Knowledge Graph Fact Prediction via Knowledge-Enriched Tensor Factorization

Ankur Padia, Konstantinos Kalpakis, Francis Ferraro and Tim Finin

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

Abstract

We present a family of novel methods for embedding knowledge graphs into real-valued tensors. These tensor-based embeddings capture the ordered relations that are typical in the knowledge graphs represented by semantic web languages like RDF. Unlike many previous models, our methods can easily use prior background knowledge provided by users or extracted automatically from existing knowledge graphs. In addition to providing more robust methods for knowledge graph embedding, we provide a provably-convergent, *linear* tensor factorization algorithm. We demonstrate the efficacy of our models for the task of *predicting new facts* across eight different knowledge graphs, achieving between 5% and 50% relative improvement over existing state-of-the-art knowledge graph embedding techniques. Our empirical evaluation shows that all of the tensor decomposition models perform well when the average degree of an entity in a graph is high, with constraint-based models doing better on graphs with a small number of highly similar relations and regularization-based models dominating for graphs with relations of varying degrees of similarity.

Keywords: knowledge graph; knowledge graph embedding; tensor decomposition; tensor factorization; representation learning; fact prediction

1. Introduction

Knowledge graphs are gaining popularity due to their effectiveness in supporting a wide range of applications, ranging from speed-reading medical articles via entity-relationship synopses [1], to training classifiers via distant supervision [2], to representing background knowledge about the world [3, 4], to sharing linguistic resources [5]. Large, broad-coverage knowledge graphs like DBpedia, Freebase, Cyc, and Nell [6] have been constructed from a combination of human input, structured and semi-structured datasets, and information extraction from text, and further refined by a mixture of machine learning and data analysis algorithms. While they are immensely useful in their current state, much work remains to be done to detect the many errors they contain and enhance them by adding relations that are missing. As a simple example, consider instances of the *spouse* relation in the DBpedia knowledge graph. This relation holds between two people and is symmetric, yet the DBpedia version from October 2016 has 3,743 relations where one of the entities is not a type of Person in DBpedia's native ontology and more than half of the inverse relations are missing[1].

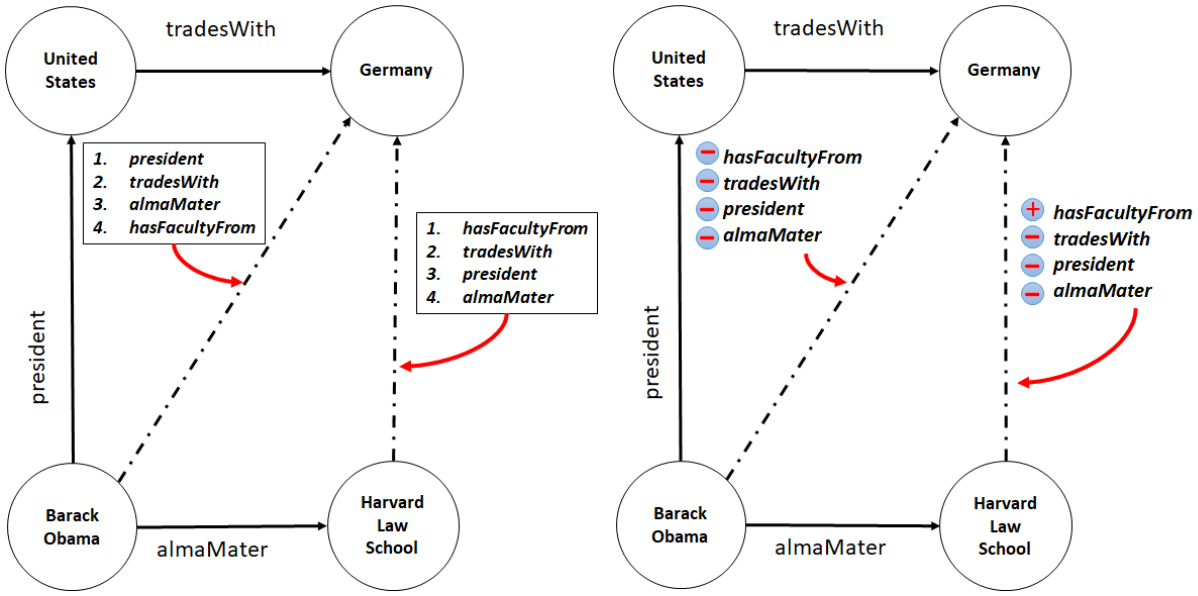
One approach to improving a large knowledge graph like DBpedia is to extend and exploit ontological knowledge, perhaps in the form of logical or probabilistic rules. However, two factors make this approach problematic: the presence of noise in the initial graphs, and the large size of the underlying ontologies. For example, in DBpedia it is infeasible to do simple reasoning with property domain and range constraints because the noisy data produces too many contradictions. The size of DBpedia's schema, with more than 62K properties and 100K types, makes a rule-based approach difficult, if not impossible.

Representation learning [8] provides a way to augment or even replace manually constructed ontology axioms and rules. The general idea is to use instances in a large knowledge graph to discover patterns that are common, and

[1] These observations were made based on data from SPARQL queries run on the public endpoint [7] in December 2017.

Tasks	Alternate terminology	Definition	Example
Link ranking (ranking)	Link prediction Link recommendation	**Input** : Given a relation, r, and an entity e_i. (e_i,r, ?) **Output** : Rank list of possible entity e_j **OR** **Input** : Given a pair of entities, e_i, and e_j. (e_i,?, e_j) **Output** : Rank list of possible relations, *r*	**Input** : Where is Statue of Liberty located? **Output** : (1) Germany (2) United States (3) **New York (city)** (4) New York (state) (5) Brazil
Fact prediction (classification)	Link classification Fact classification	**Input** : A triple (a.k.a fact), e_i, r, and e_j. **Output** : 0 (No) or 1 (Yes)	**Input** : Is the Statue of Liberty located in Germany? **Output** : 0 (No)

Table 1: Distinction among various tasks, their definition, alternate terminology, and an example to understand the phrase 'link prediction' and its usage for a given context. Our approach focuses on the *Fact Prediction* task, which is a binary classification task.



(a) Link Ranking/Link Prediction

(b) Fact Prediction/Fact classification

Figure 1: Link Ranking vs. Fact Prediction. Consider a toy knowledge graph with four entities and four relations. **Link ranking** aims to rank relations for a given pair of entities and is meaningful in the cases where at least one relation holds between a given pair of entities, e.g., (Barack_Obama, ?, United_States) and not (Barack_Obama, ?, Germany). On the other hand, **fact prediction** is the task of deciding which relations are likely to hold between a pair of entities. Link ranking (or recommendation) is a ranking problem, while fact prediction is a binary classification problem.

then use these patterns to suggest changes to the graph. The changes are often in the form of adding missing types and relations, but can also involve changes to the schema, removing incoherent instances, merging sets of instances that describe the same real-world entity, or adding or adjusting probabilities for relations. One popular approach for representation learning systems 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. The entity and relation embeddings can be learned either independently or jointly, and then used to predict additional relations that are missing. Jointly learning the embeddings allows each to enhance the other.

Current state-of-the-art systems of this type compute embeddings to support the task at hand, which might be *link ranking* (or *link recommendation*), or *fact prediction* (Table 1). Link ranking tries to populate the knowledge graph by recommending a list of relations that could hold between a subject–object pair of entities. It assumes that at least one relation exists between the given pair of entities, and is a ranking problem. On the other hand, fact prediction identifies the correct facts from incorrect ones, and is a binary classification problem. To better understand the difference between link ranking and fact prediction, consider the example shown in Figure 1. Here solid lines indicate the observed (correct) relations among the entities; dashed lines indicate the relations we are interested in making recommendation for or identifying their correctness. For the pair (Barack Obama, Germany), **none** of the

recommended relations (in the solid box) can hold. However, due to the design of the problem, a *link prediction* system is required to produce a list of potential relations. On the other hand, for the pair (Harvard Law School,Germany), one relation, *hasFacultyFrom*, can hold while the remaining ones cannot.

In the case of fact prediction, we are interested in making a determination (binary classification) whether or not a relation holds between a given pair of entities. Fact prediction is an important task, as models for it can help identify erroneous facts present in a knowledge graph and also filter facts generated by an inference system or information extraction system. As shown in Figure 1 the circumscribed minus sign ("-") indicates that the relation cannot hold and circumscribed plus sign ("+") that it may hold. Fact prediction can be used as an pre- and post- processing step to link prediction.

Many previous systems have attacked the link ranking task (see Section 2), which involves finding, scoring and ranking links that could hold between a pair of entities. Having such a ranked list is useful and could support, for example, a system that showed the list to a person and asked her to check off the ones that hold. The results of the link ranking task can also be used to predict facts that do hold between a pair of entities, of course. But it introduces the need to learn good thresholds for the scores to separate the possible from the likely. Achieving high accuracy may require that the thresholds differ from one relation to another. Thus we have a new problem that we need to train a system to solve – learning optimal thresholds for the relations. Since we are only interested in extending a knowledge graph with relations that are likely to hold (what we call facts), our approach is designed to solve it directly. Thus we have the fact prediction task: given a knowledge graph, learn a model that can classify relation instances that are very likely to hold. This task is more specific than link ranking and more directly solves an important problem.

Embedding entities and relations into a vector space has been shown to achieve state-of-the-art results. Such embeddings can be generated using tensor factorization or neural network based approaches. Tensor–based approaches like RESCAL [9] jointly learn the latent representation of entities and relations by factorizing the tensor representation of the knowledge graph. Such a tensor factorization could be further improved by imposing constraints, such as non-negativity on the factors, to achieve better sparsity and prediction performance. Moreover, tensor factorization methods, like Tucker and Canonical Polyadic (CP) decompositions [10], have also been applied to knowledge graphs to obtain ranking of facts [11]. RESCAL and its variants [12, 13] have achieved state-of-the-art results in predicting missing relations on real-world knowledge graphs. However, such extensions require additional schema information, which may be absent or require significant effort to provide.

Neural network based approaches, like TransE [14] and DistMult [15], learn an embedding of a knowledge graph by minimizing the ranking loss. As a result, they learn representations in which likely links are ranked higher than unlikely ones. These are evaluated with Mean Reciprocal Rank, which emphasizes the ordering or ranking of the candidate links rather than their correctness. DistMult further assumes that each relation is symmetric. CompleEx [16] relaxes the assumption of symmetric relations by representing the embedding in a vector space of complex, rather than real, numbers. DistMult and ComplEx have both been shown to yield state-of-the-art performance.

However, these models [9, 13, 14, 16, 15] do not explicitly exploit the similarity among the relations when computing entity and relation embeddings, nor have they studied the role that relation similarities have on regularizing and constraining the underlying relation embeddings and the effect on performance in fact prediction task. Other more distantly related methods [17, 18, 19] attempt to learn entity and relation embeddings using association among the relations but as described in the Section 2, the approaches need external text sources to determine the association among the relations/predicates and hence are not standalone like ours. The closest approach which does not depend on an external source (i.e., is *standalone*) is Minervini et al. [20], which uses limited relation similarity cases (i.e., `inverse` and `equivalence`). This can be easily modeled by our approach using weighted regularization and hence the regularization provided by [20] is a special case of our regularization approach.

Our work addresses these deficiencies and make three contributions. First, we develop a framework to learn entity and relation embeddings that incorporates similarity among the relations as prior knowledge. This framework allows us to both generalize existing work [21] and provide **three novel embedding methods**. Our models are based on the intuition that the importance of relations varies in predicting missing relations in a given multi-relational dataset, e.g., knowing that someone is a country's President greatly increases the chances of being a citizen of the country. Formally, each method optimizes an augmented reconstruction loss objective (Section 3.3) Additionally, we use Alternate Least Squares instead of gradient descent to solve the resulting optimization problems.

Second, we evaluate each model, comparing it to state-of-the-art tensor decomposition models (RESCAL and its non-negative variant Non-negative RESCAL) on **eight real-world datasets/knowledge graphs** on fact prediction

task. These datasets exhibit varying degrees of similarity among the relations, allowing us to study our framework's efficacy in varying settings. We provide insight into our models and shed light on the effect of similarity regularization on on the quality of learned embedding for the task and describe how the embedding changed with varying graph sparsity. We show that the quadratic model perform well in general and in most cases, embedding using our quadratic+constraint model perform the best. We also consider our models as *one-best fact prediction* systems, allowing us to compare against TransE, and popular benchmarks DistMult, and ComplEx. Our methods yield **consistent relative improvements of more than 20%** over these baselines, while having the same asymptotic time complexity.

Finally, we make a theoretical contribution by providing a **provably convergent** factorization algorithm that matches, and often outperforms, the baselines. We also empirically investigate its convergence on two standard datasets.

2. Related work

Significant work has been done over the past decades on methods for improving a given knowledge graph by identifying likely errors and either correcting or removing them and by predicting additional facts or relations and adding them to the graph. Paulheim [22] provides an overview of techniques for these tasks, which he calls knowledge graph refinement. Our interest is in the subset of this general problem that involves using embeddings to identify the correct relations between pairs of entities already in a knowledge graph as opposed to link prediction or recommendation,

Knowledge graph embeddings can be created using tensor factorization or neural network based approaches. Both aim to learn a scoring function which assigns a score to a triple, (s, r, o) where s is the subject, r is the relation and o is the object. They learn an embedding using a combination of techniques including the use of regularization, constraints or external information. The choice of the techniques affects both the embedding and the types of applications for which they are suited. We describe a few of them and additional details can be found in [23].

Neural network based approaches. Neural network methods like TransE [24] and Neural Tensor Network (NTN) [25] embed the entities and relations present in multi-relational data using marginal loss. The embeddings are learned in a manner that ranks correct (i.e., positive) triples higher than incorrect (i.e., negative triples). For each triple (s, r, o), TransE tries to bring the object o closer to the sum of subject s and relation r with a linear scoring function $||\mathbf{s} + \mathbf{r} - \mathbf{o}||$. NTN, on the other hand, uses the combination of a bilinear model ($\mathbf{s}^T\mathbf{W}_r\mathbf{o}$) and a linear one ($\mathbf{W}_{rs}\mathbf{s} + \mathbf{W}_{ro}\mathbf{o} + \mathbf{b}_r$) where $\mathbf{W}_{rs}, \mathbf{W}_{ro}$, and $\mathbf{W}_r$ are the relation embeddings. NTN has more parameters than TransE, making it generally more expressive.

TransE's approach was extended by TransH [26], which projects relations in a hyperplane with a translation operation on the hyperplane. Subsequently, DistMult [15] and ComplEx [16] have been shown to learn better embeddings and perform better then TransE, TransH and NTN, achieving what are currently considered to be state-of-the-art results. DistMult is a simpler version of RESCAL where the relation embedding matrix is assumed to be diagonal. However, since its scoring function is symmetric, it considers each relation to be symmetric, and consequently cannot distinguish the difference between the subject and object. This is a serious drawback in domains with asymmetric relations (e.g., $hasParent$, $attacks$, $worksFor$). ComplEx uses the same number of parameters as DistMult and overcomes this drawback by embedding relations in the vector space of complex numbers, so that each relation's embedding vector has a real and an imaginary part. ComplEx uses the both the real and imaginary parts of subject, predicate, and object embeddings to compute the score.

HoLE [27] learns entity and relation embeddings to compute a triple's score with fewer parameters than RESCAL. However, since [28] showed that the holographic embeddings are isomorphic to those of ComplEx, we limit our focus on DistMult and ComplEx. An approach from Guo et al. [29] learns embeddings using ComplEx's objective function and iteratively modifies them using rules learned with AMIE [30]. Such rules can be converted to corresponding score values as entries for the similarity matrix used in our approach (Section 3.2), using a function like Equation 6 in Guo et al. [29]. As the number of atoms in a rule can vary, engineering a function to compute a score for a variable length rule and understanding its effect on fact prediction task requires exploration; we leave this for future work. We compare the quality of our embedding with those of the frequently used baseline approaches DistMult and ComplEx and achieve significant improvement on the fact prediction task.

Tensor factorization based approaches. These approaches compute embeddings by factorizing a knowledge graph's tensor and using the learned factors to assigns a score to each triple. Scores can be boolean, reals, or non-negative reals depending on the factorization constraints. Boolean Tensor Factorization (BTF) [31] decomposes an

input tensor into multiple binary-valued factor tensors. The value of the input tensor is reconstructed using boolean operators on the corresponding individual values of the tensor factors. BTF was extended in [32] by incorporating a Tucker tensor decomposition [10] to predicts links. Each factor contains a boolean value, but since the learned values are boolean, the predicted values are constrained to be either 0 or 1. In contrast, our model assigns a real number to each possible link.

Methods like RESCAL [9] and its schema-based extension [12] decompose a tensor into a shared factor matrix and a shared compact factor tensor [33]. To better model protein interaction networks and social network data, Krompass et al. [13] imposed non-negativity constraints on these factors, but as we show empirically in Section 6, doing so increases the running time of the factorization and introduces scalability issues. Other examples of utilizing schema information include Krompass et al. [12], who use schema information to decompose a tensor using type constraints and updates the factor values following a relation's *rdfs:domain* and *rdfs:range*, and Minervini et al. [34], who incorporate schema information in latent factor models to improve the link prediction task. All of the proposed extensions seem to work well only when the average degree of the entities is high or all of the relations are equally important in predicting the correctness of (possible) facts. Finally, while these approaches offer empirical evidence for the convergence of their iterative algorithms, no convergence guarantees or analysis are available.

Work that can be considered close to ours is Minervini et al. [20], which requires pre-defined equivalence and inverse properties on relations. In contrast, we use a data-driven and self-contained approach an do not rely on or require a schema, pre-trained embeddings or external text corpus. Their approach uses two formulations: one in which the equivalences define hard constraints and another in with soft constraints. While the soft constraints take the same form as the relation regularization we use (i.e., Frobenius between relation embeddings). Our approach is supported by the intuition that not all relations participate equally to identify the fact, which provides more flexibility by weighting different relations. Due to this flexibility [20] can be considered as a special case of our approach to provide regularization described here and in our preliminary work [21]. Additionally, we do not require inclusion of a rich semantic schema. In the absence of a schema (i.e., without using regularization via equivalence or inverse axioms), their approach reduces to the that of TransE, DistMult, and ComplEx, with which we compare our approach in Section 5.

More distantly related work. There are approaches that use external information, either from a text corpus or pre-trained embeddings, to regularized knowledge graph embeddings for downstream applications. These are somewhat related to our approach, which is data driven, *self contained* and does not rely a corpus or pre-trained embeddings. Our regularization approach could be added to other regularization methods, enforcing similarity between predicates in the embeddings space. We leave analysis of addition of our regularization to distantly related work for future study.

We mention here a few references for completeness. As mentioned before, NTN [25] uses pre-trained word embeddings to guide the learning of the knowledge graph embeddings with the intuition that if the words are shared among the entity and relations they share the statistical strength. Beside the use of a text corpus, implication rules are also used to guide the embeddings in some systems [17, 18, 19]. Such implication rules can come from a lexical corpus, such as WordNet or FrameNet, extracted from the knowledge graph itself [35] or can be manually crafted. However, this may require considerable amount of human effort, depending on the availability of the lexical resources.

3. Similarity-driven knowledge graph embedding

In this section we first motivate a general *framework* for incorporating existing relational similarity knowledge. We then describe how our three models pre-compute a *similarity matrix* that measures co-occurrence of pairs of relations and use it to regularize or constrain relation embeddings. Two of the three models optimize linear factorization objectives while the third is a robust extension of the quadratic objective described in Padia et al. [21].

3.1. General framework

Our general framework for similarity-driven knowledge graph embedding relies on minimizing an augmented reconstruction loss. The reconstruction objective learns entity and relation embeddings that, when "combined" (multiplied), closely approximate the original facts and relation occurrences observed in the knowledge graph. We augment the learning process with a *relational similarity* matrix, which provides a holistic judgment of how similar pairs of relations are. These similarity scores allow certain constraints to be placed on the learned embeddings; in this way, we allow existing knowledge to enrich the entity and relation embeddings.

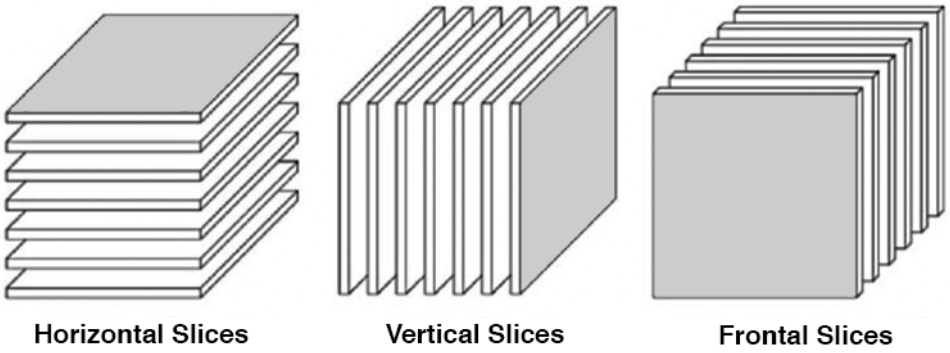


Figure 2: The similarity matrix C is used to compute the similarity of pairs of relations in the knowledge graph. Its i^{th} *frontal slice* is the adjacency matrix of the i^{th} relation, i.e., a two-dimensional matrix with a row and column for each entity whose values are 1 if the relation holds for a pair and 0 otherwise.

In our framework, we represent a multi-relational knowledge graph of N_r binary relations among N_e entities by the order-3 tensor $\mathcal{X}$ of dimension $N_e \times N_e \times N_r$. This binary tensor is often very large and sparse. Our goal is to construct dense, informative p-dimensional embeddings, where p is much smaller than either the number of entities or the number of relations. We represent the collection of p-dimensional entity embeddings by $\mathcal{A}$, the collection of relation embeddings by $\mathcal{R}$, and the similarity matrix by C. The entity embeddings collection $\mathcal{A}$ contains matrices A_α of size $N_e \times p$ while the relation embeddings collection $\mathcal{R}$ contains matrices $\mathbf{R}_k$ of size $p \times p$. Recall that the frontal slice $\boldsymbol{X}_k$ of tensor $\mathcal{X}$ is the adjacency matrix of the k^{th} binary relation, as shown in Figure 2. We use $\mathbf{A} \otimes \mathbf{B}$ to denote the Kronecker product of two matrices $\mathbf{A}$ and $\mathbf{B}$, vec $(\mathbf{B})$ to denote the vectorization of a matrix $\mathbf{B}$, and a lower italic letter like a to denote a scalar.[2]

Mathematically, our objective is to reconstruct each of the k relation slices of $\mathcal{X}$, $\mathbf{X}_k$, as the product

$$\mathbf{X}_k \approx \mathcal{A}_\alpha \mathbf{R}_k \mathcal{A}_\beta^\intercal. \tag{1}$$

Recall that both $\mathcal{A}_\alpha$ and $\mathcal{A}_\beta$ are matrices: each row is the embedding of an entity. By changing the exact form of $\mathcal{A}$—that is, the number of different entity matrices, or the different ways to index $\mathcal{A}$—we can then arrive at different models. These model variants encapsulate both mathematical and philosophical differences. In this paper, we specifically study two cases. First, we examine the case of having only a single entity embedding matrix, represented as $\mathbf{A}$—that is, $\mathcal{A}_\alpha = \mathcal{A}_\beta = \mathbf{A}$. This results in a quadratic reconstruction problem, as we approximate $\mathbf{X}_k \approx \mathbf{A}\mathbf{R}_k\mathbf{A}^\intercal$. Second, we examine the case of having two separate entity embedding matrices, represented as $\mathbf{A}_1$ and $\mathbf{A}_2$. This results in a reconstruction problem that is linear in the entity embeddings, as we approximate $\mathbf{X}_k \approx \mathbf{A}_1\mathbf{R}_k\mathbf{A}_2^\intercal$.

We learn $\mathcal{A}_\alpha, \mathcal{A}_\beta$, and $\mathcal{R}$ by minimizing the augmented reconstruction loss

$$\min_{\mathcal{A},\mathcal{R}} \underbrace{f(\mathcal{A},\mathcal{R})}_{\text{reconstruction loss}} + \overbrace{g(\mathcal{A},\mathcal{R})}^{\text{numerical regularization of the embeddings}} + \underbrace{f_s(\mathcal{A},\mathcal{R},C)}_{\text{knowledge-directed enrichment}}. \tag{2}$$

The first term of (2) reflects each of the k relational criteria given by (1). The second term 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 similarity matrix C to enrich the learning process with our extra knowledge.

We first discuss how we construct the similarity matrix C in Section 3.2 and then, starting in Section 3.3, describe how the framework readily yields three novel embedding models, while also generalizing prior efforts. Throughout,

[2] We use the standard tensor notations and definitions in Kolda and Bader [10]. Recall that the Kronecker product $\mathbf{A} \otimes \mathbf{B}$ of an (m_1, n_1) matrix $\mathbf{A}$ and a (m_2, n_2) matrix $\mathbf{B}$ returns an $(m_1 m_2, n_1 n_2)$ block matrix, where each element of $\mathbf{A}$ scales the entire matrix $\mathbf{B}$.

we show how C can be used as a second type of *regularizer* on the relation embeddings (penalizing large differences in similar relations), or as a *constraint* that forces embeddings of similar relations to be near one another and dissimilar relations to be further apart. In particular, we demonstrate that when using

1. a linear objective with C as a *regularizer* (Sect. 3.3), we obtain a competitive, provably convergent algorithm;
2. a quadratic objective with C as a *constraint* (Sect. 3.4), we obtain a method that relies on the well-known quadratic form, while resulting in significantly higher performance.

3.2. Slice similarity matrix: $\boldsymbol{C}$

Each element of the $N_r \times N_r$ matrix $\boldsymbol{C}$ represents the similarity between a pair of relations, i.e., frontal tensor slices $\boldsymbol{X}_i$ and $\boldsymbol{X}_j$, and is computed using the following equation:

$$(Symmetric) \quad C_{i,j} = \frac{|(S(X_i) \cup O(X_i)) \cap (S(X_j) \cup O(X_j))|}{|(S(X_i) \cup O(X_i)) \cup (S(X_j) \cup O(X_j))|} \forall 1 \leq i,j \leq N_r \tag{3}$$

where $S(\boldsymbol{X}_i)$ is the set of subjects of the matrix $\boldsymbol{X}$ holding the i^{th} relation, and similarly for the object $O(\boldsymbol{X}_i)$. $|S(\boldsymbol{X})|$ gives the cardinality of the set.Intuitively, we measure similarity of two relations using the overlap in the entities observed with each relation. Two relations that operate on more of the same entities are more likely to have *some* notion of being similar. The numerator equals the number of common entity pairs present across the two frontal slices (relations), while the denominator is used to normalize the score between zero and one. Beside Equation 3 we also consider several other similarity function:

$$\textit{(Agency)} \quad C = \frac{|S(X_i) \cap S(X_j)|}{|S(X_i) \cup S(X_j)|} \forall 1 \leq i,j \leq N_r \tag{4}$$

$$\textit{(Patient)} \quad C_{i,j} = \frac{|O(X_i) \cap O(X_j)|}{|O(X_i) \cup O(X_j)|} \forall 1 \leq i,j \leq N_r \tag{5}$$

$$\textit{(Transitivity)} \quad C_{i,j} = \frac{|S(X_i) \cap O(X_j)|}{|S(X_i) \cup O(X_j)|} \forall 1 \leq i,j \leq N_r \tag{6}$$

$$\textit{(Reverse Transitivity)} \quad C_{i,j} = \frac{|O(X_i) \cap S(X_j)|}{|O(X_i) \cup S(X_j)|} \forall 1 \leq i,j \leq N_r \tag{7}$$

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 [9].

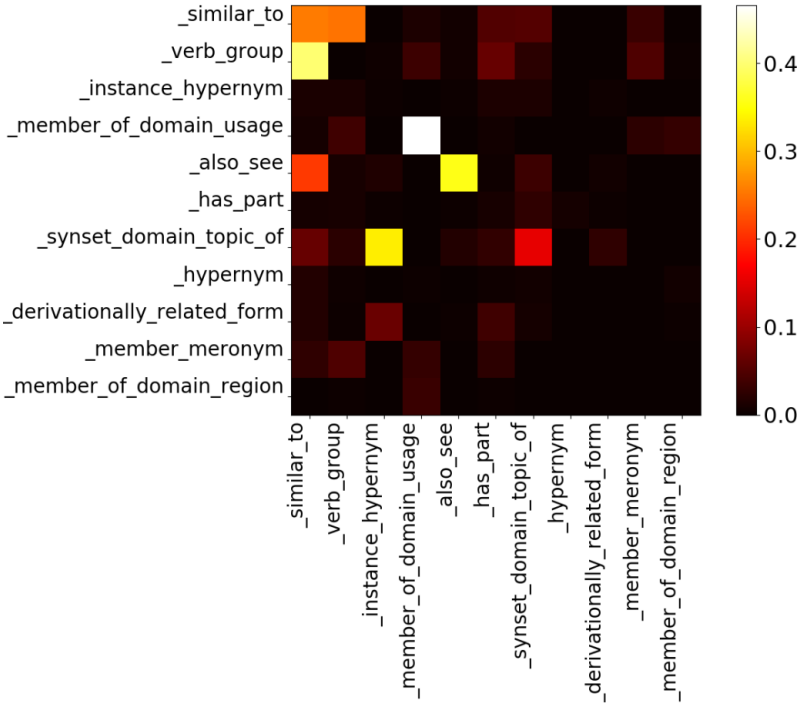
We experimented with all of the similarity functions and report the evaluation result in Section 4. For most of our experiments we used the similarity obtained from transitivity, as we found it gave the best overall performance.

Our similarity function in Eq. 3 is symmetric. An asymmetric similarity function, like the Tversky index [36], could be used, but we found its performance to be comparable to our simpler symmetric similarity function on the link rankingtask. Figure 3 shows the computed similarity matrices for two of our datasets, WordNet and Freebase, with detailed discussion given in Section 4.1).

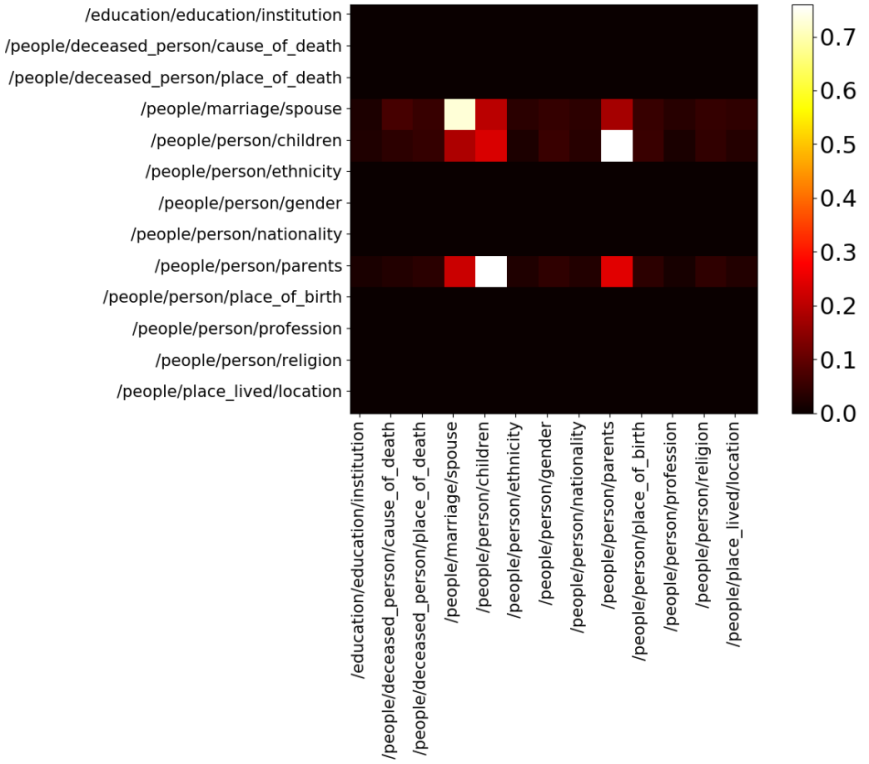
3.3. Model 1: ***Linear + Regularized***

This section presents a linear objective function that can be viewed as a longitudinal extension of previous work that focused on quadratic objectives [21]. We solve the following regularized minimization problem:

$$\min_{\mathbf{A_1}, \mathbf{A_2}, \mathbf{R}_k} \quad f(\mathbf{A_1}, \mathbf{A_2}, \mathbf{R}_k) + g(\mathbf{A_1}, \mathbf{A_2}, \mathbf{R}_k) + f_s(\mathbf{C}, \mathbf{R}_k) + f_\rho(\mathbf{A_1}, \mathbf{A_2}, \mathbf{R}_k) \tag{8}$$



(a) The computed similarity matrix for the WordNet (WN18RR) dataset using the transitivity criterion.



(b) The computed similarity matrix for the Freebase (FB13) dataset using the transitivity criterion.

Figure 3: These heatmaps visualize the similarity among the relations present in the knowledge graph for the WN18RR and FB13 datasets. More darkly colored cells represent lower similarity and brighter ones indicate higher similarity.

where we have decomposed the knowledge-directed enrichment term of (2) into two separate terms, f_s and f_ρ. Specifically, we minimize

$$f\left(\mathbf{A_1}\mathbf{,A_2}\mathbf{,R}_k\right) = \frac{1}{2}\left(\sum_k ||\mathbf{X}_k - \mathbf{A_1}\mathbf{R}_k\mathbf{A_2^T}||_F^2\right) \tag{9}$$

$$g\left(\mathbf{A_1}\mathbf{,A,R}_k\right) = \frac{1}{2}\left(\lambda_A||\mathbf{A_1}||_F^2 + \lambda_A||\mathbf{A_2}||_F^2\right) + \frac{1}{2}\lambda_e||\mathbf{A}_1 - \mathbf{A}_2||_F^2 + \frac{1}{2}\left(\lambda_r\sum_k ||\mathbf{R}_k||_F^2\right) \tag{10}$$

$$f_s\left(\mathbf{C}, \mathbf{R}_k\right) = \frac{1}{2}\lambda_s\sum_i \mathbf{C}_{k,i}\cdot||\mathbf{R}_k - \mathbf{R}_i||_F^2 \qquad \forall 1 \le i \le N_r, 1 \le k \le N_r \tag{11}$$

$$f_\rho\left(\mathbf{A}_1, \mathbf{A}_2, \mathbf{R}_k\right) = \frac{1}{\rho}\left(||\mathbf{A}_1||_F^2 + ||\mathbf{A}_2||_F^2 + ||\mathbf{R}_k||_F^2\right) \tag{12}$$

Each row i of the matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ is a latent representation of the corresponding ith entity. The frontal slice $\mathbf{R}_k$ is a $p \times p$ matrix representing the interaction of all entities with respect to the kth relationship. The precomputed matrix $\mathbf{C}$ is an $N_r \times N_r$ similarity matrix where each element is a similarity score between two tensor slices (relations). The model's objective is to factorize a given data tensor $\mathcal{X}$ into shared matrices $\mathbf{A}_1$ and $\mathbf{A}_2$, and a tensor of relatively low dimension, $\mathcal{R}$, while considering the similarity values present in the matrix $\mathbf{C}$.

In the objective function above, the first term $f\left(\mathbf{A}_1, \mathbf{A}_2, \mathbf{R}_k\right)$ forces the reconstruction to be similar to the original tensor $\mathcal{X}$. The second term, $g\left(\mathbf{A}_1, \mathbf{A}_2, \mathbf{R}_k\right)$, is a regularization term to avoid overfitting and nudge $\mathbf{A}_1$ and $\mathbf{A}_2$ to be equal. The Frobenius norm $||\cdot||_F$ promotes solutions with a small total magnitude, in the sense of Euclidean length.

The third term, $f_s\left(\mathbf{C}, \mathbf{R}_k\right)$, provides the longitudinal extension to tensor decomposition. It supports the differential contribution of tensor slices in the reconstruction of the tensor $\mathcal{X}$. The similarity values ($C_{i,j}$) force slices of the relational tensor to decrease their differences between one another. To reduce the degree of entity embeddings from quadratic to linear, we use the split-variable technique by replacing variable $\mathbf{A}$ with two variables, $\mathbf{A}_1$ and $\mathbf{A}_2$, that are constrained to be equal. To guarantee convergence, we add an additional term, f_ρ (Equation 12), to Equation 8 which has partial Hessians that are positive definite. The use of ρ is motivated at high-level from proximal algorithms [37] to ensure the strict convexity of the objective function, as described in Appendix A).

3.3.1. Computing factor matrices $\mathbf{A_1}$, $\mathbf{A_2}$ and $\mathbf{R}_k$

We compute the factor matrices with alternating least squares (ALS) [38], a non-linear block Gauss-Seidel method in which the blocks are the unknowns $\mathbf{A_1}$, $\mathbf{A_2}$ and the frontal slices of $\boldsymbol{\mathcal{R}}$. We consider the partial objective functions that need to be optimized, one for each block when the other objective function blocks are kept fixed. We find that each function is a quadratic form of the unknown block whose Hessian is always positive semi-definite, and it is positive definite whenever $\rho > 0$. In other words, each partial objective function is a strictly convex quadratic (least squares) problem with a unique global minimum. In particular, taking the gradient of Eq. 8 with respect to $\mathbf{A_1}$ and setting it equal to zero, we obtain the update rule for $\mathbf{A_1}$.

$$\mathbf{A_1} \leftarrow \left[\sum_{k=1}^{N_r}\mathbf{X}_k\mathbf{A_2}\mathbf{R}_k^T + \lambda_A\mathbf{A_2}\right]\left[\sum_{k=1}^{N_r}\mathbf{R}_k{\mathbf{A_2}}^T\mathbf{A_2}\mathbf{R}_k^T + \left(\lambda_A + \lambda_e + \frac{1}{\rho}\right)\mathbf{I}\right]^{-1} \tag{13}$$

Similarly, taking the gradient of Eq. 8 with respect to $\mathbf{A_2}$ and setting it equal to zero, we obtain the update rule for $\mathbf{A_2}$.

$$\mathbf{A_2} \leftarrow \left[\sum_{k=1}^{N_r}\mathbf{X}_k^T\mathbf{A_1}\mathbf{R}_k + \lambda_A\mathbf{A_1}\right]\left[\sum_{k=1}^{N_r}\mathbf{R}_k^T{\mathbf{A_1}}^T\mathbf{A_1}\mathbf{R}_k + \left(\lambda_A + \lambda_e + \frac{1}{\rho}\right)\mathbf{I}\right]^{-1} \tag{14}$$

The unknown matrix $\mathbf{R}_k$ can be found by solving following variant of Eq. 8, which is a ridge regression problem with positive definite Hessian.

$$\min_{vec(\mathbf{R}_k)} ||vec(\mathbf{X}_k) - (\mathbf{A_2}\otimes\mathbf{A_1})vec(\mathbf{R}_k)||^2 + \left(\lambda_r + \frac{1}{\rho}\right)||vec(\mathbf{R}_k)||^2 + \lambda_s\sum_i ||vec(\mathbf{R}_k - \mathbf{R}_i)||^2$$

Since the problem is strictly convex, the unique minimum is obtained by setting the gradient to 0, leading to the following update rule for $\mathbf{R}_k$.

$$\mathbf{R}_k \leftarrow \left((\mathbf{A_2} \otimes \mathbf{A_1})^T (\mathbf{A_2} \otimes \mathbf{A_1}) + \left(\lambda_r + \frac{1}{\rho}\right)\mathbf{I} + (\lambda_s \sum_i^{N_r} \mathbf{C}(k,i))\mathbf{I} \right)^{-1} (\mathbf{A_2} \otimes \mathbf{A_1}) vec(\mathbf{X}_k)) \quad (15)$$

3.4. Model 2: ***Quadratic + Constraint***

In the second model, we consider the decomposition of $\mathcal{X}$ into a compact relational tensor $\mathcal{R}$ and quadratic entity matrix $\mathbf{A}$. We solve the following problem

$$\min_{\mathbf{A},\mathbf{R}_k} f(\mathbf{A}, \mathbf{R}_k) + g(\mathbf{A}, \mathbf{R}_k) \quad (16)$$

under the constraint that relations with high similarity are near one another.

$$||\mathbf{R}_i - \mathbf{R}_j||_F^2 = 1 - C_{ij}, 1 \leq i, j \leq n. \quad (17)$$

The two terms of our objective are expressed as follows.

$$f(\mathbf{A}, \mathbf{R}_k) = \tfrac{1}{2}\sum_k ||\mathbf{X}_k - \mathbf{A}\mathbf{R}_k\mathbf{A}^T||_F^2 \quad (18)$$

$$g(\mathbf{A}, \mathbf{R}_k) = \tfrac{1}{2}\lambda_a||\mathbf{A}||_F^2 + \tfrac{1}{2}\lambda_r \textstyle\sum_k ||\mathbf{R}_k||_F^2 \quad (19)$$

Here $\mathbf{A}$ is a $n \times p$ matrix where each row represents the entity embeddings and $\mathbf{R}_k$ 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 $\mathbf{A}$ and $\mathbf{R}_k$ to avoid overfitting. In order to incorporate similarity constraints, we modify Eq. 16 to solve the dual objective, via Lagrange multipliers λ_{ij} as below.

$$\min_{\mathbf{A},\mathbf{R}_k} f(\mathbf{A}, \mathbf{R}_k) + g(\mathbf{A}, \mathbf{R}_k) + f_{\text{Lag}}(\mathcal{R}, \mathbf{C}) \quad (20)$$

$$f_{\text{Lag}} = \sum_i \sum_j \lambda_{ij}(1 - ||\mathbf{R}_i - \mathbf{R}_j||_F^2 + \mathbf{C}_{ij}). \quad (21)$$

The f_{lag} term represents the model's knowledge-directed enrichment component.

3.4.1. Computing Factor Matrices, $\boldsymbol{A}$*,* $\boldsymbol{R}_k$ *and Lagrange Multipliers* λ_{ij}

We compute the unknown factor matrices using Adam optimization [39], an extension to stochastic gradient descent. Each unknown is updated in the alternative fashion, in which each parameter is updated while treating the others as constants. Each unknown parameter of the model, $\mathbf{A}$ and $\mathbf{R}_k$, is updated with different learning rate. We empirically found that the error value of the objective function decreases after few iterations. Taking the partial derivative of the Eq. 20 with respect to $\mathbf{A}$ and equating to zero we obtain the following update rule for $\mathbf{A}$.

$$\mathbf{A} \leftarrow \left(\mathbf{X}_k^T\mathbf{A}\mathbf{R}_K + \mathbf{X}_k\mathbf{A}\mathbf{R}_k^T\right)\left(\mathbf{R}_k^T\mathbf{A}^T\mathbf{A}\mathbf{R}_k^T + \lambda_a\mathbf{I}\right)^{-1} \quad (22)$$

Since we are indirectly constraining the embeddings of $\mathbf{A}$ through slices of the compact relation tensor $\mathcal{R}$, we obtain the same update rule for $\mathbf{A}$ as in RESCAL [9]. By equating the partial derivatives of Eq. 20 with respect to the unknowns $\mathbf{R}_k$ and λ_{ij} to 0, and solving for those unknowns, we obtain the following updates:

$$vec(\mathbf{R}_K) \leftarrow \left(\left(\mathbf{A}^T\mathbf{A} \otimes \mathbf{A}^T\mathbf{A}\right) + \lambda_r\mathbf{I} + \lambda_{i=k,j}\mathbf{I}\right)^{-1}\left((\mathbf{A} \otimes \mathbf{A})^T vec(\mathbf{X}_k) + \sum_j \lambda_{i=k,j} vec(\mathbf{R}_j)\right) \quad (23)$$

$$\lambda_{ij} \leftarrow ||\mathbf{R}_i - \mathbf{R}_j||_F^2 + \mathbf{C}_{ij} - 1. \quad (24)$$

3.5. Model 3: ***Linear + Constraint***

This version combines the previous two models: we examine the linear reconstruction loss of Sect. 3.3 with the constraints of Sect. 3.4.

As before, we split the entity embedding of $\mathbf{A}$ into $\mathbf{A}_1$ and $\mathbf{A}_2$. Additionally, we apply the same constraint as in Eq. 17 and solve following constrained problem:

$$\min_{\mathbf{A_1},\mathbf{A_2},\mathbf{R}_k} \quad f\left(\mathbf{A_1},\mathbf{A_2},\mathbf{R}_k\right) + g\left(\mathbf{A_1},\mathbf{A_2},\mathbf{R}_k\right) \tag{25}$$

such that,

$$||\mathbf{R}_i - \mathbf{R}_j||_F^2 = 1 - C_{ij}, 1 \leq i,j \leq n \tag{26}$$

where,

$$f\left(\mathbf{A_1},\mathbf{A_2},\mathbf{R}_k\right) = \frac{1}{2}\left(\sum_k ||\mathbf{X}_k - \mathbf{A_1}\mathbf{R}_k\mathbf{A_2^T}||_F^2\right) \tag{27}$$

$$g\left(\mathbf{A_1},\mathbf{A},\mathbf{R}_k\right) = \frac{1}{2}\left(\lambda_A||\mathbf{A_1}||_F^2 + \lambda_A||\mathbf{A_2}||_F^2\right) + \frac{1}{2}\lambda_e||\mathbf{A}_1 - \mathbf{A}_2||_F^2 + \frac{1}{2}\left(\lambda_r \sum_k ||\mathbf{R}_k||_F^2\right) \tag{28}$$

We rewrite the above constrained problem into a unconstrained one using λ_{ij} as a Lagrange multiplier as follows.

$$\min_{\mathbf{A_1},\mathbf{A_2},\mathbf{R}_k} \quad f\left(\mathbf{A_1},\mathbf{A_2},\mathbf{R}_k\right) + g\left(\mathbf{A_1},\mathbf{A_2},\mathbf{R}_k\right) + f_{\text{Lag}}\left(\mathbf{R}_k,\mathbf{C}\right) \tag{29}$$

where $f\left(\mathbf{A_1},\mathbf{A_2},\mathbf{R}_k\right)$, and $g\left(\mathbf{A_1},\mathbf{A_2},\mathbf{R}_k\right)$ are same as Eq. 27. f_{Lag} is the same as Eq. 21.

3.5.1. Computing the unknowns, $\boldsymbol{A}$, $\boldsymbol{R}_k$ and λ_{ij}

As in the previous model, we use an Adam optimizer. Taking the derivative of Eq. 25 with respect to $\mathbf{A}_1$ and $\mathbf{A}_2$, respectively and equating to zero, we obtain the following update rule.

$$\mathbf{A}_1 \leftarrow \left(\lambda_e\mathbf{A}_2 + \mathbf{X}_k\mathbf{A}_2\mathbf{R}_k^T\right)\left(\mathbf{R}_k\mathbf{A}_2^T\mathbf{A}_2\mathbf{R}_k + \lambda_{a_1}I + \lambda_e I\right)^{-1} \tag{30}$$

$$\mathbf{A}_2 \leftarrow \left(\lambda_e\mathbf{A}_1 + \mathbf{X}_k^T\mathbf{A}_1\mathbf{R}_k\right)\left(\mathbf{R}_k^T\mathbf{A}_1^T\mathbf{A}_1\mathbf{R}_k + \lambda_{a_2}I + \lambda_e I\right)^{-1} \tag{31}$$

Similarly, taking derivate with respect to k^{th} slice of relation, $\mathbf{R}_k$ yields the following update rule.

$$vec(\mathbf{R}_k) \leftarrow \left(\left(\mathbf{A}_2 \otimes \mathbf{A}_1\right)^T vec(\mathbf{X}_K) - \textstyle\sum_{i=k,j}\lambda_{kj}vec(\mathbf{R}_j)\right)\left(\left(\mathbf{A}_2^T\mathbf{A}_2 \otimes \mathbf{A}_1^T\mathbf{A}_1\right) + \lambda_r\mathbf{I} - \textstyle\sum_j \lambda_{kj}\right)^{-1} \tag{32}$$

4. Experimental evaluation

We evaluated the performance of the learned entity and relation embeddings on the *fact prediction* task, which identifies correct triples from incorrect ones, and compared the results against state-of-the-art tensor decomposition techniques and translation methods, like TransE. We also demonstrate the convergence of our linear model on two standard benchmark datasets.

We carried out evaluations on eight real-world datasets. Five have been extensively used previously as benchmark relational datasets: Kinship, UMLS, WordNet (WN18), WordNet Reverse Removed (WN18RR) and Freebase (FB13). We created a sixth dataset, DBpedia-Person (DB10k), to explore how well our approach works on datasets with a larger number of relations. We created a seventh dataset from FrameNet, an ontological and lexical resource [40]. Finally, we used the FB15-237 dataset which was based on Freebase to explore how systems work with a relatively larger number of relations.

We compare our models with state-of-the-art tensor decomposition models, RESCAL and its non-negative variant NN-RESCAL, along with two popular benchmarks, DistMult, which consider the relation embedding matrix to be diagonal, and ComplEx, which represent entities and relation in complex vector space.[3]

[3]We also experimented with other tensor decomposition models [41] like PARAFAC [42, 43] and TUCKER, but the unfolding of the tensors for the larger datasets (WN18 and FB13) required more than 32GB RAM of memory, which we were unable to support on our testbed.

Table 2: Statistics of the eight datasets used in the evaluation experiments. The number of facts represents the number of triples.

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

4.1. Datasets

Table 2 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). Note that a smaller average degree or graph density indicates that the knowledge graph is sparser.

Kinship [44] is dataset with information about complex relational structure among 104 members of a tribe. It has 10,686 facts with 26 relations and 104 entities. From this, we created a tensor of size $104 \times 104 \times 26$.

UMLS [44] has data on biomedical relationships between categorized concepts of the Unified Medical Language System. It has 6,752 facts with 49 relations and 135 entities. We created a tensor of size $135 \times 135 \times 49$.

WN18 [24] contains information from WordNet [5], where entities are words that belong to *synsets*, which represent sets of synonymous words. Relations like *hypernym*, *holonym*, *meronym* and *hyponym* hold between the synsets. WN18 has 40,943 entities, 18 different relationships and more than 151,000 facts. We created a tensor of size 40,943 $\times$ 40,943 $\times$ 18.

WN18RR [45] is a dataset derived from WN18 that corrects some problems inherent in WN18 due to the large number of symmetric relations. These symmetric relations make it harder to create good training and testing datasets, a fact noticed by [46] and [47]. For example, a training set might contain (e_1, r_1, e_2) and test might contain its inverse (e_2, r_1, e_1), or a fact occurring with $e_1 and e_2$ with some relation r_2.

FB13 [24] is a subset of a facts from Freebase [4] that contains general information like "*Johnny Depp won MTV Generation Award*". FB13 has 81,061 entities, 13 relationship and 360,517 facts. We created a tensor of size 81,061 $\times$ 81,061 $\times$ 13.

FrameNet [48] is 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 textttPeriod_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`.

We processed FrameNet 1.7 to produce triples representing these frame-to-frame, frame-to-role, and frame-to-word relationships. FrameNet 1.7 defines roughly 1,000 frames, 10,000 lexical triggers, and 11,000 (frame-specific) roles. In total, we used 16 relations to describe the relationship among these items.

DB10k is a real-world dataset with about 10,000 facts involving 4,397 entities of type Person (e.g., Barack Obama) and 140 relations. We used a DBpedia public SPARQL endpoint [7] to collect the facts which were processed in the following manner. When the object value was a date or number, we replaced the object value with fixed tag. For example, *"Barack Obama marriedOn 1992-10-03 (xsd:date)"* is processed to produce *"Barack Obama*

Hyperparameter	Meaning	Possible Values
λ_A	Coefficient of the entity embedding regularizers	$\{0.0001, 0.01, 0.1, 0, 1, 10, 100, 1000\}$
λ_r	Coefficient of the relation embedding regularizers	$\{0.002, 0.2, 0.01, 0.1, 0, 1, 10, 100, 1000\}$
λ_E	Coefficient of the entity embedding dissimilarity penalty	$\{1, 2, 5, 10\}$
λ_{sim}	Coefficient of the relation similarity $\mathbf{C}$ regularizer	$\{0.00002, 0.02, 0.2, 0.1, 0, 1\}$

Table 3: The possible values our hyperparameters could take.

marriedOn date". In case object is an entity it is left unchanged. For example "Barack Obama is-a President" as President is an entity. Such an assumption can strengthen the overall learning process as entities with similar attribute relations will tend to have similar value in the tensor. After processing, a tensor of size 4,397 $\times$ 4,397 $\times$ 140 was created.

FB15-237 is a dataset containing subset of the Freebase with 237 relations and nearly 15K entities. It has triples coupled textual mention obtained from ClubWeb12. More details about the dataset can be found in [49, 46].

4.2. Tensor creation and parameter selection

We created a 0-1 tensor for each dataset as shown in Figure 2. If entity *s* had relation *r* with entity *o*, then the value of (*s, r, o*) entry in the tensor is set to 1, otherwise it is set to 0. Each of the created tensors was used to generate a slice-similarity matrix using Eq. 3.

We fixed the parameters for different datasets using co-ordinate descent, changing only one hyperparameter at a time and always making a change from the best configuration of hyperparameters found so far. The number of latent variables for the compact relational tensor $\mathcal{R}$ was set to number of relations present in the dataset. In order to capture similarity, we computed the similarity matrix $\mathbf{C}$ using various similarity metric discussed in Section 3.2 and present results produced by *Transitivity*, as it gave better performance overall. See Table 3 for the values our hyperparameters could take.

4.3. Evaluation protocol and metrics

We considered *fact prediction* as a classification task with labels *correct* (i.e., value 1) for the positive class, and *incorrect* (i.e., value 0) for the negative class for a given pair of entities and a relationship. We follow the same evaluation metric used in RESCAL [9], masking the test instances during training and using area under the curve as one of the evaluation metrics.

We conducted evaluations in three different categories. The first used a *stratified-uniform* sampling for which we created a stratified sampling links with 60% *correct* and 40% *incorrect*. To create the test dataset we selected ten instances from each slice for the smaller and fewer entity datasets (Kinship, UMLS, DB10k, FrameNet, and FB15-237) and 200 instances from each slice for the larger ones (WN18, FB13, and WN18RR). We masked the test instances during training. We refer to this category as uniform since all of the relation participate equally in the generated test dataset. The results from this dataset are available in Table 4

The second category used a *stratified-weighted* sampling with 60% *correct* and 40% *incorrect* links, but instead of generating five test sets we used the test dataset that was publicly available and tested it on FB13 and WN18RR. The original dataset contained 5000 positive examples. We randomly sampled 60% of these for positive instances and used the remaining 40% to generate negative instances by replacing their objects with randomly chosen new ones. We followed a similar procedure for FB15-237. We evaluate on the datasets in Table 5.

The third evaluation dataset category is *balanced-weighted*. This is the dataset made publicly available by Socher et al. [25] in his Neural Tensor Network approach. For simplicity we name the dataset as FB13NTN and WN11NTN. Details of the results are explained in Section 5.2.

4.4. Results and discussion of tensor based decomposition models

In this section we provide a detailed analysis 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 tensor based models. We

compare our models with neural-based ones in Section 5 and provide insight on how each model performs with respect to different relations.

4.4.1. Comparison with other tensor based models

Table 4 shows the performance of all our models using three different metrics. We first focus our discussion on area under the curve (AUC), where we see our models obtain relative performance gains ranging from 5% to 50%. We note that AUC was the evaluation metric used by Nickel et al. [9], and we use it as one of our evaluation metrics for consistency. We include an in-depth examination of the different similarity encodings in Figure 5, and then examine the standard information extraction F1 metric in more detail.

The Kinship and UMLS datasets have a significantly higher graph density compared to our other five datasets, as shown in Table 2. Combining this observation with the results in Table 4, 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 with non-knowledge-graph specific regularization or constraints, such as RESCAL, could be used to give acceptable performance. On the other hand, this also suggests that for lower density graphs, different mechanisms for learning embeddings perform differently.

Focusing on the datasets with lower density graphs, we see that while the Linear+Constraint and Linear+Regularized models often matched or surpassed RESCAL, they achieved comparable or lower performance compared to their corresponding quadratic models. This is due to the fact that the distinction of the subject and object made by $\mathbf{A}_1$ and $\mathbf{A}_2$ embeddings tends not to hold in many of the standard datasets. That is, objects can behave as subjects (and vice versa), as is the case in WN18. 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 strings rather than entities. In most cases the non-negative RESCAL variant outperforms the linear models.

The Quad+Constraint model significantly outperforms RESCAL and performs relatively better compared to our other three models. This emphasizes the importance of the flexible penalization that the Lagrange multipliers provides. Compared to RESCAL, regularization using similarity provides additional gain through the better quality of entity and relational embeddings. However, when compared to non-negative RESCAL, the regularized model performs relatively similar. We believe that for fact prediction, regularizing the embeddings results a similar effect as introducing high sparsity in the embeddings through non-negativity constraint. Compared to all others, the Quad+Constraint model performs better in most of the cases, since the Lagrange multiplier introduces flexibility in penalizing the latent relational embeddings while learning. We also conducted statistical significance using Wilcoxon rank sum paired test across all the algorithms and all datasets at significance level of 1% (0.01) and found the Quad+Constraint model to perform better compared to the other algorithms.

We observe similar trends with other standard classification metrics, such as micro- or macro-averaged F1. These can be seen in Tables 4b and 4c, respectively. We see that, as with AUC, the Quad+Constraint model performs well overall. Meanwhile, the Linear+Reg model performs well on Kinship and comparably to the top performing system on UMLS; this reflects the prior observed connection between higher graph density and overall competitiveness of all models involved. While there can be large variability both within and across micro- and macro-F1 in the knowledge-endowed, tensor factorization models, the Quad+Constraint model yields a high performing classifier that may not be as sensitive to less-frequently occurring relations as other factorization methods. This further highlights the the knowledge encoding's positive impact.

In summary, both the Quadratic and Linear models are important depending on the data, with the Quad+Constrain model performing the best overall and the Linear models performing comparably, depending on the data.

4.4.2. Behavior of tensor-based models and knowledge graph density

In order to understand the behavior of different tensor based models to handle knowledge graph of different density we conducted experiments in which we reduced the number of subjects present in the graph and kept the objects constant. Reducing the number of subject with constant number of objects simulates the effect of the graph getting denser. For our experiment we used FB13 which has nearly 16K objects and 76K subjects, indicating that on an average each object entity connected to nearly five subjects.

(a) Fact prediction performance using Area Under the Curve (AUC) as the metric

Area Under the Curve								
Model Name	**Kinship**	**UMLS**	**WN18**	**FB13**	**DB10**	**Framenet**	**WN18RR**	**FB15-237**
Previous tensor factorization models								
RESCAL	93.24	**88.53**	62.13	65.37	61.27	82.54	66.63	92.56
Non Neg RESCAL	92.19	88.37	83.93	79.13	**81.72**	82.6	68.49	93.03
Regularized/Constrained tensor factorization models								
Linear + Reg	**93.99**	88.22	81.86	80.07	80.79	78.11	69.15	90.00
Quad + Reg	93.89	88.11	84.41	79.12	80.47	82.34	66.73	**93.07**
Linear + Constraint	92.87	84.71	80.18	75.79	80.67	73.64	66.46	81.88
★ Quad + Constraint	93.84	86.17	**91.07**	**85.15**	81.69	**86.24**	**72.62**	86.47

(b) Fact prediction performance using F1 Micro as the metric

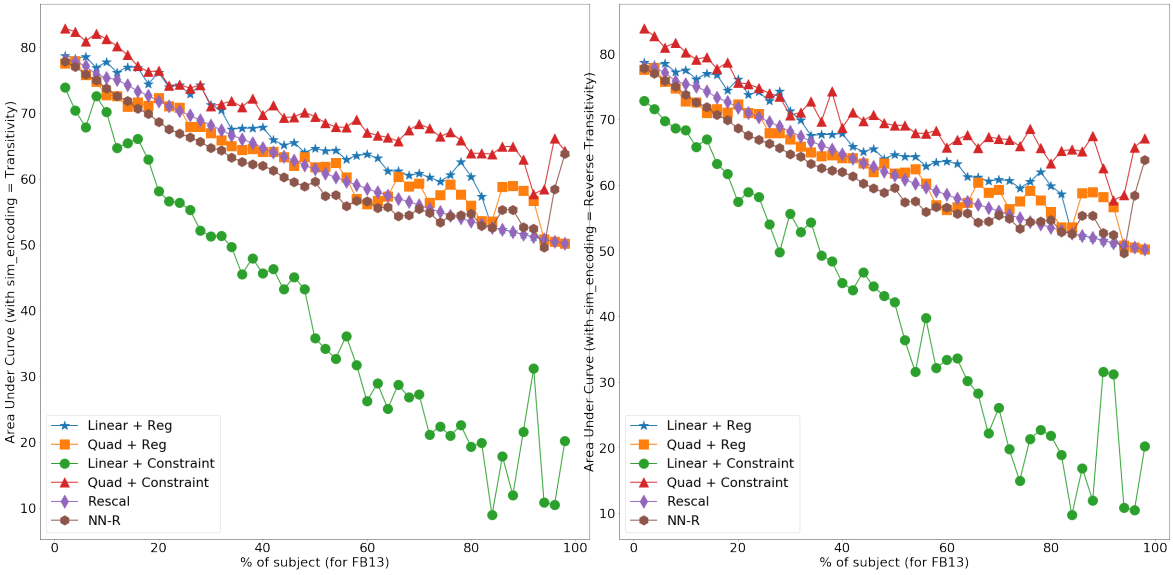
F1 Micro								
Model Name	**Kinship**	**UMLS**	**WN18**	**FB13**	**DB10**	**Framenet**	**WN18RR**	**FB15-237**
Previous tensor factorization models								
RESCAL	81.31	67.71	40.01	47.04	40.00	60.75	56.57	78.84
Non Negative RESCAL	77.23	**69.43**	63.69	58.28	**47.73**	60.75	52.35	79.45
Regularized/Constrained tensor factorization models								
Linear + Reg	**81.54**	68.04	60.31	57.96	47.39	54.75	47.58	70.80
Quad + Reg	81.38	67.35	64.09	57.22	47.39	60.62	44.92	**79.70**
Linear + Constraint	78.46	58.73	57.04	49.15	46.13	47	46.05	13.70
★ Quad + Constraint	81.23	62.00	**79.62**	**67.88**	44.12	**66.5**	**68.01**	59.59

(c) Fact prediction performance using F1 Macro as the metric

F1 Macro								
Model Name	**Kinship**	**UMLS**	**WN18**	**FB13**	**DB10**	**Framenet**	**WN18RR**	**FB15-237**
Previous tensor factorization models								
RESCAL	74.54	51.85	3.01	18.53	0.41	41.23	40.45	69.71
Non Negative RESCAL	71.29	**55.87**	51.67	40.13	**15.06**	42.08	32.49	70.60
Regularized/Constrained tensor factorization models								
Linear + Reg	**74.77**	53.55	46.44	38.43	14.9	30.65	24.01	55.38
Quad + Reg	74.6	52.09	52.26	37.79	14.9	41.64	20.53	**71.19**
Linear + Constraint	71.5	37.55	42.62	27.9	10.86	23.07	26.82	46.80
★ Quad + Constraint	74.37	42.15	**78.21**	**62.41**	13.53	**58.23**	**63.5**	36.63

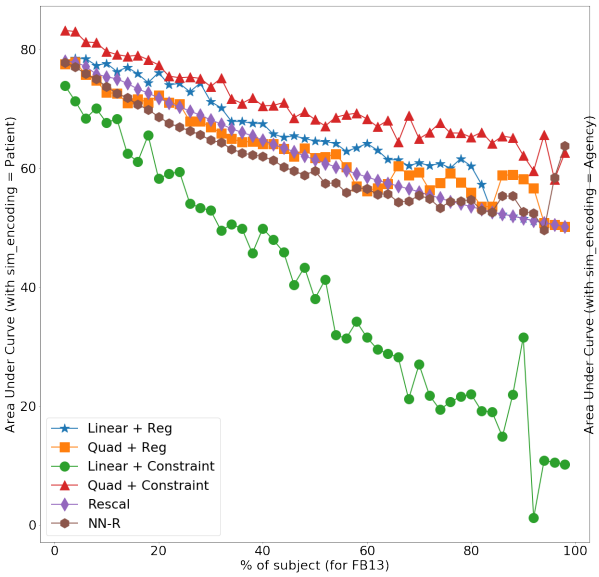
Table 4: Fact prediction performance for all models using different metrics: AUC, micro-averaged F1, and macro-averaged F1. Linear + Reg is the linear tensor decomposition with regularization on $\mathcal{R}$. Quad + Reg, our previous work [21], is the quadratic tensor decomposition with regularization on $\mathcal{R}$. Linear + Constraint and Quad + Constraint are the linear and quadratic tensor decomposition with constraints on $\mathcal{R}$ incorporated as a Lagrange model multiplier. Transitivity similarity measure is used as prior. The ★ next to an algorithm means it performed best overall measured with statistically significant using Wilcoxon paired rank sum test at significance level of 1% (0.01) .

Figure 4 shows the behavior of different tensor based models when 2% to 100% of the subjects are used, where 100% represents the original dataset. Each of the tensor based models benefits when fewer subjects are considered, increasing the knowledge graph's density. Among all the models, Linear+Constraint model improves significantly faster when the number of subjects is reduced irrespective of the similarity metric, eventually achieving comparable performance with other tensor based models. The Quad+Constraint model performs the best irrespective of the graph's density.

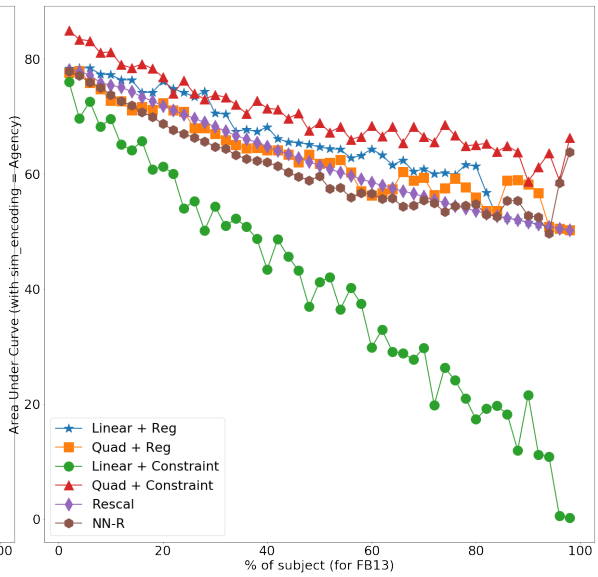


(a) Transitivity Similarity Matrix

(b) Reverse Transitivity Similarity Matrix



(c) Patient Matrix



(d) Agency Similarity Matrix

Figure 4: Increasing performance of tensor based model when reducing % of subjects in a knowledge graph. Here 100% represent the original dataset.

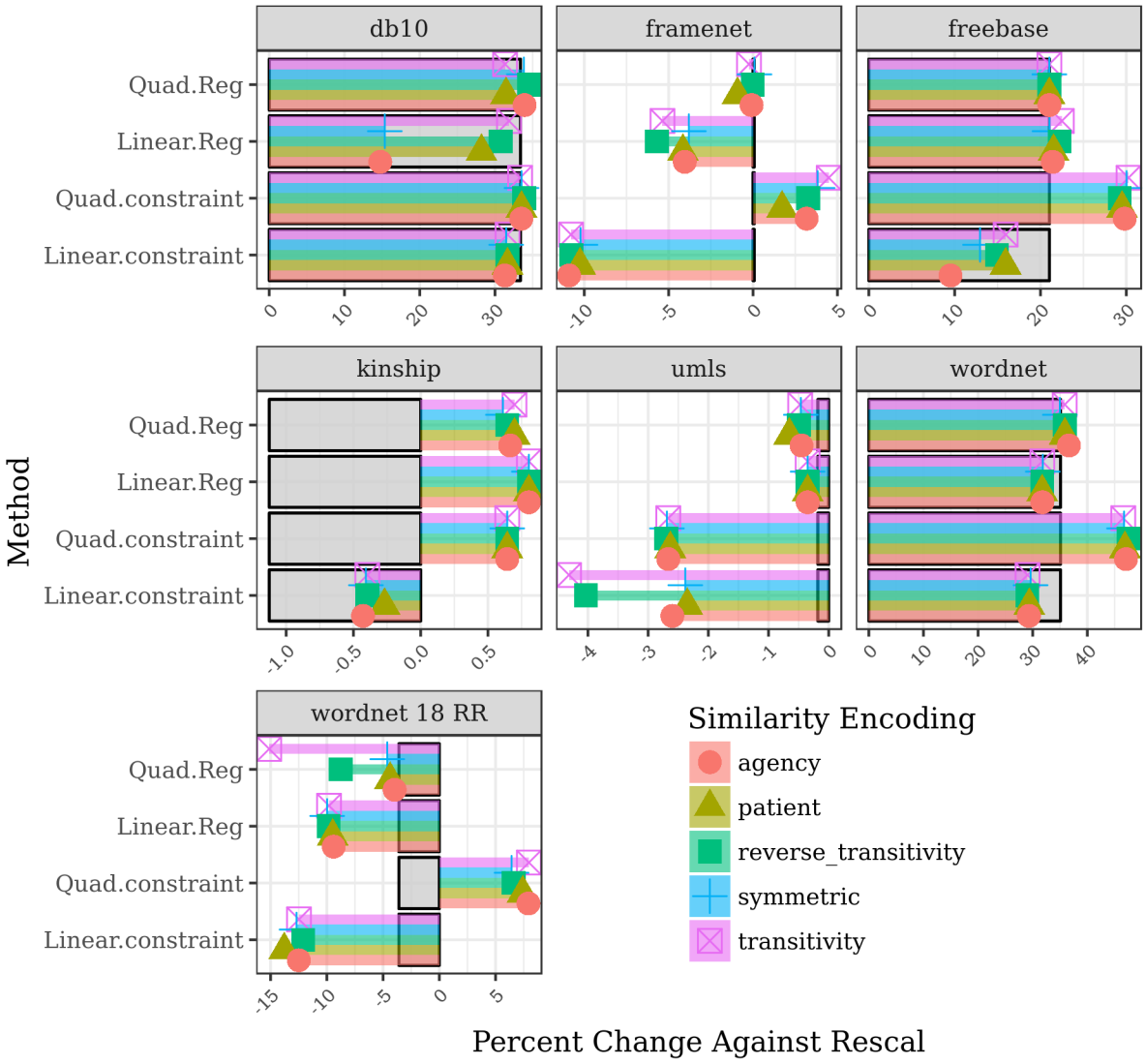


Figure 5: The percent change in AUC that our four models, each with the five different similarity encoding C methods, achieve over RESCAL. The percent change that non-negative RESCAL has over RESCAL is show via the gray boxes.

4.4.3. Effect of different similarity encoding

Figure 5 describes the relative changes in performance of each of the similarity metrics introduced in Section 3.2. Here we examine the performance of these instances of our framework vs. the well studied RESCAL model. The figure shows the percent change of the methods against RESCAL (grouped by how we encode the knowledge). The gray boxes show the percent change of Non-Negative RESCAL vs. RESCAL. This shows how our approach is doing against both baselines.

Most of the similarity encoding approaches perform equally well. However, the encoding can yield a significant performance gain, especially for certain datasets. Consider the dataset db10 (top left) using Linear+Regularized. Here the *agency* and *symmetric* similarity encodings give poor performance. However, when the encoding is changed to *transitivity* or *reverse_transitivity* there is a large gain in performance. On the other hand if WN18RR is considered, *transitivity* and *reverse_transitivity* with the Linear+Regularized model both perform poorly. The Linear+Constraint model performs similarly for all kinds of encoding. Moreover, Quad+Constraint performs consistently well compared

to all the baselines without being affected by the similarity encoding.

In general, while we find that different kinds of similarity encoding methods *can*, and do, influence performance, on the datasets examined here we can see the effect of how that knowledge is encoded. For example, whether a similarity encoding uses a symmetric or transitive approach may be less important than whether or not accurate knowledge is encoded at all. That is, the knowledge enrichment that the encoding provides can result in effective model generalization beyond what simple, knowledge-poor regularizations, such as a simple Frobenius norm regularization, provides.

5. Comparison with TransE, DistMult and ComplEx

This section gives a detailed comparison of our models with TransE, DistMult, and ComplEx, each of which uses a different approach to learn embeddings of entities and relationships, as described in Section 2. We demonstrate that our tensor based method perform significantly better on the fact prediction task for the datasets FB13 and WN18RR and is a close second for the FB15-237 dataset, as shown in Table 5. We also show that including prior information using relation similarity results in a significant performance gain for the fact prediction task.

5.1. Evaluation protocol and datasets

Link ranking tasks are useful for recommendation systems and have been used in previous work to determine performance of a system to predict missing links in a multi-relational data. Each fact in the data is a triple (s, r, o) where s and r are given and each entity is treated as a potential object o to predict its score and sorted rank. If the object has rank above a given threshold, it is considered a hit and is used to measure the performance of a recommendation system.

While calculating the performance of the system, TransE considers translation from source to object for a given relation and vice-versa to calculate the mean rank. Such evaluation protocol may hold true when recommending the top-n links and may not generally hold for relations like "hasParent" or "bornIn'. For example, [Albert_Einstein · bornIn · Germany] is a valid fact, but [Germany · bornIn · Albert_Einstein] is not. Hence we considered translation from source to object only and compare our approach with TransE accordingly. Similarly for the DistMult and ComplEx. Moreover, as the fact prediction task is one of binary classification, we consider a fixed threshold for all relation such that if the score exceeds it, the relation is considered positive/correct else negative/incorrect.

We follow what we believe to be an advisable practice having a single threshold for all relations, rather than using hyperparameters for relation-specific thresholds that mist be tuned or learned. Part of our motivation is knowing that the relation thresholds used in [25] are not publicly available.

As TransE, DistMult, and ComplEx have been evaluated on the link ranking task, it considers only *correct* links and no *incorrect* links. Hence the available dataset contains only positive examples. We consider both *positive* and *negative* links while comparing performance. We evaluated the performance using the standard AUC metric. We used the TransE implementation made available by the authors[4] and set the hyperparameters as mentioned in the paper. For DistMult and ComplEx we used the code available from the author[5] and set the hyperparameters to find the learning rate and epoch that gave best performance.

We used the FB13, WN18RR and FB15-237 datasets and created a training, test and validation file for each. In order to generate *incorrect* links, we considered a stratified testing dataset with 60% positive instances and randomly generated 40% negative instances to keep testing consistent with other datasets. Negative instances were created by keeping the subject and relation fixed and randomly sampling from the pool of objects such that the result did not overlap with positive test instances. We maintained the same distribution of train, test and validation as mentioned in [14]. As mentioned before, beside stratified-weighted sampling we considered balanced and challenged datasets available from [25], which we call WN11NTN and FB13NTN, that contain equal number of positive and negative examples. We evaluated them for the sake of completeness and briefly discuss the results in the next Section.

[4] https://github.com/glorotxa/SME

[5] https://github.com/ttrouill/complex

Table 5: Fact prediction evaluation of FB13, WN18RR and FB15-237 by all systems and models

Dataset	FB13				WN18RR				FB15-237			
Metric	AUC	F1		ACC	AUC	F1		ACC	AUC	F1		ACC
		Macro	Micro			Macro	Micro			Macro	Micro	
Rescal	80	4.08	40.00	40.00	69.63	38.18	49.8	49.8	97.61	94.03	73.41	94.03
Non Neg Rescal	77.76	40.95	51.38	51.38	67.41	67.41	45.52	45.52	**97.81**	**94.48**	73.59	**94.48**
TransE	52.3	16.497	40.72	42.76	68.39	46.91	62.1	62.08	50.84	41.11	4.21	41.12
DistMult	54.36	29.2	53.76	61.97	67.39	37.76	61	60.88	70.28	64.65	45.33	64.65
ComplEx	61.09	28.86	54.06	53.32	67.61	34.235	60.88	61.14	67.64	61.59	35.12	61.59
Linear+Reg	76.51	35.44	50.6	55.08	68.69	28.95	48.9	48.9	96.49	91.08	59.59	91.08
Quad+Reg	75.15	34.85	50.52	54.62	68.46	17.96	48.5	48.5	97.2	92.91	**73.8**	92.91
Linear+Constraint	73.23	27.82	44.72	47.04	66.66	26.99	44.72	44.72	80.00	43.53	1.06	43.53
Quad+Constraint	**82.49**	**56.48**	**59.04**	**66.48**	**81.86**	**59.09**	**62.54**	**65.49**	94.59	84.34	53.56	84.34

5.2. Analysis and discussion

Table 5 shows the performance of previous tensor based models, with TransE, DistMult, ComplEx and our models. Our Quad+Constraint model provides significant improvement over TransE, DistMult, and ComplEx.

One reason our models outperform TransE and DistMult is that the embeddings learned by these system is task-specific and are more suitable for a link ranking task than for a fact prediction one. The results for ComplEx suggest that the embedding learning method in complex space is work better for link ranking than fact prediction. When comparing the baseline methods DistMult and ComplEx on balanced WN11NTN and FB13NTN datasets, our approach performed better with a 4-5% absolute improvement, indicating that the current embedding based method are better suited for link rankingthan fact prediction.

For the FB15-237 dataset with 237 relations, the quadratic based tensor models, i.e., Rescal, Non-Negative Rescal, Quad+Reg, and Quad+Constraint, give comparable or best AUC scores compared to TransE, DistMult, and ComplEx. Moreover, considering other metrics, the quadratic models, either regularized or constrained, perform better overall (as seen in the F1-Macro performance) and also at individual level (as seen in F1-Micro). On the other hand, the lower score of the Linear+Constraint model is due to it frequently predicting a given fact to be incorrect. Comparing the performance of linear models, we note that regularizing embedding model performs better than the constraint one and that the quadratic versions dominate their linear counterparts.

A review of the results in Tables 4 and 5 show that our Quad+Constraint model is better overall, and there is significance improvement when graph density is very low. We believe that the lack of information inherent in a relatively sparse graph is better captured by the constraint introduced by the similarity term. Moreover, Table 5 suggests that the embedding learned using DistMult, ComplEx and TransE work well for a link ranking task and less so for a fact prediction one. In contrast the tensor based model perform better at fact prediction task. Moreover, RESCAL and Non-Negative RESCAL perform poorly compared to our models when the graph density is low, which again demonstrate the effect of constraining the embedding using similarity for a fact prediction task.

5.3. Per Relation Analysis with F1-macro and F1-micro

Figures 6 and 7 show the models' performance broken down by relation. For simplicity of analysis, we selected the WN18RR and FB13 datasets. We first consider FB13. To better understand these results we argue we can group the FB13 relations in to three categories: (i) logically symmetric relations, (ii) knowledge-graph transitive relations, and (iii) what we refer to as *hub* relations.

Logically symmetric relations, like people/marriage/spouse, satisfy the normal definition of a symmetric relation: for a relation r and entities x and y, if $r(x, y)$ is true, then $r(y, x)$ is also true. We identified only one FB13 logically symmetric relation. This contrasts with KG transitive relations that, while not necessarily representing logically

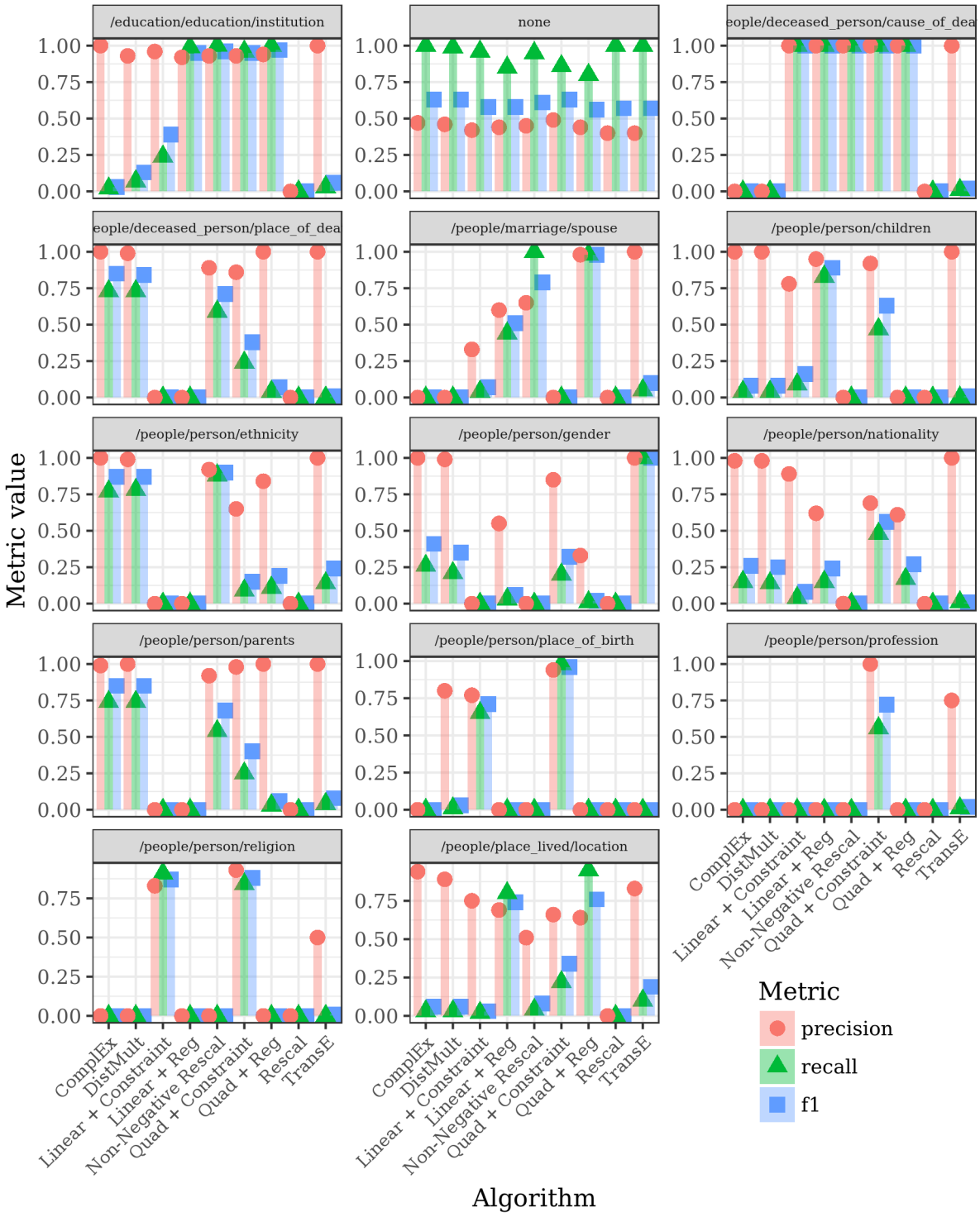


Figure 6: Precision, recall, and F_1 per relation for the FB13 dataset.

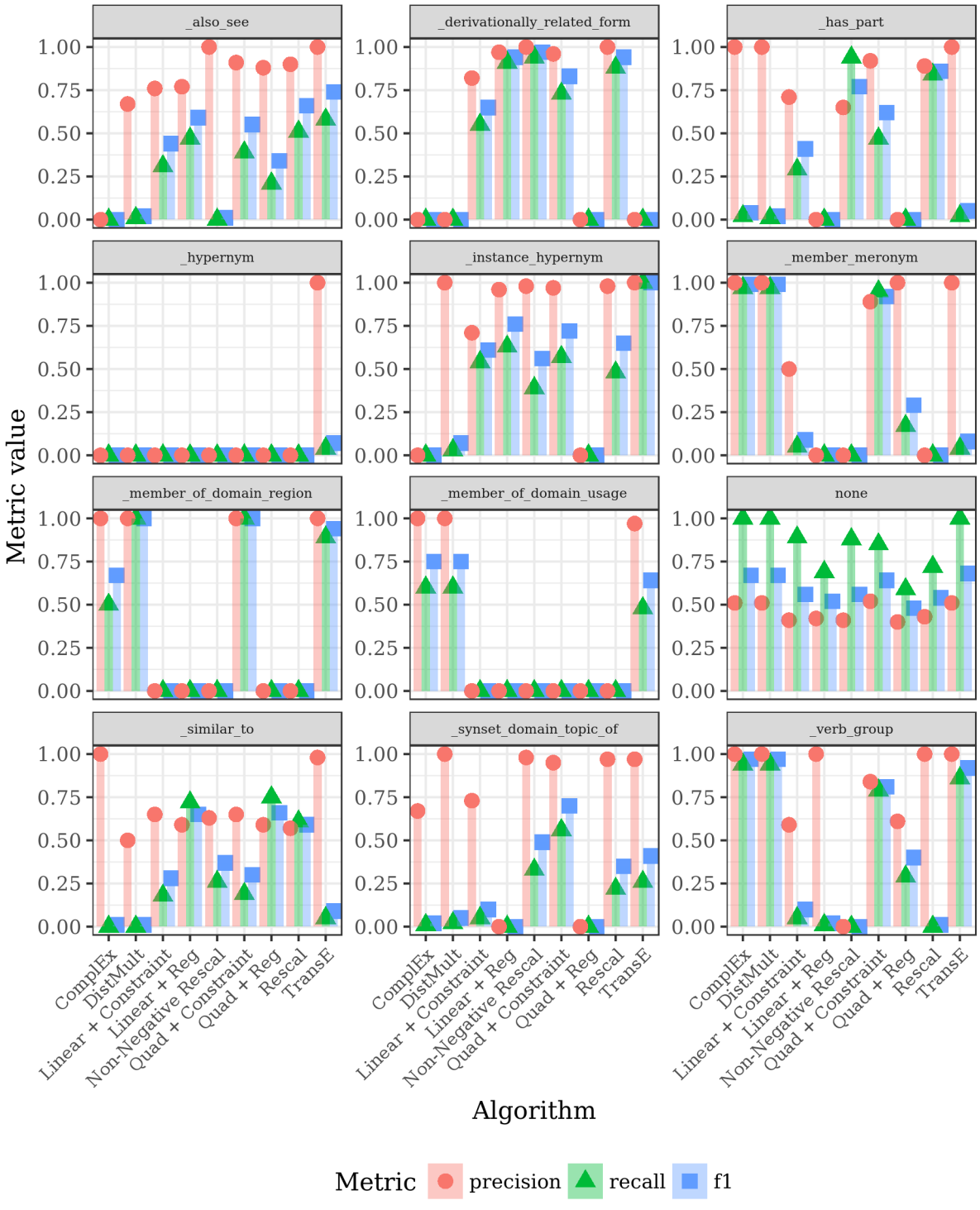


Figure 7: Precision, recall, and F_1 per relation for the WN18RR dataset.

Table 6: Running times (in seconds) per iteration for each of the algorithms on the eight evaluation datasets; – means not available.

	Kinship	**UMLS**	**WN18**	**FB13**	**DB10k**	**FrameNet**	**WN18RR**	**FB15-237**
relations	*26*	*49*	*18*	*13*	*140*	*16*	*11*	*237*
entities	*104*	*135*	*40,943*	*81,061*	*4,397*	*62,344*	*40,943*	*14,541*
RESCAL	0.01	0.07	0.26	0.29	1.7	0.1	0.11	21.96
NN-R	0.01	0.08	0.38	0.41	8.72	0.22	0.12	46.31
TransE	–	–	8.26	19.37	–	–	4.45	–
DistMult	–	–	20.27	56.7	–	–	14.4	48.93
ComplEx	–	–	71.86	156.24	–	–	41.3	157.53
Linear+Regularized	0.02	0.04	0.85	1.02	4.86	0.39	0.43	23.38
Quad+Regularized	0.03	0.11	0.52	0.62	4.78	0.24	0.25	52.96
Linear+Constrained	0.02	0.1	0.33	0.39	3.13	0.15	0.15	43.68
Quad+Constrained	0.03	0.09	0.69	0.71	3.26	0.12	0.12	54.28

transitive relations, can be productively combined with other relations to form meaningful relation chains. FB13 KG transitive relations are people/person/children, people/place_lived/location, people/person/parents. Finally, we identify the remaining FB13 relations as *hub* relations that do not readily yield logically symmetric nor knowledge graph transitive relations. For example, subjects of hub relations like people/deceased_person/cause_of_death and person/person/nationality cannot be easily used, under the FB13 schema, as objects of other hub relations.[6]

Both ComplEx and DistMult generally have high precision but suffer from low recall resulting in poor F1 scores. On the other hand, TransE gives high precision for hub relations. For logically symmetric relations like spouse, Quad+Regularized does well, which makes sense as the relationship is two-way and is captured by the quadratic objective function. Moreover, Linear+Constraint performs poorly as it tries to model the behavior in opposition to the reality that either of the relation's arguments could be used as the subject or the object. Quad+Constraint performs better compared to other models across all relation except spouse, indicating that such symmetric relations are better modeled with regularization than a Lagrangian constraint.

Second, we examine the relation-level F1 performance on WN18RR. As seen in Figure 7, the Quad+Constraint model performs consistently well across all of the relations—especially when compared to the other methods. We believe that this stems from the way similarity is incorporated and the embeddings learned. For example, consider a relation _synset_domain_topic_of and the heatmap shown in Figure 3a. We believe the better performance stems from the level of similarity shared between the four relations—_synset_domain_topic_of, _instance_hypernym, _derivationally_relation_form, and _has_part.

6. Time complexity

The asymptotic time and space complexity of our models are the same as RESCAL's. Table 6 shows the run times per iteration taken by each approach to update the unknown variables. We ran all for a maximum of 100 iterations and report the average running time per iteration. In the case of TransE, we considered an epoch as an iteration, since each iteration sees all the data values.

As expected, the running time increases with the number of relations, with the FB15-237 dataset, which has 237 relations, taking the longest and DB10K with 140 relation in second place. The non-negative constraint on non-negative RESCAL increases the running time of that model. Regularized models require more time when compared to

[6]We identify the following FB13 relations as hub relations: people/deceased_person/cause_of_death, person/person/nationality, people/person/place_of_birth, education/education/institution, people/person/gender, people/person/place_of_death, people/person/religion, people/person/ethnicity, and people/person/profession.

other models due to presence of additional terms to bring $\mathbf{A}_1$ and $\mathbf{A}_2$ closer, which introduces additional computation during the update rules.

TransE has much longer running time per iteration since it computes pairwise distances among positive and negative instances, so its time increases with the number of entities. Similarly, DistMult and ComplEx consume considerable amount of time as we believe it is due to running on CPU. We also saw that running on a GPU, DistMult executed faster than TransE (3.73sec for WN18, 10.33sec for FB13, and 2.75sec for WN18RR) and ComplEx took longer than TransE (9.77sec for WN18, 21.2sec for FB13, and 8.75sec for WN18RR).

7. Effect of ρ on convergence

To illustrate convergence of the Linear+Regularized model, Figure 8 shows the effect of ρ on the maximum component–wise relative change in consecutive iterations for the variables during optimization. If $\boldsymbol{z}_t$ is the vector of all of the unknowns at iteration t, i.e. $\mathbf{A}_1$, $\mathbf{A}_2$, $\mathcal{R}$, we use the following equation to measure the maximum relative change on the unknowns at each iteration.

$$\delta(\boldsymbol{z}_t, \boldsymbol{z}_{t+1}) \quad = \quad \max_i \left| \frac{z_t(i) - z_{t+1}(i)}{z_t(i) + z_{t+1}(i)/2} \right| \tag{33}$$

For each value of ρ we follow a cold-start procedure, i.e., for each new value of ρ we randomly initialize all the variables. The termination condition is that we reached the maximum iteration number (chosen as 100) or that the maximum change δ in the unknown is below a threshold (chosen as 10^{-6}). Here, the blue dots with dashed lines indicate the maximum relative change δ vs. iterations when $\rho = \infty$.

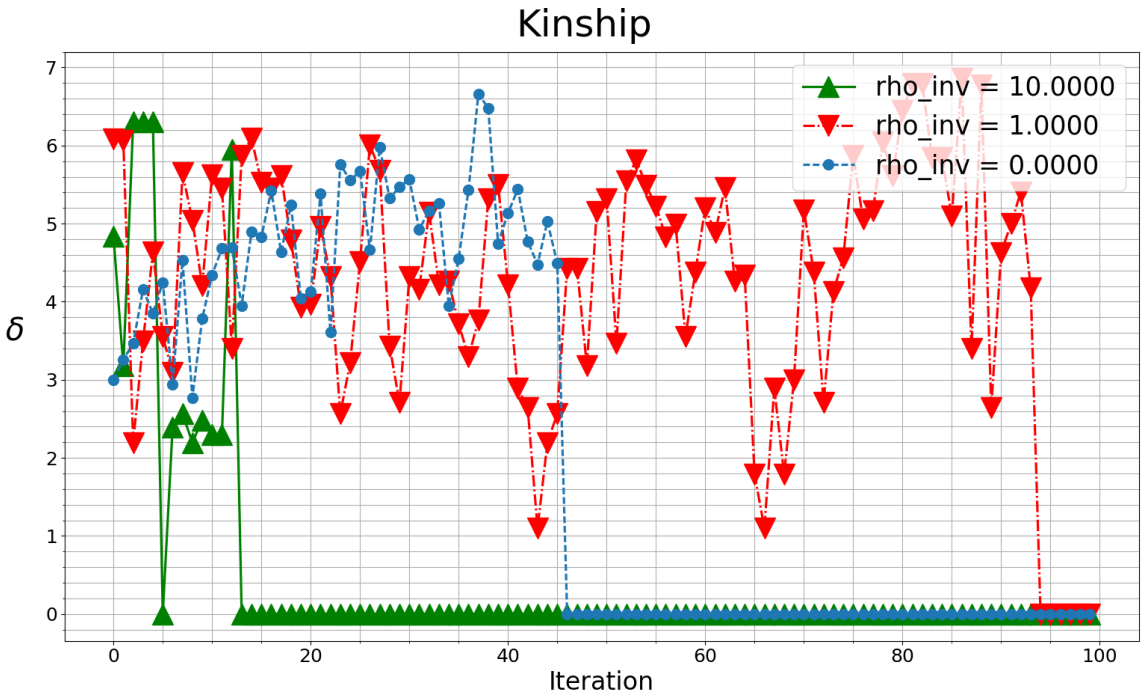
8. Conclusions and future work

We proposed a framework for learning knowledge-endowed entity and relation embeddings. The framework includes four readily obtainable novel models that generalize existing efforts. Two of the models optimize a linear factorization objective and two a quadratic one. We evaluated the quality of embeddings on the task of fact predictionand demonstrated significant improvements ranging from 5% to 50% over state-of-the-art tensor decomposition models and translation based models on a number of real-world datasets. 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 among our models.

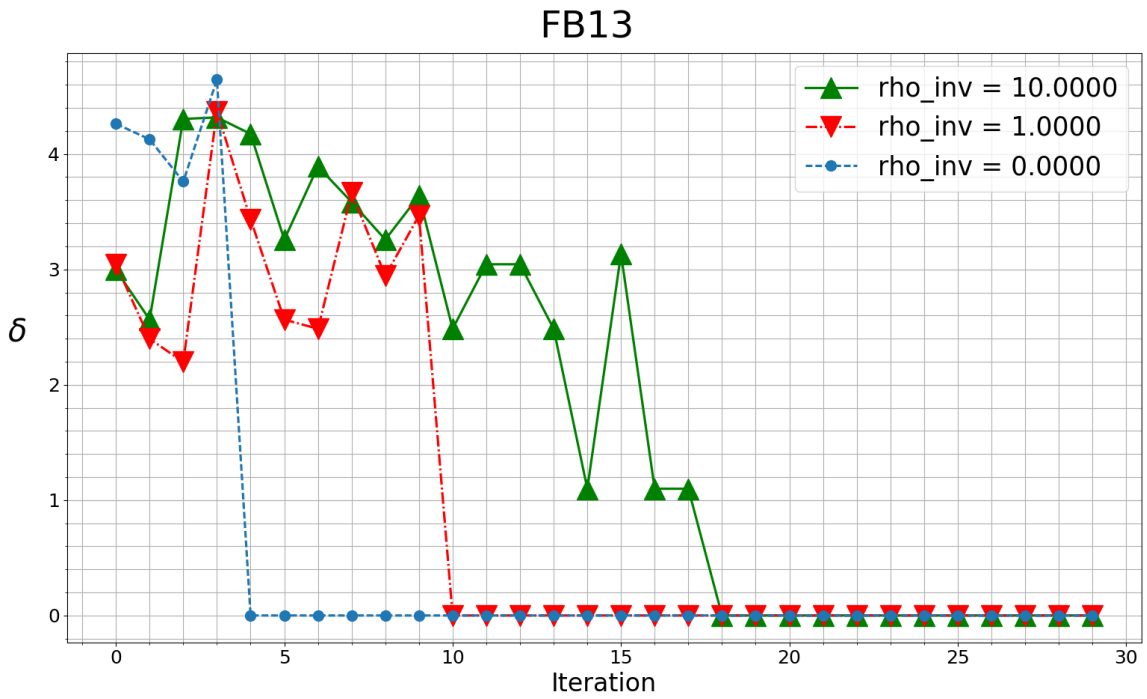
We observed that for the task of fact prediction, better embeddings are obtained by the Quadratic+Constrained model. Linear models are better suited when there is a one way interaction from subject to the object in which the object cannot also serve as a subject. We find the quadratic models perform better in general, irrespective of the position of the entity as subject or object. Constraint-based models perform better compared to regularized models and constraint-based models with a quadratic objective are better suited for the task—irrespective of the sparsity of the knowledge graph. We showed detailed experimental analyses of the model's strengths and weaknesses in predicting facts with particular relations, and we provided complementary qualitative analysis of commonalities among those relations. On the theoretical side, we proved that the Linear+Regularized model has the desirable property of convergence and illustrated its convergence on two standard benchmark datasets.

Our future work will explore the use of our models in several application contexts that use natural language understanding systems to extract entities, relations and events from text documents. Such systems can benefit from a *fact prediction* module that can help eliminate facts extracted in error. Our experience in the NIST Knowledge Base Population (KBP) tasks [50] showed the need to independently assess the quality of extracted relations. The KBP tasks are well suited for an approach like our trained on general-purpose knowledge graphs like DBpedia, Freebase and Wikidata.

A second application is a system we are developing to identify possible cybersecurity attacks from data collected from host computers and networks represented in an RDF knowledge graph using the Unified Cybersecurity Ontology [51]. This system [52] draws on background knowledge encoded in graphs populated with information extracted from cybersecurity-related documents and from semi-structured data from cybersecurity threat intelligence data sources.



(a) Convergence on Kinship



(b) Convergence on FB13

Figure 8: Change in the unknowns/variables at each iteration for three different values of $\rho = 0.1, 1, \infty$. Here *rho_inv* equals of $1/\rho$.

A third application is as one component of a general-purpose system under development for cleaning noisy knowledge graphs [53]. Its current architecture consists of an ensemble of modules that try to identify, characterize and explain different types of errors that many current text information extraction systems can make. Using a version of our approach to see of an extracted fact is predicted or not will be a useful feature.

Acknowledgement Partial support for this research was provided by gifts from IBM through the IBM AI Horizons Network and from Northop Grumman Corporation.

References

[1] P. Ernst, A. Siu, G. Weikum, Knowlife: a versatile approach for constructing a large knowledge graph for biomedical sciences, BMC bioinformatics (2015).
[2] M. Mintz, S. Bills, R. Snow, D. Jurafsky, Distant supervision for relation extraction without labeled data, in: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2, Association for Computational Linguistics, 2009, pp. 1003–1011.
[3] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, Z. Ives, Dbpedia: A nucleus for a web of open data, in: International Semantic Web Conference, Springer, 2007, pp. 722–735.
[4] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, J. Taylor, Freebase: a collaboratively created graph database for structuring human knowledge, in: Proceedings of the 2008 ACM SIGMOD international conference on Management of data, ACM, 2008, pp. 1247–1250.
[5] G. A. Miller, Wordnet: a lexical database for english, Communications of the ACM 38 (1995).
[6] D. Ringler, H. Paulheim, One knowledge graph to rule them all? analyzing the differences between DBpedia, YAGO, Wikidata & co., in: Joint German/Austrian Conference on Artificial Intelligence, Springer, 2017, pp. 366–372.
[7] DBpedia, Dbpedia sparql endpoint, `http://dbpedia.org/sparql`, 2017.
[8] Y. Bengio, A. Courville, P. Vincent, Representation learning: A review and new perspectives, IEEE transactions on pattern analysis and machine intelligence 35 (2013) 1798–1828.
[9] M. Nickel, V. Tresp, H.-P. Kriegel, A three-way model for collective learning on multi-relational data, in: Proceedings of the 28th international conference on machine learning (ICML-11), 2011, pp. 809–816.
[10] T. G. Kolda, B. W. Bader, Tensor decompositions and applications, SIAM review (2009).
[11] T. Franz, A. Schultz, S. Sizov, S. Staab, Triplerank: Ranking semantic web data by tensor decomposition, in: International Semantic Web Conference, Springer, 2009, pp. 213–228.
[12] D. Krompass, S. Baier, V. Tresp, Type-constrained representation learning in knowledge graphs, in: International Semantic Web Conference, Springer, 2015.
[13] D. Krompass, M. Nickel, X. Jiang, V. Tresp, Non-negative tensor factorization with RESCAL, in: Tensor Methods for Machine Learning, ECML Workshop, 2013.
[14] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, in: NIPS, 2013, pp. 2787–2795.
[15] B. Yang, W.-T. Yih, X. He, J. Gao, L. Deng, Embedding entities and relations for learning and inference in knowledge bases, arXiv preprint arXiv:1412.6575 (2014).
[16] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, G. Bouchard, Complex embeddings for simple link prediction, in: International Conference on Machine Learning, 2016, pp. 2071–2080.
[17] T. Demeester, T. Rocktäschel, S. Riedel, Lifted rule injection for relation embeddings, in: Conference on Empirical Methods in Natural Language Processing, 2016, pp. 1389–1399.
[18] P. Minervini, T. Demeester, T. Rocktäschel, S. Riedel, Adversarial sets for regularising neural link predictors, in: 33rd Conference on Uncertainty in Artificial Intelligence, 2017, pp. 1–10.
[19] B. J. Lengerich, A. L. Maas, C. Potts, Retrofitting distributional embeddings to knowledge graphs with functional relations, arXiv preprint arXiv:1708.00112 (2017).
[20] P. Minervini, L. Costabello, E. Muñoz, V. Nováček, P.-Y. Vandenbussche, Regularizing knowledge graph embeddings via equivalence and inversion axioms, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2017, pp. 668–683.
[21] A. Padia, K. Kalpakis, T. Finin, Inferring relations in knowledge graphs with tensor decompositions, in: International Conference on Big Data, IEEE, 2016, pp. 4020–4022.
[22] H. Paulheim, Knowledge graph refinement: A survey of approaches and evaluation methods, Semantic web 8 (2017) 489–508.
[23] M. Nickel, K. Murphy, V. Tresp, E. Gabrilovich, A review of relational machine learning for knowledge graphs, Proceedings of the IEEE 104 (2016) 11–33.
[24] A. Bordes, X. Glorot, J. Weston, Y. Bengio, A semantic matching energy function for learning with multi-relational data, Machine Learning (2014) 233–259.
[25] R. Socher, D. Chen, C. D. Manning, A. Ng, Reasoning with neural tensor networks for knowledge base completion, in: Advances in neural information processing systems, 2013, pp. 926–934.
[26] Z. Wang, J. Zhang, J. Feng, Z. Chen, Knowledge graph embedding by translating on hyperplanes., in: AAAI, volume 14, 2014, pp. 1112–1119.
[27] M. Nickel, L. Rosasco, T. A. Poggio, et al., Holographic embeddings of knowledge graphs., in: AAAI, volume 2, 2016, pp. 3–2.
[28] K. Hayashi, M. Shimbo, On the equivalence of holographic and complex embeddings for link prediction, arXiv preprint arXiv:1702.05563 (2017).
[29] S. Guo, Q. Wang, L. Wang, B. Wang, L. Guo, Knowledge graph embedding with iterative guidance from soft rules, in: AAAI, 2018.

[30] L. A. Galárraga, C. Teflioudi, K. Hose, F. Suchanek, Amie: association rule mining under incomplete evidence in ontological knowledge bases, in: Proceedings of the 22nd international conference on World Wide Web, ACM, 2013, pp. 413–422.
[31] P. Miettinen, Boolean tensor factorizations, in: 11th International Conference on Data Mining (ICDM), IEEE, 2011, pp. 447–456.
[32] D. Erdos, P. Miettinen, Discovering facts with boolean tensor tucker decomposition, in: Proceedings of the 22nd¡ international conference on Conference on information & knowledge management, ACM, 2013, pp. 1569–1572.
[33] M. Nickel, V. Tresp, H.-P. Kriegel, Factorizing YAGO: scalable machine learning for linked data, in: Proceedings of the 21st International Conference on World Wide Web, ACM, 2012, pp. 271–280.
[34] P. Minervini, C. d'Amato, N. Fanizzi, F. Esposito, Leveraging the schema in latent factor models for knowledge graph completion, in: Proceedings of the 31st Annual ACM Symposium on Applied Computing, SAC '16, ACM, New York, NY, USA, 2016, pp. 327–332. URL: http://doi.acm.org/10.1145/2851613.2851841. doi:10.1145/2851613.2851841.
[35] L. Han, T. Finin, A. Joshi, D. Cheng, Querying RDF Data with Text Annotated Graphs, in: 27th International Conference on Scientific and Statistical Database Management, 2015.
[36] J. M. Gawron, Improving sparse word similarity models with asymmetric measures., in: ACL (2), 2014, pp. 296–301.
[37] N. Parikh, S. Boyd, Proximal algorithms, Foundations and Trends in Optimization 1 (2014) 123–231.
[38] B. W. Bader, R. A. Harshman, T. G. Kolda, Temporal analysis of semantic graphs using ASALSAN, in: Proceedings of the 7th International Conference on Data Mining, IEEE, 2007, pp. 33–42.
[39] D. Kingma, J. Ba, Adam: A method for stochastic optimization, in: Proceedings of the Third International Conference on Learning Representations, 2014.
[40] C. F. Baker, C. J. Fillmore, J. B. Lowe, The berkeley framenet project, in: 36th Annual Meeting of the ACL and 17th Int. Conf. on Computational Linguistics, ACL, 1998, pp. 86–90.
[41] M. Nickel, Tensor factorization library, https://github.com/mnick/scikit-tensor, 2013. [Online; accessed 10-Aug-2017].
[42] R. A. Harshman, M. E. Lundy, Parafac: Parallel factor analysis, Computational Statistics & Data Analysis (1994).
[43] R. Bro, Parafac. tutorial and applications, Chemometrics and intelligent laboratory systems 38 (1997) 149–171.
[44] C. Kemp, J. B. Tenenbaum, T. L. Griffiths, T. Yamada, N. Ueda, Learning systems of concepts with an infinite relational model, in: Proceedings of the 21st National Conference on Artificial Intelligence, AAAI, 2006, pp. 381—-388.
[45] T. Dettmers, P. Minervini, P. Stenetorp, S. Riedel, Convolutional 2d knowledge graph embeddings, arXiv preprint arXiv:1707.01476 (2018). Extended AAAI18 paper.
[46] K. Toutanova, D. Chen, Observed versus latent features for knowledge base and text inference, in: Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality, 2015, pp. 57–66.
[47] R. Kadlec, O. Bajgar, J. Kleindienst, Knowledge base completion: Baselines strike back, in: Proceedings of the 2nd Workshop on Representation Learning for NLP, 2017, pp. 69–74.
[48] C. F. Baker, C. J. Fillmore, J. B. Lowe, The berkeley framenet project, in: Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1, Association for Computational Linguistics, 1998, pp. 86–90.
[49] K. Toutanova, D. Chen, P. Pantel, H. Poon, P. Choudhury, M. Gamon, Representing text for joint embedding of text and knowledge bases, in: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, 2015, pp. 1499–1509.
[50] T. Finin, D. Lawrie, P. McNamee, J. Mayfield, D. Oard, N. Peng, N. Gao, Y.-C. Lin, J. MacKin, T. Dowd, HLTCOE Participation in TAC KBP 2015: Cold Start and TEDL, in: 8th Text Analysis Conference, NIST, 2015.
[51] Z. Syed, A. Padia, M. L. Mathews, T. Finin, A. Joshi, UCO: A Unified Cybersecurity Ontology, in: Proceedings of the AAAI Workshop on Artificial Intelligence for Cyber Security, AAAI Press, 2016.
[52] S. N. Narayanan, A. Ganesan, K. P. Joshi, T. Oates, A. Joshi, T. Finin, Early detection of cybersecurity threats using collaborative cognition, in: Proceedings of the 4th IEEE International Conference on Collaboration and Internet Computing (CIC), IEEE, 2018.
[53] A. Padia, Cleaning Noisy Knowledge Graphs, in: Proceedings of the Doctoral Consortium at the 16th International Semantic Web Conference, volume 1962, CEUR Workshop Proceedings, 2017.
[54] M. Nickel, Tensor factorization for relational learning, Ph.D. thesis, Ludwig-Maximilians-Universität München, 2013.
[55] L. Grippof, M. Sciandrone, Globally convergent block-coordinate techniques for unconstrained optimization, Optimization methods and software 10 (1999).

Appendix A. Proof of convergence

Appendix A.1. Propositions and Lemma

We assume that order–3 tensors $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{N \times N \times K}$, and $\boldsymbol{\mathcal{R}} \in \mathbb{R}^{N' \times N' \times K'}$, matrix $\mathbf{A} \in \mathbb{R}^{N \times N'}$, and symmetric matrix $\mathbf{C} = (c_{ij}) \in \mathbb{R}^{K \times K}$.

Proposition 1. *For any matrices* $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ *we use the following properties of Kronecker products:*

$$(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = (\mathbf{AC}) \otimes (\mathbf{BD}) \tag{A.1}$$
$$(\mathbf{A} \otimes \mathbf{B})^T = \mathbf{A}^T \otimes \mathbf{B}^T \tag{A.2}$$
$$(\mathbf{A} \otimes \mathbf{B})^{1/2} = \mathbf{A}^{1/2} \otimes \mathbf{B}^{1/2} \tag{A.3}$$
$$(\mathbf{A} \otimes \mathbf{B})^{+} = \mathbf{A}^{+} \otimes \mathbf{B}^{+} \tag{A.4}$$
$$(\mathbf{A} \otimes \mathbf{I}_n + \alpha \mathbf{I}_m)^{-1} = (\mathbf{A} + \alpha \mathbf{I}_{m-n})^{-1} \otimes \mathbf{I}_n \tag{A.5}$$
$$\mathbf{A} \otimes (\mathbf{B} \otimes \mathbf{C}) = (\mathbf{A} \otimes \mathbf{B}) \otimes \mathbf{C} \tag{A.6}$$
$$\mathrm{vec}\,(\mathbf{ABC}) = (\mathbf{C}^T \otimes \mathbf{A})\mathrm{vec}\,(\mathbf{B}) \tag{A.7}$$

Proposition 2 ([10])**.** *For order–N tensors* $\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}}$ *and sequence of matrices* $\mathbf{A}^{(i)}$*,* $i = 1, 2, \ldots, N$*,*

$$\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{X}} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \ldots \times_N \mathbf{A}^{(N)} \tag{A.8}$$

if and only if, for all n

$$\mathbf{Y}_{(n)} = \mathbf{A}^{(n)} \mathbf{X}_{(n)} \left(\mathbf{A}^{(N)} \otimes \ldots \otimes \mathbf{A}^{(n+1)} \otimes \mathbf{A}^{(n-1)} \ldots \otimes \mathbf{A}^{(1)} \right)^T \tag{A.9}$$

Further,

$$\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{X}} \times_n \mathbf{A} \textit{ iff } \mathbf{Y}_{(n)} = \mathbf{A}\mathbf{X}_{(n)} \tag{A.10}$$

$$\|\boldsymbol{\mathcal{X}}\|_F^2 = \sum_i \|\mathbf{X}_i\|_F^2 = \sum_i \|\mathrm{vec}\,(\mathbf{X}_i)\,\|_2^2 = \|\mathrm{vec}\,(\boldsymbol{\mathcal{X}})\,\|_2^2 \tag{A.11}$$

Proposition 3. *For any order–3 tensors* $\boldsymbol{\mathcal{X}}$ *and* $\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{X}} \times_1 \mathbf{A}_1 \times_2 \mathbf{A}_2$*, we have the following. Both mode-1 and mode-2 unfoldings of order–3 tensors are block matrices with the same number of blocks, and*

$$\mathbf{X}_{(2)} = [\mathbf{X}'_1 \ldots \mathbf{X}'_k \ldots], \tag{A.12}$$
$$\mathbf{Y}_{(1)} = \mathbf{A}_1 \mathbf{X}_{(1)} (\mathbf{I} \otimes \mathbf{A}_2)^T = [\ldots \mathbf{A}_1 \mathbf{X}_k \mathbf{A}_2^T \ldots] \tag{A.13}$$
$$\mathbf{Y}_{(2)} = \mathbf{A}_2 \mathbf{X}_{(2)} (\mathbf{I} \otimes \mathbf{A}_1)^T = [\ldots \mathbf{A}_2 \mathbf{X}'_k \mathbf{A}_1^T \ldots] \tag{A.14}$$

Proposition 4.

$$\mathrm{argmin}_{\mathbf{X}} \|\mathbf{B} - \mathbf{AXC}\|_F^2 = \mathrm{argmin}_{\mathbf{X}} \|\mathrm{vec}\,(\mathbf{B}) - (\mathbf{C}^T \otimes \mathbf{A})\mathrm{vec}\,(\mathbf{X})\,\|^2 \tag{A.15}$$

and the solution of a least squares problem

$$\mathbf{x} \leftarrow \mathrm{argmin}_{\mathbf{x}} \|\mathbf{Ax} - \mathbf{b}\|^2 \tag{A.16}$$
$$= \mathrm{argmin}_{\mathbf{x}} (\mathbf{x}^T (\mathbf{A}^T \mathbf{A})\mathbf{x} - 2(\mathbf{A}^T \mathbf{b})^T \mathbf{x} + \mathbf{b}^T \mathbf{b}) = \tag{A.17}$$
$$= (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b} = \mathbf{A}^{+} \mathbf{b} \tag{A.18}$$

where $\mathbf{A}^{+}$ *the Moore–Penrose matrix pseudo–inverse (provided* $\mathbf{A}^T\mathbf{A}$ *is full–rank and hence invertible), which is a left–inverse of* $\mathbf{A}$ *with least Frobenius norm among all left–inverses of* $\mathbf{A}$*. Furthermore, its gradient and Hessian are* $2\mathbf{A}^T\mathbf{Ax} - 2\mathbf{A}^T\mathbf{b}$ *and* $2\mathbf{A}^T\mathbf{A}$ *respectively.*

Lemma 1. *For any order–3 tensor* $\boldsymbol{\mathcal{X}}$ *and symmetric matrix* $\mathbf{C}$*, we have that*

$$\|\boldsymbol{\mathcal{X}} \times_3 (\mathrm{diag}(\mathbf{C} \cdot \mathbf{1}_{n_2}) - \mathbf{C})^{1/2}\|_F^2 = \sum_{ij} c_{ij} \|\mathbf{X}_i - \mathbf{X}_j\|_F^2 \tag{A.19}$$

Proof. Suppose that $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$. Consider the tensor's mode–1 unfolding $\mathbf{X}_{(1)} = [\mathbf{X}_1 \mathbf{X}_2 \dots \mathbf{X}_K]$. Clearly, $\text{vec}\left(\mathbf{X}_{(1)}\right) = \text{vec}\left([\text{vec}\left(\mathbf{X}_1\right) \dots \text{vec}\left(\mathbf{X}_K\right)]\right)$. If $\mathbf{L} = \deg(\mathbf{C}) - \mathbf{C}$, where $\deg(\mathbf{C})$ is a diagonal matrix with the row sums of $\mathbf{C}$, then recall ($\mathbf{L}$ is the Laplacian matrix of an undirected graph with weighted adjacency matrix $\mathbf{C}$) that

$$\mathbf{x}^T \mathbf{L} \mathbf{x} = \sum_{ij} c_{ij} (x_i - x_j)^2. \tag{A.20}$$

Consequently, we have

$$\begin{aligned} \sum_{ij} c_{ij} \|\mathbf{X}_i - \mathbf{X}_j\|_F^2 &= \text{vec}\left(\mathbf{X}_{(1)}\right)^T (\mathbf{L} \otimes \mathbf{I}_{n_1 n_2}) \text{vec}\left(\mathbf{X}_{(1)}\right) \\ &= \|(\mathbf{L}^{1/2} \otimes \mathbf{I}_{n_1 n_2}) \text{vec}\left(\mathbf{X}_{(1)}\right)\|_F^2 \end{aligned} \tag{A.21}$$

since $\mathbf{I}^{1/2} = \mathbf{I}$. Using the properties of Kronecker products above, we have

$$\begin{aligned} (\mathbf{L}^{1/2} \otimes \mathbf{I}_{n_1 n_2}) \text{vec}\left(\mathbf{X}_{(1)}\right) &= ((\mathbf{L}^{1/2} \otimes \mathbf{I}_{n_2}) \otimes \mathbf{I}_{n_1}) \text{vec}\left(\mathbf{X}_{(1)}\right) \\ &= \mathbf{I}_{n_1} \mathbf{X}_{(1)} (\mathbf{L}^{1/2} \otimes \mathbf{I}_{n_2}) \\ &= \boldsymbol{\mathcal{X}} \times_1 \mathbf{I} \times_2 \mathbf{I} \times_3 \mathbf{L}^{1/2} \end{aligned} \tag{A.22}$$

Therefore,

$$\sum_{ij} c_{ij} \|\mathbf{X}_i - \mathbf{X}_j\|_F^2 = \|\boldsymbol{\mathcal{X}} \times_1 \mathbf{I} \times_2 \mathbf{I} \times_3 \mathbf{L}^{1/2}\|_F^2 \tag{A.23}$$

■

Appendix A.2. Proof

Recall that our model considers the similarity among the frontal slices of the tensor $\boldsymbol{\mathcal{R}}$. Using Lemma 1, the objective function of our model is

$$\begin{aligned} f(\boldsymbol{\mathcal{R}}, \mathbf{A}) &= \|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{R}} \times_1 \mathbf{A} \times_2 \mathbf{A}\|_F^2 + \lambda_a \|\mathbf{A}\|_F^2 + \lambda_g \|\boldsymbol{\mathcal{R}}\|_F^2 + \\ &\quad \lambda_s \|\boldsymbol{\mathcal{R}} \times_3 \mathbf{S}\|_F^2 \end{aligned} \tag{A.24}$$

where $\mathbf{S} = (\deg(\mathbf{C}) - \mathbf{C})^{1/2}$ and $\boldsymbol{\lambda} \geq \mathbf{0}$. Since f is of degree 4 in $\mathbf{A}$, it will difficult to optimize it efficiently. We employ the split–variable trick [54, 37], by splitting the variable matrix $\mathbf{A}$ into a tensor $\boldsymbol{\mathcal{A}}$ with exactly two frontal square slices $\mathbf{A}_1$ and $\mathbf{A}_2$ and enforce the constraint $\|\mathbf{A}_1 - \mathbf{A}_2\|_F^2 = 0$, to obtain the objective function

$$\begin{aligned} f'(\boldsymbol{\mathcal{R}}, \mathbf{A}_1, \mathbf{A}_2) &= \|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{R}} \times_1 \mathbf{A}_1 \times_2 \mathbf{A}_2\|_F^2 + \lambda_a \|\mathbf{A}_1\|_F^2 + \lambda_a \|\mathbf{A}_2\|_F^2 \\ &\quad + \lambda_g \|\boldsymbol{\mathcal{R}}\|_F^2 + \lambda_s \|\boldsymbol{\mathcal{R}} \times_3 \mathbf{S}\|_F^2 + \lambda_e \|\mathbf{A}_1 - \mathbf{A}_2\|_F^2 \end{aligned} \tag{A.25}$$

Upon finding a minimizer $(\boldsymbol{\mathcal{R}}, \mathbf{A}_1, \mathbf{A}_2)$ of f', we use the point $(\boldsymbol{\mathcal{R}}, (\mathbf{A}_1 + \mathbf{A}_2)/2)$ for our original objective function f.

Because the Frobenius norm is convex, convexity is preserved under affine transformation, and the sum of convex functions is convex, it follows that our function $f'(\boldsymbol{\mathcal{R}}, \mathbf{A}_1, \mathbf{A}_2)$ is only separately convex with respect to $\boldsymbol{\mathcal{R}}$, $\mathbf{A}_1$, and $\mathbf{A}_2$ (i.e block separately convex). Unfortunately, f' is not strictly convex with respect to these three block arguments, and is neither separately convex in just two blocks, which may lead to ALS not converging when trying to optimize f'. Notice that ALS is essentially block-based Gauss–Seidel with 3 blocks of unknowns/variables, the blocks $\boldsymbol{\mathcal{R}}$, $\mathbf{A}_1$, and $\mathbf{A}_2$, since optimizing for one block while keeping the other two blocks fixed is a least–squares problem.

We seek to avoid non-convergence by constructing a modified objective function $\hat{f}_\rho(\boldsymbol{\mathcal{R}}, \mathbf{A}_1, \mathbf{A}_2)$ such that $\lim_{\rho \to \infty} \hat{f}_\rho(\boldsymbol{\mathcal{R}}, \mathbf{A}_1, \mathbf{A}_2) = f'(\boldsymbol{\mathcal{R}}, \mathbf{A}_1, \mathbf{A}_2)$, and for which block–based Gauss–Seidel is guaranteed to converge to minimizer of $\hat{f}_\rho$ for each ρ. To this end, we modify f' to make it strictly convex with respect to each of three blocks of unknowns $\boldsymbol{\mathcal{R}}, \mathbf{A}_1, \mathbf{A}_2$. In particular, we employ a trick used in proximal algorithms [37] (utilizing the fact that

strict convexity is retained upon addition with a convex function), and add for each block a strictly convex term for that block that goes to 0 as $\rho \to \infty$.

$$\hat{f}_\rho(\boldsymbol{\mathcal{R}}, \mathbf{A}_1, \mathbf{A}_2) = f'(\boldsymbol{\mathcal{R}}, \mathbf{A}_1, \mathbf{A}_2) + \frac{1}{\rho}(\|\boldsymbol{\mathcal{R}}\|_F^2 + \|\mathbf{A}_1\|_F^2 + \|\mathbf{A}_2\|_F^2) \tag{A.26}$$

Note that the block–partial Hessians of the added term is $\mathbf{I}/\rho$ which is positive definite and hence strictly convex. The block-gauss siedle (which coincides with ALS in this case) for $prox_f$ converges to a critical point [55](Th. 6.1.6.2,6.3) At the same time, for each $\hat{f}$, $\rho \to \infty$, a critical point in $\hat{f}$ converges to a critical point in f as $\hat{f}$ is continuous in ρ. Hence, we have an algorithm for finding a critical point of f. The complexity of finding a critical point of $\hat{f}$ is same as the ALASLAN for REGAL.