_j))|}$$  

where $S(X_i)$ is the set of subjects of the matrix $X$ holding the $i^{th}$ relation, and similarly for the object $O(X_i)$. $|S(X)|$ gives the cardinality of the set.
We measure relation similarity using the overlap in the entities observed with each relation. In addition to computing similarity matrices based on relation symmetry using Eq. [3], we support four additional measures: (1) Agency: number of times both the relations have share the same subject, (2) Patient: number of times both the relations have share the same object, (3) Transitivity: number of times object of relation \( i \) is the subject of relation \( j \), and (4) Reverse Transitivity: number of times subject of relation \( i \) is the object of relation \( j \). We experimented with all of the similarity functions and report the evaluation result in Section 3. For most of our experiments we used the similarity obtained from transitivity, as we found it gave the best overall performance. Figure 1 shows the computed similarity matrix for the WordNet dataset.

The QC model: Quadratic+Constraint. The QC model casts the decomposition of the order-3 tensor \( X \) into a compact relational tensor \( R \) and quadratic entity matrix \( A \) and solve the following problem.

\[
\min_{A, R_k} f(A, R_k) + g(A, R_k) + f_{\text{Lag}}(R, C) \quad (4)
\]

The terms of our objective are expressed as follows.

\[
f(A, R_k) = \frac{1}{2} \sum_k |X_k - AR_k A^T|^2_F \quad (5)
\]

\[
g(A, R_k) = \frac{1}{2} \lambda_a ||A||^2_F + \frac{1}{2} \lambda_r \sum_k ||R_k||^2_F \quad (6)
\]

\[
f_{\text{Lag}} = \sum_i \sum_j \lambda_{ij} (1 - ||R_i - R_j||^2_F + C_{ij}). \quad (7)
\]

Here \( A \) is a \( n \times p \) matrix where each row represents the entity embeddings and \( R_k \) is a \( p \times p \) matrix representing the embedding for the \( k^{th} \) relation capturing the interaction between the entities. The first term \( f \) forces the reconstruction to be similar to the original tensor and the second regularizes the unknown \( A \) and \( R_k \) to avoid overfitting.

The \( f_{\text{Lag}} \) term represents the model’s knowledge-directed enrichment component. In order to incorporate similarity constraints, we solve the dual objective for a constraint in Eq. [8] such that relations with high similarity are near one another. We convert a constraint problem into an unconstrained problem via Lagrange multipliers \( \lambda_{ij} \).

\[
||R_i - R_j||^2_F = 1 - C_{ij}, 1 \leq i, j \leq n. \quad (8)
\]

Computing Factor Matrices, \( A \), \( R_k \) and Lagrange Multipliers \( \lambda_{ij} \). We compute the unknown factor matrices using the Adam optimization [8] stochastic gradient descent extension. Each unknown is updated in the alternative fashion, in which each parameter is updated while treating the others as constants. The model’s unknown parameters in \( A \) and \( R_k \) are updated with different learning rates. We empirically found that the error value of the objective function decreases after few iterations. Taking the partial derivative of the Eq. [4] with respect to \( A \) and equating to zero, we obtain the update rule for \( A \). Since we are indirectly constraining the embeddings of \( A \) through slices of the compact relation tensor \( R \), we obtain the same update rule for \( A \) as in RESCAL. By equating the partial derivatives of Eq. [4] with respect to the unknowns \( R_k \) and \( \lambda_{ij} \) to 0, we obtain updates for \( R_k \) and \( \lambda_{ij} \) as describe in the [18].
Table 2. Statistics of the eight datasets used in evaluation experiments

<table>
<thead>
<tr>
<th>Name</th>
<th># Entities</th>
<th># Relations</th>
<th># Facts</th>
<th>Avg. Degree</th>
<th>Graph Density</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kinship</td>
<td>104</td>
<td>26</td>
<td>10,686</td>
<td>102.75</td>
<td>0.98798</td>
</tr>
<tr>
<td>UMLS</td>
<td>135</td>
<td>49</td>
<td>6,752</td>
<td>50.01</td>
<td>0.37048</td>
</tr>
<tr>
<td>FB15-237</td>
<td>14,541</td>
<td>237</td>
<td>310,116</td>
<td>21.32</td>
<td>0.00147</td>
</tr>
<tr>
<td>DB10k</td>
<td>4,397</td>
<td>140</td>
<td>10,000</td>
<td>2.27</td>
<td>0.00052</td>
</tr>
<tr>
<td>FrameNet</td>
<td>22,298</td>
<td>16</td>
<td>62,344</td>
<td>2.79</td>
<td>0.00013</td>
</tr>
<tr>
<td>WN18</td>
<td>40,943</td>
<td>18</td>
<td>151,442</td>
<td>3.70</td>
<td>0.00009</td>
</tr>
<tr>
<td>FB13</td>
<td>81,061</td>
<td>13</td>
<td>360,517</td>
<td>4.45</td>
<td>0.00005</td>
</tr>
<tr>
<td>WN18RR</td>
<td>40,943</td>
<td>11</td>
<td>93,003</td>
<td>2.27</td>
<td>0.00005</td>
</tr>
</tbody>
</table>

3 Experimental evaluation

We evaluated the performance of the entity and relation embeddings on the fact prediction task, which distinguishes correct from incorrect triples, and compared the results with state-of-the-art tensor decomposition models RESCAL and NN-RESCAL, and two popular neural-based benchmarks, DistMult (which considers relation embedding matrices to be diagonal) and ComplEx (which represents entities and relation in complex vector space). Here we present results only for the tensor-based models; additional evaluation data and comparisons with neural-based models are in [18].

Datasets. Table 3 summarizes the key statistics of the datasets: the number of entities ($N_e$), relations ($N_r$) and facts (non-zero entries in the tensor), the average degree of entities across all relations (the ratio of facts to entities) and the graph density (the number of facts divided by square of the number of entities). A smaller average degree or graph density indicates that the knowledge graph is sparser.

We used eight datasets in our evaluation, including both previous graph-embedding benchmarks and new ones. They include (1) Kinship [7], which encodes complex family links among 104 members of a tribe, (2) UMLS, a biomedical dataset [7] based on the Unified Medical Language System, (3,4) WN18 [3] and WN18RR [5] (reverse removed), linguistic datasets of relations between words like hypernym, holonym, meronym and hyponym, (5,6,7) three general world knowledge graphs: FB13 [3], DB10k and FB15-237 which contained more relations compared to FB13, and (8) FrameNet [1], a lexical database describing how language can be used to evoke complex representations of frames describing events, relations or objects and their participants. For example, the Commerce_buy frame represents the interrelated concepts surrounding stereotypical commercial transactions. Frames have roles for expected participants (e.g., Buyer, Goods, Seller), modifiers (e.g., Imposed purpose and Period_of_iterations), and inter-frame relations defining inheritance and usage hierarchies (e.g., Commerce_buy inherits from the more general Getting and is inherited by the more specific Renting.

Tensor creation, parameter selection and performance metric. We created a 0-1 tensor for each dataset, as described in [18]. If entity $s$ has relation $r$ with entity $o$, then the value of the $(s, r, o)$ entry in the tensor is set to 1, otherwise it is set to 0. Each of the created tensors is used to generate a slice-similarity matrix using Eq. 3. We fix the parameters for different datasets using coordinate descent, changing only one hyperparameter at a time and always making a change from the best configuration of hyperpa-
rameters found so far. The latent dimension is equal to the number of relations. We use the same performance metric as RESCAL on three samples: stratified-uniform (sampling 60% correct and 40% incorrect from each relation), stratified-weighted (sampling 60% correct and 40% incorrect from the public dataset with few relations are mentioned more frequently then other), and balanced-weighted (dataset mentioned in [20]).

**Results and discussion.** In this section we analyze and the results of our models, which include a quantitative comparison with other tensor-based models and the impact of knowledge graph sparsity on the models. Table 3 has the AUC performance of our models which range from 5% to 50%.

The Kinship and UMLS datasets have a significantly higher graph density compared to the others, as shown in Table 3. Combining this observation with the results in Table 3, we notice that graphs with lower density result in larger performance variability across both the baseline systems and our models. This suggests that when learning knowledge graph embeddings on dense graphs, basic tensor methods like RESCAL can give acceptable performance, but that for lower density graphs, different embedding learning mechanisms may be better. Focusing on the datasets with lower graph density, we see that while the LC and LR models often matched or surpassed RESCAL, they achieved comparable or lower performance compared to their corresponding quadratic models (QC and QR). This is due to the fact that the distinction of the subject and object made by A₁ and A₂ embeddings tends not to hold in many of the standard datasets. That is, objects can behave as subjects (and vice versa), as in the WN18 dataset. Hence the distinction between the subject and the object may not always be needed.

The performance difference between the quadratic and linear versions is high for WN18 and FB13, though the difference is relatively small for DB10k. This is largely because the DBpedia dataset includes many datatype properties, i.e., properties whose values are literals rather than entities.

In most cases, non-negative RESCAL outperforms the linear models. The QC model significantly outperforms RESCAL and performs relatively better compared to our other three models, emphasizing the importance of the flexible penalization that the Lagrange multipliers provide. Compared to all others, the QC model performs better in most of the cases, since the Lagrange multiplier introduces flexibility in penalizing the latent relational embeddings while learning. We also did significance tests using a

<table>
<thead>
<tr>
<th>Model Name</th>
<th>Kinship</th>
<th>UMLS</th>
<th>WN18</th>
<th>FB13</th>
<th>DB10</th>
<th>Framenet</th>
<th>WN18RR</th>
<th>FB15-237</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESCAL</td>
<td>93.24</td>
<td>88.53</td>
<td>62.13</td>
<td>65.37</td>
<td>61.27</td>
<td>82.54</td>
<td>66.63</td>
<td>92.56</td>
</tr>
<tr>
<td>NN-RESCAL</td>
<td>92.19</td>
<td>88.37</td>
<td>83.93</td>
<td>79.13</td>
<td>81.72</td>
<td>82.6</td>
<td>68.49</td>
<td>93.03</td>
</tr>
<tr>
<td><strong>LR</strong></td>
<td>93.99</td>
<td>88.22</td>
<td>81.86</td>
<td>80.07</td>
<td>80.79</td>
<td>78.11</td>
<td>69.15</td>
<td>90.00</td>
</tr>
<tr>
<td><strong>QR</strong></td>
<td>93.89</td>
<td>88.11</td>
<td>84.41</td>
<td>79.12</td>
<td>80.47</td>
<td>82.34</td>
<td>66.73</td>
<td>93.07</td>
</tr>
<tr>
<td><strong>LC</strong></td>
<td>92.87</td>
<td>84.71</td>
<td>80.18</td>
<td>75.79</td>
<td>80.67</td>
<td>73.64</td>
<td>66.46</td>
<td>81.88</td>
</tr>
<tr>
<td><strong>QC</strong></td>
<td>93.84</td>
<td>86.17</td>
<td>91.07</td>
<td>85.15</td>
<td>81.69</td>
<td>86.24</td>
<td>72.62</td>
<td>86.47</td>
</tr>
</tbody>
</table>

Table 3. Fact prediction AUC performance for all models. ★ indicating best overall. LR and LC are linear regularized/constrained models; QR and QC are quadratic regularized/constrained.
Wilcoxon signed-rank test for all algorithms and datasets at significance level of 1% and found the QC model to perform better compared to the others. In summary, both the quadratic and linear models are important depending on the data, with the QC model performing the best overall and the linear models performing comparably.

**Effect of knowledge graph density.**
We explored how our models handle knowledge graphs of different density by reducing the number of subjects while keeping the objects constant, making the graph sparser. We used the FB13 dataset, which has nearly 16K objects and 76K subjects, so each object entity is connected to nearly five subjects on average. Figure 2 shows the behavior of different tensor-based models on reducing the number of subjects at step size of 2% with higher the value increasingly removing the number of subjects. We see that removing subjects gradually reduces performance. Compared to other models, LC improves significantly faster when subjects are added back, irrespective of the similarity metric, eventually achieving comparable performance with other tensor-based models. The QC model performs the best irrespective of the graph’s density.

![Fig. 2. Change in performance with sparse knowledge graph](image)

**Effect of similarity encoding.** Figure 3 shows the relative changes in performance of each similarity metric compared to RESCAL, grouped by how we encode the knowledge. The gray boxes show the percent change of the two RESCAL versions. Most encoding approaches perform well, but the encoding can yield a significant performance gain for certain datasets. In DB10 (top left) using LR the *agency* and *symmetric* encodings give poor performance. Changing the encoding to *transitivity* or *reverse transitivity* yields a large performance gain. On the other hand, for WN18RR both *transitivity* and *reverse transitivity* with the LR model perform poorly. The LC model performs similarly for all kinds of encoding. Moreover, QC performs consistently well compared to all the baselines without being affected by the similarity encoding. While we find that different kinds of similarity encoding methods can, and do, influence performance, we see the effect of how that knowledge is encoded in these datasets. For example, whether an encoding uses a symmetric or transitive approach may be less important than whether or not accurate knowledge is encoded at all. The knowledge enrichment that the encoding provides can result in effective model generalization beyond what simple, knowledge-poor regularizations, like Frobenius norm regularization, give.
4 Conclusions and future work

We described a new framework for learning knowledge-enriched entity and relation embeddings and four readily obtainable models that generalize existing efforts and demonstrate significant improvements over both state-of-the-art tensor decomposition and neural-based translation models. We motivated and empirically explored different methods for encoding prior knowledge into the tensor factorization algorithm, finding that using transitive relationship chains resulted in the highest overall performance. We further characterized the conditions under which each model performed well and concluded the QC model is typically the best choice. Finally, we proved in [18] that the LR model has the desirable property of convergence.

Our future work will use the KGFP framework for fact prediction in three different scenarios: (1) improving information extraction in tasks like the TAC Knowledge Base Population [6], (2) enhancing the knowledge graphs used in systems [13,12] that identify possible cybersecurity attacks, and (3) cleaning noisy knowledge graphs [16,17] by identifying and possibly correcting errors. The evaluation datasets and some code will be available at the KGFP repository [15].
References