Modeling Relational Data with Graph Convolutional Networks

Michael Schlichtkrull[*]
University of Amsterdam
m.s.schlichtkrull@uva.nl

Thomas N. Kipf[*]
University of Amsterdam
t.n.kipf@uva.nl

Peter Bloem
VU Amsterdam
p.bloem@vu.nl

Rianne van den Berg
University of Amsterdam
r.vandenberg@uva.nl

Ivan Titov
University of Amsterdam
titov@uva.nl

Max Welling
University of Amsterdam, CIFAR[†]
m.welling@uva.nl

Abstract

Knowledge graphs enable a wide variety of applications, including question answering and information retrieval. Despite the great effort invested in their creation and maintenance, even the largest (e.g., Yago, DBPedia or Wikidata) remain incomplete. We introduce Relational Graph Convolutional Networks (R-GCNs) and apply them to two standard knowledge base completion tasks: Link prediction (recovery of missing facts, i.e. subject-predicate-object triples) and entity classification (recovery of missing entity attributes). R-GCNs are related to a recent class of neural networks operating on graphs, and are developed specifically to deal with the highly multi-relational data characteristic of realistic knowledge bases. We demonstrate the effectiveness of R-GCNs as a stand-alone model for entity classification. We further show that factorization models for link prediction such as DistMult can be significantly improved by enriching them with an encoder model to accumulate evidence over multiple inference steps in the relational graph, demonstrating a large improvement of 29.8% on FB15k-237 over a decoder-only baseline.

1 Introduction

Knowledge bases organize and store factual knowledge, enabling a multitude of applications including question answering (Yao and Van Durme 2014; Bao et al. 2014; Seyler, Yahya, and Berberich 2015; Hixon, Clark, and Hajishirzi 2015; Bordes et al. 2015; Dong et al. 2015) and information retrieval (Kotov and Zhai 2012; Dalton, Dietz, and Allan 2014; Xiong and Callan 2015b; 2015a). Even the largest knowledge bases (e.g. DBPedia, Wikidata or Yago), despite enormous effort invested in their maintenance, are incomplete, and the lack of coverage harms downstream applications. Predicting missing information in knowledge bases is the main focus of statistical relational learning (SRL).

Following previous work on SRL, we assume that knowledge bases store collections of triples of the form (subject, predicate, object). Consider, for example, the triple (*Mikhail Baryshnikov*, *educated_at*, *Vaganova Academy*), where we will refer to *Baryshnikov* and *Vaganova Academy* as entities and to *educated_at* as a relation. Additionally, we assume that entities are labeled with types (e.g., *Vaganova Academy* is marked as a *university*). It is convenient to represent knowledge bases as directed labeled multigraphs with entities corresponding to nodes and triples encoded by labeled edges (see Figure 1).

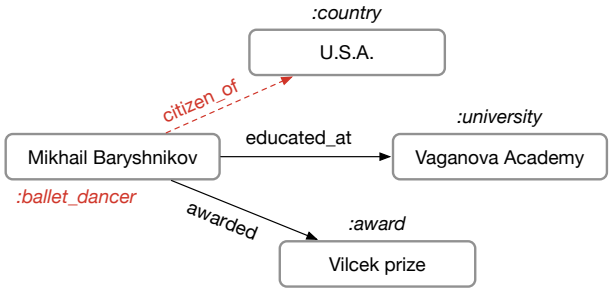


Figure 1: A knowledge base fragment: The nodes are entities, the edges are relations labeled with their types, the nodes are labeled with entity types (e.g., *university*). The edge and the node label shown in red are the missing information to be inferred.

We consider two fundamental SRL tasks: link prediction (recovery of missing triples) and entity classification (assigning types or categorical properties to entities). In both cases, many missing pieces of information can be expected to reside within the graph encoded through the neighborhood structure – i.e. knowing that *Mikhail Baryshnikov* was educated at the *Vaganova Academy* implies both that *Mikhail Baryshnikov* should have the label person, and that the triple (*Mikhail Baryshnikov*, *lived_in*, *Russia*) must belong to the knowledge graph. Following this intuition, we develop an encoder model for entities in the relational graph and apply it to both tasks.

Our entity classification model, similarly to Kipf and Welling (2017), uses softmax classifiers at each node in the graph. The classifiers take node representations supplied by a relational graph convolutional network (R-GCN) and predict the labels. The model, including R-GCN parameters, is learned by optimizing the cross-entropy loss.

Our link prediction model can be regarded as an autoencoder consisting of (1) an encoder: an R-GCN producing latent feature representations of entities, and (2) a decoder: a tensor factorization model exploiting these representations

[*]Equal contribution.
[†]Canadian Institute for Advanced Research

to predict labeled edges. Though in principle the decoder can rely on any type of factorization (or generally any scoring function), we use one of the simplest and most effective factorization methods: DistMult (Yang et al. 2014). We observe that our method achieves competitive results on standard benchmarks, outperforming, among other baselines, direct optimization of the factorization (i.e. vanilla DistMult). This improvement is especially large when we consider the more challenging FB15k-237 dataset (Toutanova and Chen 2015). This result demonstrates that explicit modeling of neighborhoods in R-GCNs is beneficial for recovering missing facts in knowledge bases.

Our main contributions are as follows. To the best of our knowledge, we are the first to show that the GCN framework can be applied to modeling relational data, specifically to link prediction and entity classification tasks. Secondly, we introduce techniques for parameter sharing and to enforce sparsity constraints, and use them to apply R-GCNs to multigraphs with large numbers of relations. Lastly, we show that the performance of factorization models, at the example of DistMult, can be significantly improved by enriching them with an encoder model that performs multiple steps of information propagation in the relational graph.

2 Neural relational modeling

We introduce the following notation: we denote directed and labeled multi-graphs as $G = (\mathcal{V}, \mathcal{E}, \mathcal{R})$ with nodes (entities) $v_i \in \mathcal{V}$ and labeled edges (relations) $(v_i, r, v_j) \in \mathcal{E}$, where $r \in \mathcal{R}$ is a relation type.[1]

2.1 Relational graph convolutional networks

Our model is primarily motivated as an extension of GCNs that operate on local graph neighborhoods (Duvenaud et al. 2015; Kipf and Welling 2017) to large-scale relational data. These and related methods such as graph neural networks (Scarselli et al. 2009) can be understood as special cases of a simple differentiable message-passing framework (Gilmer et al. 2017):

$$h_i^{(l+1)} = \sigma\left(\sum_{m \in \mathcal{M}_i} g_m(h_i^{(l)}, h_j^{(l)})\right), \qquad (1)$$

where $h_i^{(l)} \in \mathbb{R}^{d^{(l)}}$ is the hidden state of node v_i in the l-th layer of the neural network, with $d^{(l)}$ being the dimensionality of this layer's representations. Incoming messages of the form $g_m(\cdot,\cdot)$ are accumulated and passed through an element-wise activation function $\sigma(\cdot)$, such as the $\mathrm{ReLU}(\cdot) = \max(0, \cdot)$.[2] $\mathcal{M}_i$ denotes the set of incoming messages for node v_i and is often chosen to be identical to the set of incoming edges. $g_m(\cdot,\cdot)$ is typically chosen to be a (message-specific) neural network-like function or simply a linear transformation $g_m(h_i, h_j) = Wh_j$ with a weight matrix W such as in Kipf and Welling (2017).

[1] $\mathcal{R}$ contains relations both in canonical direction (e.g. *born_in*) and in inverse direction (e.g. *born_in_inv*).

[2] Note that this represents a simplification of the message passing neural network proposed in (Gilmer et al. 2017) that suffices to include the aforementioned models as special cases.

This type of transformation has been shown to be very effective at accumulating and encoding features from local, structured neighborhoods, and has led to significant improvements in areas such as graph classification (Duvenaud et al. 2015) and graph-based semi-supervised learning (Kipf and Welling 2017).

Motivated by these architectures, we define the following simple propagation model for calculating the forward-pass update of an entity or node denoted by v_i in a relational (directed and labeled) multi-graph:

$$h_i^{(l+1)} = \sigma\left(\sum_{r \in \mathcal{R}} \sum_{j \in \mathcal{N}_i^r} \frac{1}{c_{i,r}} W_r^{(l)} h_j^{(l)} + W_0^{(l)} h_i^{(l)}\right), \qquad (2)$$

where $\mathcal{N}_i^r$ denotes the set of neighbor indices of node i under relation $r \in \mathcal{R}$. $c_{i,r}$ is a problem-specific normalization constant that can either be learned or chosen in advance (such as $c_{i,r} = |\mathcal{N}_i^r|$).

Intuitively, (2) accumulates transformed feature vectors of neighboring nodes through a normalized sum. Different from regular GCNs, we introduce relation-specific transformations, i.e. depending on the type and direction of an edge. To ensure that the representation of a node at layer $l+1$ can also be informed by the corresponding representation at layer l, we add a single self-connection of a special relation type to each node in the data. Note that instead of simple linear message transformations, one could choose more flexible functions such as multi-layer neural networks (at the expense of computational efficiency). We leave this for future work.

A neural network layer update consists of evaluating (2) in parallel for every node in the graph. In practice, (2) can be implemented efficiently using sparse matrix multiplications to avoid explicit summation over neighborhoods. Multiple layers can be stacked to allow for dependencies across several relational steps. We refer to this graph encoder model as a relational graph convolutional network (R-GCN). The computation graph for a single node update in the R-GCN model is depicted in Figure 2.

2.2 Regularization

A central issue with applying (2) to highly multi-relational data is the rapid growth in number of parameters with the number of relations in the graph. In practice this can easily lead to overfitting on rare relations and to models of very large size.

To address this issue, we introduce two separate methods for regularizing the weights of R-GCN-layers: *basis*- and *block-diagonal*-decomposition. With the basis decomposition, each $W_r^{(l)}$ is defined as follows:

$$W_r^{(l)} = \sum_{b=1}^{B} a_{rb}^{(l)} V_b^{(l)}, \qquad (3)$$

i.e. as a linear combination of basis transformations $V_b^{(l)} \in \mathbb{R}^{d^{(l+1)} \times d^{(l)}}$ with coefficients $a_{rb}^{(l)}$ such that only the coefficients depend on r. In the block-diagonal decomposition, we

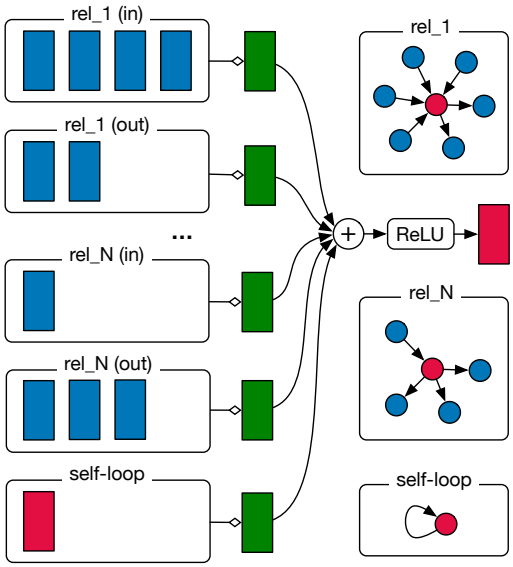


Figure 2: Diagram for computing the update of a single graph node/entity (red) in the R-GCN model. Activations (d-dimensional vectors) from neighboring nodes (dark blue) are gathered and then transformed for each relation type individually (for both in- and outgoing edges). The resulting representation (green) is accumulated in a (normalized) sum and passed through an activation function (such as the ReLU). This per-node update can be computed in parallel with shared parameters across the whole graph.

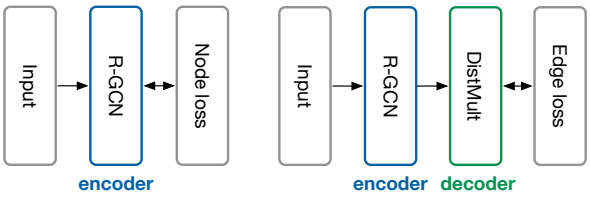
let each $W_r^{(l)}$ be defined through the direct sum over a set of low-dimensional matrices:

$$W_r^{(l)} = \bigoplus_{b=1}^{B} Q_{br}^{(l)}. \tag{4}$$

Thereby, $W_r^{(l)}$ are block-diagonal matrices: $\mathrm{diag}(Q_{1r}^{(l)}, \ldots, Q_{Br}^{(l)})$ with $Q_{br}^{(l)} \in \mathbb{R}^{(d^{(l+1)}/B)\times(d^{(l)}/B)}$.

The basis function decomposition (3) can be seen as a form of effective weight sharing between different relation types, while the block decomposition (4) can be seen as a sparsity constraint on the weight matrices for each relation type. The block decomposition structure encodes an intuition that latent features can be grouped into sets of variables which are more tightly coupled within groups than across groups. Both decompositions reduce the number of parameters needed to learn for highly multi-relational data (such as realistic knowledge bases). At the same time, we expect that the basis parameterization can alleviate overfitting on rare relations, as parameter updates are shared between both rare and more frequent relations.

The overall R-GCN model then takes the following form: We stack L layers as defined in (2) – the output of the previous layer being the input to the next layer. The input to the first layer can be chosen as a unique one-hot vector for each node in the graph if no other features are present. For the block representation, we map this one-hot vector to a dense representation through a single linear transformation. While



(a) Entity classification (b) Link prediction

Figure 3: (a) Depiction of an R-GCN model for entity classification with a per-node loss function. (b) Link prediction model with an R-GCN encoder (interspersed with fully-connected/dense layers) and a DistMult decoder that takes pairs of hidden node representations and produces a score for every (potential) edge in the graph. The loss is evaluated per edge.

we only consider such a featureless approach in this work, we note that it was shown in Kipf and Welling (2017) that it is possible for this class of models to make use of pre-defined feature vectors (e.g. a bag-of-words description of a document associated with a specific node).

3 Entity classification

For (semi-)supervised classification of nodes (entities), we simply stack R-GCN layers of the form (2), with a $\mathrm{softmax}(\cdot)$ activation (per node) on the output of the last layer. We minimize the following cross-entropy loss on all labeled nodes (while ignoring unlabeled nodes):

$$\mathcal{L} = -\sum_{i\in\mathcal{Y}}\sum_{k=1}^{K} t_{ik} \ln h_{ik}^{(L)}, \tag{5}$$

where $\mathcal{Y}$ is the set of node indices that have labels and $h_{ik}^{(L)}$ is the k-th entry of the network output for the i-th labeled node. t_{ik} denotes its respective ground truth label. In practice, we train the model using (full-batch) gradient descent techniques. A schematic depiction of our entity classification model is given in Figure 3a.

4 Link prediction

Link prediction deals with prediction of new facts (i.e. triples *(subject, relation, object)*). Formally, the knowledge base is represented by a directed, labeled graph $G = (\mathcal{V}, \mathcal{E}, \mathcal{R})$. Rather than the full set of edges $\mathcal{E}$, we are given only an incomplete subset $\hat{\mathcal{E}}$. The task is to assign scores $f(s, r, o)$ to possible edges (s, r, o) in order to determine how likely those edges are to belong to $\mathcal{E}$.

In order to tackle this problem, we introduce a graph auto-encoder model, comprised of an entity encoder and a scoring function (decoder). The encoder maps each entity $v_i \in \mathcal{V}$ to a real-valued vector $e_i \in \mathbb{R}^d$. The decoder reconstructs edges of the graph relying on the vertex representations; in other words, it scores *(subject, relation, object)*-triples through a function $s : \mathbb{R}^d \times \mathcal{R} \times \mathbb{R}^d \to \mathbb{R}$. Most existing approaches to link prediction (for example, tensor and neural factorization methods (Socher et al. 2013;

Lin et al. 2015; Toutanova et al. 2016; Yang et al. 2014; Trouillon et al. 2016)) can be interpreted under this framework. The crucial distinguishing characteristic of our work is the reliance on an encoder. Whereas most previous approaches use a single, real-valued vector e_i for every $v_i \in \mathcal{V}$ optimized directly in training, we compute representations through an R-GCN encoder with $e_i = h_i^{(L)}$, similar to the graph auto-encoder model introduced in Kipf and Welling (2016) for unlabeled undirected graphs. Our full link prediction model is schematically depicted in Figure 3b.

In our experiments, we use the DistMult factorization (Yang et al. 2014) as the scoring function, which is known to perform well on standard link prediction benchmarks when used on its own. In DistMult, every relation r is associated with a diagonal matrix $R_r \in \mathbb{R}^{d \times d}$ and a triple (s, r, o) is scored as

$$f(s, r, o) = e_s^T R_r e_o \,. \tag{6}$$

As in previous work on factorization (Yang et al. 2014; Trouillon et al. 2016), we train the model with negative sampling. For each observed example we sample ω negative ones. We sample by randomly corrupting either the subject or the object of each positive example. We optimize for cross-entropy loss to push the model to score observable triples higher than the negative ones:

$$\mathcal{L} = -\frac{1}{(1+\omega)|\hat{\mathcal{E}}|} \sum_{(s,r,o,y) \in \mathcal{T}} y \log l\big(f(s,r,o)\big) + (1-y) \log\big(1 - l\big(f(s,r,o)\big)\big)\,, \tag{7}$$

where $\mathcal{T}$ is the total set of real and corrupted triples, l is the logistic sigmoid function, and y is an indicator set to $y = 1$ for positive triples and $y = 0$ for negative ones.

5 Empirical evaluation

5.1 Entity classification experiments

Here, we consider the task of classifying entities in a knowledge base. In order to infer, for example, the type of an entity (e.g. person or company), a successful model needs to reason about the relations with other entities that this entity is involved in.

Datasets We evaluate our model on four datasets[3] in Resource Description Framework (RDF) format (Ristoski, de Vries, and Paulheim 2016): AIFB, MUTAG, BGS, and AM. Relations in these datasets need not necessarily encode directed subject-object relations, but are also used to encode the presence, or absence, of a specific feature for a given entity. In each dataset, the targets to be classified are properties of a group of entities represented as nodes. The exact statistics of the datasets can be found in Table 1. For a more detailed description of the datasets the reader is referred to Ristoski, de Vries, and Paulheim (2016). We remove relations that were used to create entity labels: *employs* and *affiliation* for AIFB, *isMutagenic* for MUTAG, *hasLithogenesis* for BGS, and *objectCategory* and *material* for AM.

[3] http://dws.informatik.uni-mannheim.de/en/research/a-collection-of-benchmark-datasets-for-ml

Dataset	AIFB	MUTAG	BGS	AM
Entities	8,285	23,644	333,845	1,666,764
Relations	45	23	103	133
Edges	29,043	74,227	916,199	5,988,321
Labeled	176	340	146	1,000
Classes	4	2	2	11

Table 1: Number of entities, relations, edges and classes along with the number of labeled entities for each of the datasets. *Labeled* denotes the subset of entities that have labels and that are to be classified.

Baselines As a baseline for our experiments, we compare against recent state-of-the-art classification results from RDF2Vec embeddings (Ristoski and Paulheim 2016), Weisfeiler-Lehman kernels (WL) (Shervashidze et al. 2011; de Vries and de Rooij 2015), and hand-designed feature extractors (Feat) (Paulheim and Fümkranz 2012). Feat assembles a feature vector from the in- and out-degree (per relation) of every labeled entity. RDF2Vec extracts walks on labeled graphs which are then processed using the Skipgram (Mikolov et al. 2013) model to generate entity embeddings, used for subsequent classification. See Ristoski and Paulheim (2016) for an in-depth description and discussion of these baseline approaches. All entity classification experiments were run on CPU nodes with 64GB of memory.

Results All results in Table 2 are reported on the train/test benchmark splits from Ristoski, de Vries, and Paulheim (2016). We further set aside 20% of the training set as a validation set for hyperparameter tuning. For R-GCN, we report performance of a 2-layer model with 16 hidden units (10 for AM), basis function decomposition (Eq. 3), and trained with Adam (Kingma and Ba 2014) for 50 epochs using a learning rate of 0.01. The normalization constant is chosen as $c_{i,r} = |\mathcal{N}_i^r|$. Further details on (baseline) models and hyperparameter choices are provided in the supplementary material.

Model	AIFB	MUTAG	BGS	AM
Feat	55.55	77.94	72.41	66.66
WL	80.55	**80.88**	86.20	87.37
RDF2Vec	88.88	67.20	**87.24**	88.33
R-GCN	**95.83**	73.23	83.10	**89.29**

Table 2: Entity classification results in accuracy (averaged over 10 runs) for a feature-based baseline (see main text for details), WL (Shervashidze et al. 2011; de Vries and de Rooij 2015), RDF2Vec (Ristoski and Paulheim 2016), and R-GCN (this work). Test performance is reported on the train/test set splits provided by Ristoski, de Vries, and Paulheim (2016).

Our model achieves state-of-the-art results on AIFB and AM. To explain the gap in performance on MUTAG and BGS it is important to understand the nature of these

datasets. MUTAG is a dataset of molecular graphs, which was later converted to RDF format, where relations either indicate atomic bonds or merely the presence of a certain feature. BGS is a dataset of rock types with hierarchical feature descriptions which was similarly converted to RDF format, where relations encode the presence of a certain feature or feature hierarchy. Labeled entities in MUTAG and BGS are only connected via high-degree hub nodes that encode a certain feature.

We conjecture that the fixed choice of normalization constant for the aggregation of messages from neighboring nodes is partly to blame for this behavior, which can be particularly problematic for nodes of high degree. A potential way to overcome this limitation is to introduce an attention mechanism, i.e. to replace the normalization constant $1/c_{i,r}$ with data-dependent attention weights $a_{ij,r}$, where $\sum_{j,r} a_{ij,r} = 1$. We expect this to be a promising avenue for future research.

5.2 Link prediction experiments

As shown in the previous section, R-GCNs serve as an effective encoder for relational data. We now combine our encoder model with a scoring function (which we will refer to as a decoder, see Figure 3b) to score candidate triples for link prediction in knowledge bases.

Datasets Link prediction algorithms are commonly evaluated on FB15k, a subset of the relational database Freebase, and WN18, a subset of WordNet containing lexical relations between words. In Toutanova and Chen (2015), a serious flaw was observed in both datasets: The presence of inverse triplet pairs $t = (e_1, r, e_2)$ and $t' = (e_2, r^{-1}, e_1)$ with t in the training set and t' in the test set. This reduces a large part of the prediction task to memorization of affected triplet pairs. A simple baseline LinkFeat employing a linear classifier on top of sparse feature vectors of observed training relations was shown to outperform existing systems by a large margin. To address this issue, Toutanova and Chen proposed a reduced dataset FB15k-237 with all such inverse triplet pairs removed. We therefore choose FB15k-237 as our primary evaluation dataset. Since FB15k and WN18 are still widely used, we also include results on these datasets using the splits introduced by Bordes et al. (2013).

Dataset	WN18	FB15K	FB15k-237
Entities	40,943	14,951	14,541
Relations	18	1,345	237
Train edges	141,442	483,142	272,115
Val. edges	5,000	50,000	17,535
Test edges	5,000	59,071	20,466

Table 3: Number of entities and relation types along with the number of edges per split for the three datasets.

Baselines A common baseline for both experiments is direct optimization of *DistMult* (Yang et al. 2014). This factorization strategy is known to perform well on standard

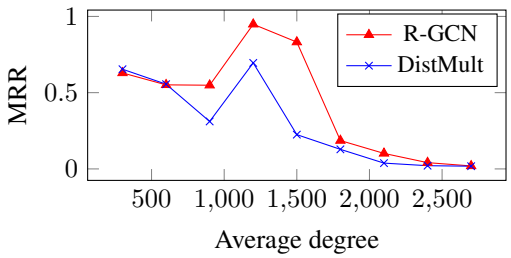


Figure 4: Mean reciprocal rank (MRR) for R-GCN and DistMult on the FB15k validation data as a function of the node degree (average of subject and object).

datasets, and furthermore corresponds to a version of our model with fixed entity embeddings in place of the R-GCN encoder as described in Section 4. As a second baseline, we add the simple neighbor-based LinkFeat algorithm proposed in Toutanova and Chen (2015).

We further compare to ComplEx (Trouillon et al. 2016) and HolE (Nickel, Rosasco, and Poggio 2015), two state-of-the-art link prediction models for FB15k and WN18. ComplEx facilitates modeling of asymmetric relations by generalizing DistMult to the complex domain, while HolE replaces the vector-matrix product with circular correlation. Finally, we include comparisons with two classic algorithms – CP (Hitchcock 1927) and TransE (Bordes et al. 2013).

Results We provide results using two commonly used evaluation metrics: *mean reciprocal rank* (MRR) and *Hits at* n (H@n). Following Bordes et al. (2013), both metrics can be computed in a raw and a filtered setting. We report both filtered and raw MRR (with filtered MRR typically considered more reliable), and filtered Hits at 1, 3, and 10.

We evaluate hyperparameter choices on the respective validation splits. We found a normalization constant defined as $c_{i,r} = c_i = \sum_r |\mathcal{N}_i^r|$ — in other words, applied across relation types – to work best. For FB15k and WN18, we report results using basis decomposition (Eq. 3) with two basis functions, and a single encoding layer with 200-dimensional embeddings. For FB15k-237, we found block decomposition (Eq. 4) to perform best, using two layers with block dimension 5×5 and 500-dimensional embeddings. We regularize the encoder through edge dropout applied before normalization, with dropout rate 0.2 for self-loops and 0.4 for other edges. Using edge droupout makes our training objective similar to that of denoising autoencoders (Vincent et al. 2008). We apply $l2$ regularization to the decoder with a penalty of 0.01.

We use the Adam optimizer (Kingma and Ba 2014) with a learning rate of 0.01. For the baseline and the other factorizations, we found the parameters from Trouillon et al. (2016) – apart from the dimensionality on FB15k-237 – to work best, though to make the systems comparable we maintain the same number of negative samples (i.e. $\omega = 1$). We use full-batch optimization for both the baselines and our model.

On FB15k, local context in the form of inverse relations

	FB15k					WN18				
	MRR		Hits @			MRR		Hits @		
Model	Raw	Filtered	1	3	10	Raw	Filtered	1	3	10
LinkFeat		0.779			0.804		0.938			0.939
DistMult	0.248	0.634	0.522	0.718	0.814	0.526	0.813	0.701	0.921	0.943
R-GCN	0.251	0.651	0.541	0.736	0.825	0.553	0.814	0.686	0.928	0.955
R-GCN+	**0.262**	**0.696**	**0.601**	**0.760**	**0.842**	0.561	0.819	0.697	0.929	**0.964**
CP*	0.152	0.326	0.219	0.376	0.532	0.075	0.058	0.049	0.080	0.125
TransE*	0.221	0.380	0.231	0.472	0.641	0.335	0.454	0.089	0.823	0.934
HolE**	0.232	0.524	0.402	0.613	0.739	**0.616**	0.938	0.930	**0.945**	0.949
ComplEx*	0.242	0.692	0.599	0.759	0.840	0.587	**0.941**	**0.936**	**0.945**	0.947

Table 4: Results on the the Freebase and WordNet datasets. Results marked (*) taken from Trouillon et al. (2016). Results marks (**) taken from Nickel, Rosasco, and Poggio (2015). R-GCN+ denotes an ensemble between R-GCN and DistMult – see main text for details.

is expected to dominate the performance of the factorizations, contrasting with the design of the R-GCN model. To better understand the difference, we plot in Figure 4 the FB15k performance of the best R-GCN model and the baseline (DistMult) as functions of degree of nodes corresponding to entities in the considered triple (namely, the average of degrees for the subject and object entities). It can be seen that our model performs better for nodes with high degree where contextual information is abundant. The observation that the two models are complementary suggests combining the strengths of both into a single model, which we refer to as R-GCN+. On FB15k and WN18 where local and long-distance information can both provide strong solutions, we expect R-GCN+ to outperform each individual model. On FB15k-237 where local information is less salient, we do not expect the combination model to outperform a pure R-GCN model significantly. To test this, we evaluate an ensemble (R-GCN+) with a trained R-GCN model and a separately trained DistMult factorization model: $f(s,r,t)_{\text{R-GCN+}} = \alpha f(s,r,t)_{\text{R-GCN}} + (1-\alpha) f(s,r,t)_{\text{DistMult}}$, with $\alpha = 0.4$ selected on FB15k development data.

In Table 4, we evaluate the R-GCN model and the combination model (R-GCN+) on FB15k and WN18.

On the FB15k and WN18 datasets, R-GCN and R-GCN+ both outperform the DistMult baseline, but like all other systems underperform on these two datasets compared to the LinkFeat algorithm. The strong result from this baseline highlights the contribution of inverse relation pairs to high-performance solutions on these datasets. Interestingly, R-GCN+ yields better performance than ComplEx for FB15k, even though the R-GCN decoder (DistMult) does not explicitly model asymmetry in relations, as opposed to ComplEx.

This suggests that combining the R-GCN encoder with the ComplEx scoring function (decoder) may be a promising direction for future work. The choice of scoring function is orthogonal to the choice of encoder; in principle, any scoring function or factorization model could be incorporated as a decoder in our auto-encoder framework.

In Table 5, we show results for FB15k-237 where (as previously discussed) inverse relation pairs have been removed and the LinkFeat baseline fails to generalize[4]. Here, our R-GCN model outperforms the DistMult baseline by a large margin of 29.8%, highlighting the importance of a separate encoder model. As expected from our earlier analysis, R-GCN and R-GCN+ show similar performance on this dataset. The R-GCN model further compares favorably against other factorization methods, despite relying on a DistMult decoder which shows comparatively weak performance when used without an encoder.

	MRR		Hits @		
Model	Raw	Filtered	1	3	10
LinkFeat		0.063			0.079
DistMult	0.100	0.191	0.106	0.207	0.376
R-GCN	**0.158**	0.248	**0.153**	0.258	0.414
R-GCN+	0.156	**0.249**	0.151	**0.264**	**0.417**
CP	0.080	0.182	0.101	0.197	0.357
TransE	0.144	0.233	0.147	0.263	0.398
HolE	0.124	0.222	0.133	0.253	0.391
ComplEx	0.109	0.201	0.112	0.213	0.388

Table 5: Results on FB15k-237, a reduced version of FB15k with problematic inverse relation pairs removed. CP, TransE, and ComplEx were evaluated using the code published for Trouillon et al. (2016), while HolE was evaluated using the code published for Nickel, Rosasco, and Poggio (2015).

[4] Our numbers are not directly comparable to those reported in Toutanova and Chen (2015), as they use pruning both for training and testing (see their sections 3.3.1 and 4.2). Since their pruning schema is not fully specified (e.g., values of the relation-specific parameter t are not given) and the code is not available, it is not possible to replicate their set-up.

6 Related Work

6.1 Relational modeling

Our encoder-decoder approach to link prediction relies on DistMult (Yang et al. 2014) in the decoder, a special and simpler case of the RESCAL factorization (Nickel, Tresp, and Kriegel 2011), more effective than the original RESCAL in the context of multi-relational knowledge bases. Numerous alternative factorizations have been proposed and studied in the context of SRL, including both (bi-)linear and non-linear ones (e.g., (Bordes et al. 2013; Socher et al. 2013; Chang et al. 2014; Nickel, Rosasco, and Poggio 2015; Trouillon et al. 2016)). Many of these approaches can be regarded as modifications or special cases of classic tensor decomposition methods such as CP or Tucker; for a comprehensive overview of tensor decomposition literature we refer the reader to Kolda and Bader (2009).

Incorporation of paths between entities in knowledge bases has recently received considerable attention. We can roughly classify previous work into (1) methods creating auxiliary triples, which are then added to the learning objective of a factorization model (Guu, Miller, and Liang 2015; Garcia-Duran, Bordes, and Usunier 2015); (2) approaches using paths (or walks) as features when predicting edges (Lin et al. 2015); or (3) doing both at the same time (Neelakantan, Roth, and McCallum 2015; Toutanova et al. 2016). The first direction is largely orthogonal to ours, as we would also expect improvements from adding similar terms to our loss (in other words, extending our decoder). The second research line is more comparable; R-GCNs provide a computationally cheaper alternative to these path-based models. Direct comparison is somewhat complicated as path-based methods used different datasets (e.g., sub-sampled sets of walks from a knowledge base).

6.2 Neural networks on graphs

Our R-GCN encoder model is closely related to a number of works in the area of neural networks on graphs. It is primarily motivated as an adaption of previous work on GCNs (Bruna et al. 2014; Duvenaud et al. 2015; Defferrard, Bresson, and Vandergheynst 2016; Kipf and Welling 2017) for large-scale and highly multi-relational data, characteristic of realistic knowledge bases.

Early work in this area includes the graph neural network by Scarselli et al. (2009). A number of extensions to the original graph neural network have been proposed, most notably (Li et al. 2016) and (Pham et al. 2017), both of which utilize gating mechanisms to facilitate optimization.

R-GCNs can further be seen as a sub-class of message passing neural networks (Gilmer et al. 2017), which encompass a number of previous neural models for graphs, including GCNs, under a differentiable message passing interpretation.

7 Conclusions

We have introduced relational graph convolutional networks (R-GCNs) and demonstrated their effectiveness in the context of two standard statistical relation modeling problems: link prediction and entity classification. For the entity classification problem, we have demonstrated that the R-GCN model can act as a competitive, end-to-end trainable graph-based encoder. For link prediction, the R-GCN model with DistMult factorization as the decoding component outperformed direct optimization of the factorization model, and achieved competitive results on standard link prediction benchmarks. Enriching the factorization model with an R-GCN encoder proved especially valuable for the challenging FB15k-237 dataset, yielding a 29.8% improvement over the decoder-only baseline.

There are several ways in which our work could be extended. For example, the graph autoencoder model could be considered in combination with other factorization models, such as ComplEx (Trouillon et al. 2016), which can be better suited for modeling asymmetric relations. It is also straightforward to integrate entity features in R-GCNs, which would be beneficial both for link prediction and entity classification problems. To address the scalability of our method, it would be worthwhile to explore subsampling techniques, such as in Hamilton, Ying, and Leskovec (2017). Lastly, it would be promising to replace the current form of summation over neighboring nodes and relation types with a data-dependent attention mechanism. Beyond modeling knowledge bases, R-GCNs can be generalized to other applications where relation factorization models have been shown effective (e.g. relation extraction).

Acknowledgements

We would like to thank Diego Marcheggiani, Ethan Fetaya, and Christos Louizos for helpful discussions and comments. This project is supported by the European Research Council (ERC StG BroadSem 678254), the SAP Innovation Center Network and the Dutch National Science Foundation (NWO VIDI 639.022.518).

References

Bao, J.; Duan, N.; Zhou, M.; and Zhao, T. 2014. Knowledge-based question answering as machine translation. In *ACL*.

Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; and Yakhnenko, O. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*.

Bordes, A.; Usunier, N.; Chopra, S.; and Weston, J. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*.

Bruna, J.; Zaremba, W.; Szlam, A.; and LeCun, Y. 2014. Spectral networks and locally connected networks on graphs. In *ICLR*.

Chang, K.-W.; tau Yih, W.; Yang, B.; and Meek, C. 2014. Typed Tensor Decomposition of Knowledge Bases for Relation Extraction. In *EMNLP*.

Dalton, J.; Dietz, L.; and Allan, J. 2014. Entity query feature expansion using knowledge base links. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, 365–374.

de Vries, G. K. D., and de Rooij, S. 2015. Substructure counting graph kernels for machine learning from rdf data.

Web Semantics: Science, Services and Agents on the World Wide Web 35:71–84.

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*.

Dong, L.; Wei, F.; Zhou, M.; and Xu, K. 2015. Question answering over freebase with multi-column convolutional neural networks. In *ACL*.

Duvenaud, D. K.; Maclaurin, D.; Iparraguirre, J.; Bombarell, R.; Hirzel, T.; Aspuru-Guzik, A.; and Adams, R. P. 2015. Convolutional networks on graphs for learning molecular fingerprints. In *NIPS*.

Garcia-Duran, A.; Bordes, A.; and Usunier, N. 2015. Composing relationships with translations. Technical Report, CNRS, Heudiasyc.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*.

Guu, K.; Miller, J.; and Liang, P. 2015. Traversing knowledge graphs in vector space. *arXiv preprint arXiv:1506.01094*.

Hamilton, W. L.; Ying, R.; and Leskovec, J. 2017. Inductive representation learning on large graphs. *arXiv preprint arXiv:1706.02216*.

Hitchcock, F. L. 1927. The expression of a tensor or a polyadic as a sum of products. *Studies in Applied Mathematics* 6(1-4):164–189.

Hixon, B.; Clark, P.; and Hajishirzi, H. 2015. Learning knowledge graphs for question answering through conversational dialog. In *Proceedings of NAACL HLT*, 851–861.

Kingma, D., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kipf, T. N., and Welling, M. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*.

Kipf, T. N., and Welling, M. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

Kolda, T. G., and Bader, B. W. 2009. Tensor decompositions and applications. *SIAM review* 51(3):455–500.

Kotov, A., and Zhai, C. 2012. Tapping into knowledge base for concept feedback: leveraging conceptnet to improve search results for difficult queries. In *WSDM*.

Li, Y.; Tarlow, D.; Brockschmidt, M.; and Zemel, R. 2016. Gated graph sequence neural networks. In *ICLR*.

Lin, Y.; Liu, Z.; Luan, H.; Sun, M.; Rao, S.; and Liu, S. 2015. Modeling relation paths for representation learning of knowledge bases. *arXiv preprint arXiv:1506.00379*.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

Neelakantan, A.; Roth, B.; and McCallum, A. 2015. Compositional vector space models for knowledge base completion. *arXiv preprint arXiv:1504.06662*.

Nickel, M.; Rosasco, L.; and Poggio, T. 2015. Holographic embeddings of knowledge graphs. *arXiv preprint arXiv:1510.04935*.

Nickel, M.; Tresp, V.; and Kriegel, H.-P. 2011. A three-way model for collective learning on multi-relational data. In *ICML*.

Paulheim, H., and Fümkranz, J. 2012. Unsupervised generation of data mining features from linked open data. In *Proceedings of the 2nd international conference on web intelligence, mining and semantics*, 31.

Pham, T.; Tran, T.; Phung, D.; and Venkatesh, S. 2017. Column networks for collective classification. In *AAAI*.

Ristoski, P., and Paulheim, H. 2016. Rdf2vec: Rdf graph embeddings for data mining. In *International Semantic Web Conference*, 498–514. Springer.

Ristoski, P.; de Vries, G. K. D.; and Paulheim, H. 2016. A collection of benchmark datasets for systematic evaluations of machine learning on the semantic web. In *International Semantic Web Conference*, 186–194. Springer.

Scarselli, F.; Gori, M.; Tsoi, A. C.; Hagenbuchner, M.; and Monfardini, G. 2009. The graph neural network model. *IEEE Transactions on Neural Networks* 20(1):61–80.

Seyler, D.; Yahya, M.; and Berberich, K. 2015. Generating quiz questions from knowledge graphs. In *Proceedings of the 24th International Conference on World Wide Web*.

Shervashidze, N.; Schweitzer, P.; Leeuwen, E. J. v.; Mehlhorn, K.; and Borgwardt, K. M. 2011. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research* 12(Sep):2539–2561.

Socher, R.; Chen, D.; Manning, C. D.; and Ng, A. 2013. Reasoning with neural tensor networks for knowledge base completion. In *NIPS*.

Toutanova, K., and Chen, D. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, 57–66.

Toutanova, K.; Lin, V.; Yih, W.-t.; Poon, H.; and Quirk, C. 2016. Compositional learning of embeddings for relation paths in knowledge base and text. In *ACL*.

Trouillon, T.; Welbl, J.; Riedel, S.; Gaussier, E.; and Bouchard, G. 2016. Complex embeddings for simple link prediction. In *ICML*.

Vincent, P.; Larochelle, H.; Bengio, Y.; and Manzagol, P.-A. 2008. Extracting and composing robust features with denoising autoencoders. In *ICML*.

Xiong, C., and Callan, J. 2015a. Esdrank: Connecting query and documents through external semi-structured data. In *CIKM*.

Xiong, C., and Callan, J. 2015b. Query expansion with freebase. In *Proceedings of the 2015 International Conference on The Theory of Information Retrieval*, 111–120.

Yang, B.; Yih, W.-t.; He, X.; Gao, J.; and Deng, L. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*.

Yao, X., and Van Durme, B. 2014. Information extraction over structured data: Question answering with freebase. In *ACL*.

Further experimental details on entity classification

For the entity classification benchmarks described in our paper, the evaluation process differs subtly between publications. To eliminate these differences, we repeated the baselines in a uniform manner, using the canonical test/train split from (Ristoski, de Vries, and Paulheim 2016). We performed hyperparameter optimization on only the training set, running a single evaluation on the test set after hyperparameters were chosen for each baseline. This explains why the numbers we report differ slightly from those in the original publications (where cross-validation accuracy was reported).

For WL, we use the *tree* variant of the Weisfeiler-Lehman subtree kernel from the Mustard library.[5] For RDF2Vec, we use an implementation provided by the authors of (Ristoski and Paulheim 2016) which builds on Mustard. In both cases, we extract explicit feature vectors for the instance nodes, which are classified by a linear SVM.

For the MUTAG task, our preprocessing differs from that used in (de Vries and de Rooij 2015; Ristoski and Paulheim 2016) where for a given target relation (s, r, o) all triples connecting s to o are removed. Since o is a boolean value in the MUTAG data, one can infer the label after processing from other boolean relations that are still present. This issue is now mentioned in the Mustard documentation. In our preprocessing, we remove only the specific triples encoding the target relation.

Hyperparameters for baselines are chosen according to the best model performance in (Ristoski and Paulheim 2016), i.e. WL: 2 (tree depth), 3 (number of iterations); RDF2Vec: 2 (WL tree depth), 4 (WL iterations), 500 (embedding size), 5 (window size), 10 (SkipGram iterations), 25 (number of negative samples). We optimize the SVM regularization constant $C \in \{0.001, 0.01, 0.1, 1, 10, 100, 1000\}$ based on performance on a 80/20 train/validation split (of the original training set).

For R-GCN, we choose an $l2$ penalty on first layer weights $C_{l2} \in \{0, 5 \cdot 10^{-4}\}$ and the number of basis functions $B \in \{0, 10, 20, 30, 40\}$ based on validation set performance, where $B = 0$ refers to using no basis function decomposition. Using the block decomposition did not improve results. Otherwise, hyperparameters are chosen as follows: 50 (number of epochs), 16 (number of hidden units), and $c_{i,r} = |\mathcal{N}_i^r|$ (normalization constant). We do not use dropout. For AM, we use a reduced number of 10 hidden units for R-GCN to reduce the memory footprint.

Results with standard error (omitted in main paper due to spatial constraints) are summarized in Table 7. All entity classification experiments were run on CPU nodes with 64GB of memory.

R-GCN setting	AIFB	MUTAG	BGS	AM
$l2$ penalty	0	$5 \cdot 10^{-4}$	$5 \cdot 10^{-4}$	$5 \cdot 10^{-4}$
# basis functions	0	30	40	40
# hidden units	16	16	16	10

Table 6: Best hyperparameter choices based on validation set performance for 2-layer R-GCN model.

Model	AIFB	MUTAG	BGS	AM
Feat	55.55 ± 0.00	77.94 ± 0.00	72.41 ± 0.00	66.66 ± 0.00
WL	80.55 ± 0.00	$\mathbf{80.88} \pm 0.00$	86.20 ± 0.00	87.37 ± 0.00
RDF2Vec	88.88 ± 0.00	67.20 ± 1.24	$\mathbf{87.24} \pm 0.89$	88.33 ± 0.61
R-GCN (Ours)	$\mathbf{95.83} \pm 0.62$	73.23 ± 0.48	83.10 ± 0.80	$\mathbf{89.29} \pm 0.35$

Table 7: Entity classification results in accuracy (average and standard error over 10 runs) for a feature-based baseline (see main text for details), WL (Shervashidze et al. 2011; de Vries and de Rooij 2015), RDF2Vec (Ristoski and Paulheim 2016), and R-GCN (this work). Test performance is reported on the train/test set splits provided by (Ristoski, de Vries, and Paulheim 2016).

[5] https://github.com/Data2Semantics/mustard