Modeling Relational Data with Graph Convolutional Networks

Michael Schlichtkrull[1], Thomas N. Kipf[1(✉)], Peter Bloem[2], Rianne van den Berg[1], Ivan Titov[1,3], and Max Welling[1,4]

[1] University of Amsterdam, Amsterdam, The Netherlands
{m.s.schlichtkrull,t.n.kipf,r.vandenberg,titov,m.welling}@uva.nl
[2] Vrije Universiteit Amsterdam, Amsterdam, The Netherlands
p.bloem@vu.nl
[3] University of Edinburgh, Edinburgh, UK
[4] Canadian Institute for Advanced Research, Toronto, Canada

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 handle 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 through the use of an R-GCN encoder model to accumulate evidence over multiple inference steps in the 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 [1–6] and information retrieval [7–10]. 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).

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. Following

M. Schlichtkrull and T. N. Kipf—Equal contribution.

© Springer International Publishing AG, part of Springer Nature 2018
A. Gangemi et al. (Eds.): ESWC 2018, LNCS 10843, pp. 593–607, 2018.
https://doi.org/10.1007/978-3-319-93417-4_38

this intuition, we develop an encoder model for entities in the relational graph and apply it to both tasks.

Our entity classification model 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 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 [11]. We observe that our method achieves significant improvements on the challenging FB15k-237 dataset [12], as well as competitive performance on FB15k and WN18. Among other baselines, our model outperforms direct optimization of the factorization (i.e. vanilla DistMult). 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 [13,14] to large-scale relational data. These and related methods such as graph neural networks [15] can be understood as special cases of a simple differentiable message-passing framework [16]:

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

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

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 [14]. 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 [13] and graph-based semi-supervised learning [14].

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:

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

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 $c_{i,r} = |\mathcal{N}_i^r|$).

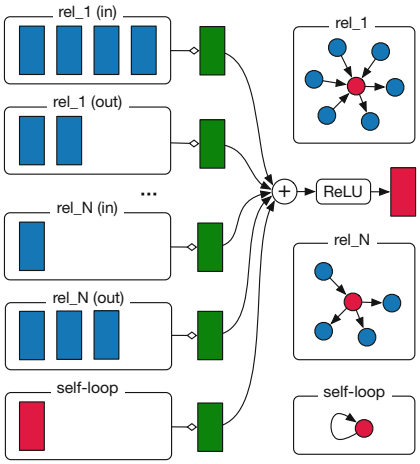
Intuitively, (2) accumulates transformed feature vectors of neighboring nodes through a normalized sum. Choosing linear transformations of the form Wh_j that only depend on the neighboring node has crucial computational benefits: (1) we do not need to store intermediate edge-based representations which could require a significant amount of memory, and (2) it allows us to implement Eq. 2 in vectorized form using efficient sparse-dense $\mathcal{O}(|\mathcal{E}|)$ matrix multiplications, similar to [14]. 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.

A neural network layer update consists of evaluating (2) in parallel for every node in the graph. 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 Fig. 1.

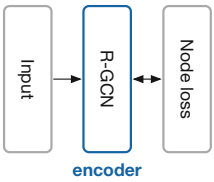
2.2 Regularization

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

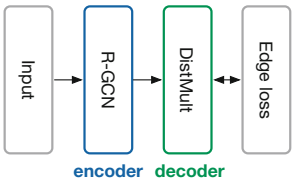
[2] Note that this represents a simplification of the message passing neural network proposed in [16] that suffices to include the aforementioned models as special cases.



(a) Single R-GCN layer



(b) Entity classification model



(c) Link prediction model

Fig. 1. 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. (b) Depiction of an R-GCN model for entity classification with a per-node loss function. (c) Link prediction model with an R-GCN encoder (interspersed with fully-connected/dense layers) and a DistMult decoder. (Color figure online)

large size. Two intuitive strategies to address such issues is to share parameters between weight matrices, and to enforce sparsity in weight matrices so as to limit the total number of parameters.

Corresponding to these two strategies, 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:

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

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 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:

$$\text{diag}(Q_{1r}^{(l)}, \ldots, Q_{Br}^{(l)}) \quad \text{with} \quad Q_{br}^{(l)} \in \mathbb{R}^{(d^{(l+1)}/B) \times (d^{(l)}/B)}. \tag{5}$$

Note that for $B = d$, each Q has dimension 1 and W_r becomes a diagonal matrix. The block-diagonal decomposition is as such a generalization of the diagonal sparsity constraint used in the decoder in e.g. DistMult [11].

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).

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 in this work we only consider the featureless approach, we note that GCN-type models can incorporate predefined feature vectors [14].

3 Entity Classification

For (semi-)supervised classification of nodes (entities), we simply stack R-GCN layers of the form (2), with a 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):

$$\mathcal{L} = -\sum_{i \in \mathcal{Y}} \sum_{k=1}^{K} t_{ik} \ln h_{ik}^{(L)}, \tag{6}$$

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 the model is given in Fig. 1b.

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 (see Fig. 1c), 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 $s : \mathbb{R}^d \times \mathcal{R} \times \mathbb{R}^d \rightarrow \mathbb{R}$. Most existing approaches to link prediction (for example, tensor and neural factorization methods [11,17–20]) 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 [21] for unlabeled undirected graphs.

In our experiments, we use the DistMult factorization [11] 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

$$f(s, r, o) = e_s^T R_r e_o. \tag{7}$$

As in previous work on factorization [11,20], 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:

$$\mathcal{L} = -\frac{1}{(1+\omega)|\hat{\mathcal{E}}|} \sum_{(s,r,o,y)\in\mathcal{T}} y \log l\big(f(s,r,o)\big) + (1-y)\log\big(1 - l\big(f(s,r,o)\big)\big), \tag{8}$$

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 [22]: AIFB, MUTAG, BGS, and AM. Relations in

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

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 [22]. 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.

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 [22]. 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).

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.

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

Baselines. As a baseline for our experiments, we compare against recent state-of-the-art classification results from RDF2Vec embeddings [23], Weisfeiler-Lehman kernels (WL) [24,25], and hand-designed feature extractors (Feat) [26]. 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 [27] model to generate entity embeddings, used for subsequent classification. See [23] for an in-depth description and discussion of these baseline approaches. All entity classification experiments were run on CPU nodes with 64 GB of memory.

For WL, we use the *tree* variant of the Weisfeiler-Lehman subtree kernel from the Mustard library.[4] For RDF2Vec, we use an implementation provided by the authors of [23] which builds on Mustard. In both cases, we extract explicit feature vectors for the instance nodes, which are classified by a linear SVM. For

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

the MUTAG task, our preprocessing differs from that used in [23,25] 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.

Results. All results in Table 2 are reported on the train/test benchmark splits from [22]. 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 [28] for 50 epochs using a learning rate of 0.01. The normalization constant is chosen as $c_{i,r} = |\mathcal{N}_i^r|$.

Hyperparameters for baselines are chosen according to the best model performance in [23], 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 no basis decomposition. 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. All entity classification experiments were run on CPU nodes with 64 GB of memory.

Table 2. Entity classification results in accuracy (average and standard error over 10 runs) for a feature-based baseline (see main text for details), WL [24,25], RDF2Vec [23], and R-GCN (this work). Test performance is reported on the train/test set splits provided by [22].

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	**80.88** ± 0.00	86.20 ± 0.00	87.37 ± 0.00
RDF2Vec	88.88 ± 0.00	67.20 ± 1.24	**87.24** ± 0.89	88.33 ± 0.61
R-GCN (Ours)	**95.83** ± 0.62	73.23 ± 0.48	83.10 ± 0.80	**89.29** ± 0.35

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 potentially promising way to overcome this limitation in future work 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 $\sum_{j,r} a_{ij,r} = 1$.

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 refer to as a decoder, see Fig. 1c) 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. In [12], 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, and a simple baseline LinkFeat employing a linear classifier and features of observed training relations was shown to outperform existing systems by a large margin. 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 in [29] (Table 3).

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

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

Baselines. A common baseline for both experiments is direct optimization of *DistMult* [11]. This factorization strategy is known to perform well on standard datasets, and furthermore corresponds to a version of our model with fixed entity embeddings in place of the R-GCN encoder as described in Sect. 4. As a second baseline, we add the simple neighbor-based LinkFeat algorithm proposed in [12].

We further compare to ComplEx [20] and HolE [30], 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 [31] and TransE [29].

Table 4. 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 [20], while HolE was evaluated using the code published for [30]. R-GCN+ denotes an ensemble between R-GCN and DistMult.

	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

Results. We provide results using two commonly used evaluation metrics: *mean reciprocal rank* (MRR) and *Hits at n* (H@n). Following [29], both metrics can be computed in a raw and a filtered setting. We report filtered and raw MRR, and filtered Hits at 1, 3, and 10.

We evaluate hyperparameter choices on the respective validation splits. We found a normalization constant defined as $c_{i,r} = c_i = \sum_r |\mathcal{N}_i^r|$, i.e. 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 via edge dropout applied before normalization, with dropout rate 0.2 for self-loops and 0.4 for other edges. We apply $l2$ regularization to the decoder with a penalty of 0.01.

We use the Adam optimizer [28] with a learning rate of 0.01. For the baseline and the other factorizations, we found the parameters from [20] – 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 is expected to dominate the performance of the factorizations, contrasting with the design of the R-GCN model. Preliminary experiments revealed that R-GCN still improved performance on high-degree vertices, where contextual knowledge is abundant. Since the two models for this dataset appear complementary, we attempt to combine the strengths of both into a single model R-GCN+: $f(s,r,t)_{\text{R-GCN+}} = \alpha f(s,r,t)_{\text{R-GCN}} + (1-\alpha) f(s,r,t)_{\text{DistMult}}$, with $\alpha = 0.4$ selected on FB15k development data. To facilitate a fair comparison to R-GCN, we use half-size embeddings for each component of 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.

In Table 4, we show results for FB15k-237 where (as previously discussed) inverse relation pairs have been removed and the LinkFeat baseline fails to generalize[5]. 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. The high variance between different decoder-only models suggests that performance could be improved by combining R-GCN with a task-specific decoder selected through validation. As decoder choice is orthogonal to the development of our encoder model, we leave this as a promising avenue for future work.

In Table 5, we evaluate the R-GCN model and the combination model 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.

6 Related Work

6.1 Relational Modeling

Our encoder-decoder approach to link prediction relies on DistMult [11] in the decoder, a special and simpler case of the RESCAL factorization [32], more

[5] Our numbers are not directly comparable to those reported in [12], as they use pruning both for training and testing (see their Sects. 3.3.1 and 4.2). Since their pruning schema is not fully specified (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.

Table 5. Results on the FB15k and WN18 datasets. Results marked (*) taken from [20]. Results marks (**) taken from [30].

	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

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 nonlinear ones (e.g., [17,20, 29,30,33,34]). Many of these approaches can be regarded as modifications or special cases of classic tensor decomposition methods such as CP or Tucker; for an overview of tensor decomposition literature we refer the reader to [35].

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 [36,37]; (2) approaches using paths (or walks) as features when predicting edges [18]; or (3) doing both at the same time [19,38]. 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 [13,14,39,40] for large-scale and highly multi-relational data, characteristic of realistic knowledge bases.

Early work in this area includes the graph neural network (GNN) [15]. A number of extensions to the original GNN have been proposed, most notably [41,42], both of which use gating mechanisms to facilitate optimization.

R-GCNs can further be seen as a sub-class of message passing neural networks [16], which encompass a number of previous neural models for graphs, including GCNs, under a differentiable message passing interpretation.

As mentioned in Sect. 5, we do not in this paper experiment with subsampling of neighborhoods, a choice which limits our training algorithm to full-batch descent. Recent work including [43–45] have experimented with various subsampling strategies for graph-based neural networks, demonstrating promising results.

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 decoder 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 ConvE [34], 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 [43]. 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

1. Yao, X., Van Durme, B.: Information extraction over structured data: question answering with freebase. In: ACL (2014)
2. Bao, J., Duan, N., Zhou, M., Zhao, T.: Knowledge-based question answering as machine translation. In: ACL (2014)

3. Seyler, D., Yahya, M., Berberich, K.: Generating quiz questions from knowledge graphs. In: Proceedings of the 24th International Conference on World Wide Web (2015)
4. Hixon, B., Clark, P., Hajishirzi, H.: Learning knowledge graphs for question answering through conversational dialog. In: Proceedings of NAACL HLT, pp. 851–861 (2015)
5. Bordes, A., Usunier, N., Chopra, S., Weston, J.: Large-scale simple question answering with memory networks. arXiv preprint arXiv:1506.02075 (2015)
6. Dong, L., Wei, F., Zhou, M., Xu, K.: Question answering over freebase with multi-column convolutional neural networks. In: ACL (2015)
7. Kotov, A., Zhai, C.: Tapping into knowledge base for concept feedback: leveraging conceptnet to improve search results for difficult queries. In: WSDM (2012)
8. Dalton, J., Dietz, L., Allan, J.: Entity query feature expansion using knowledge base links. In: ACM SIGIR (2014)
9. Xiong, C., Callan, J.: Query expansion with freebase. In: Proceedings of the 2015 International Conference on The Theory of Information Retrieval, pp. 111–120 (2015)
10. Xiong, C., Callan, J.: Esdrank: connecting query and documents through external semi-structured data. In: CIKM (2015)
11. Yang, B., Yih, W., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. arXiv preprint arXiv:1412.6575 (2014)
12. Toutanova, K., Chen, D.: Observed versus latent features for knowledge base and text inference. In: Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality, pp. 57–66 (2015)
13. Duvenaud, D.K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., Adams, R.P.: Convolutional networks on graphs for learning molecular fingerprints. In: NIPS (2015)
14. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: ICLR (2017)
15. Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: The graph neural network model. IEEE Trans. Neural Netw. **20**(1), 61–80 (2009)
16. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: ICML (2017)
17. Socher, R., Chen, D., Manning, C.D., Ng, A.: Reasoning with neural tensor networks for knowledge base completion. In: NIPS (2013)
18. Lin, Y., Liu, Z., Luan, H., Sun, M., Rao, S., Liu, S.: Modeling relation paths for representation learning of knowledge bases. In: EMNLP (2015)
19. Toutanova, K., Lin, V., Yih, W., Poon, H., Quirk, C.: Compositional learning of embeddings for relation paths in knowledge base and text. In: ACL (2016)
20. Trouillon, T., Welbl, J., Riedel, S., Gaussier, E., Bouchard, G.: Complex embeddings for simple link prediction. In: ICML (2016)
21. Kipf, T.N., Welling, M.: Variational graph auto-encoders. arXiv preprint arXiv:1611.07308 (2016)
22. Ristoski, P., de Vries, G.K.D., Paulheim, H.: A collection of benchmark datasets for systematic evaluations of machine learning on the semantic web. In: Groth, P., Simperl, E., Gray, A., Sabou, M., Krötzsch, M., Lecue, F., Flöck, F., Gil, Y. (eds.) ISWC 2016. LNCS, vol. 9982, pp. 186–194. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46547-0_20

23. Ristoski, P., Paulheim, H.: RDF2Vec: RDF Graph embeddings for data mining. In: Groth, P., Simperl, E., Gray, A., Sabou, M., Krötzsch, M., Lecue, F., Flöck, F., Gil, Y. (eds.) ISWC 2016. LNCS, vol. 9981, pp. 498–514. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46523-4_30
24. Shervashidze, N., Schweitzer, P., Leeuwen, E.J., Mehlhorn, K., Borgwardt, K.M.: Weisfeiler-lehman graph kernels. J. Mach. Learn. Res. **12**(Sep), 2539–2561 (2011)
25. de Vries, G.K.D., de Rooij, S.: Substructure counting graph kernels for machine learning from rdf data. Web Semant. Sci. Serv. Agents World Wide Web **35**, 71–84 (2015)
26. Paulheim, H., Fümkranz, J.: Unsupervised generation of data mining features from linked open data. In: Proceedings of the 2nd International Conference on Web Intelligence, Mining And Semantics, p. 31 (2012)
27. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: NIPS (2013)
28. Kingma, D., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
29. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: NIPS (2013)
30. Nickel, M., Rosasco, L., Poggio, T.: Holographic embeddings of knowledge graphs. In: AAAI (2016)
31. Hitchcock, F.L.: The expression of a tensor or a polyadic as a sum of products. Stud. Appl. Math. **6**(1–4), 164–189 (1927)
32. Nickel, M., Tresp, V., Kriegel, H.P.: A three-way model for collective learning on multi-relational data. In: ICML (2011)
33. Chang, K.W., Yih, W., Yang, B., Meek, C.: Typed tensor decomposition of knowledge bases for relation extraction. In: EMNLP (2014)
34. Dettmers, T., Minervini, P., Stenetorp, P., Riedel, S.: Convolutional 2D knowledge graph embeddings. In: AAAI (2018)
35. Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM Rev. **51**(3), 455–500 (2009)
36. Guu, K., Miller, J., Liang, P.: Traversing knowledge graphs in vector space. In: EMNLP (2015)
37. Garcia-Duran, A., Bordes, A., Usunier, N.: Composing relationships with translations. Technical report. CNRS, Heudiasyc (2015)
38. Neelakantan, A., Roth, B., McCallum, A.: Compositional vector space models for knowledge base completion. In: ACL (2015)
39. Bruna, J., Zaremba, W., Szlam, A., LeCun, Y.: Spectral networks and locally connected networks on graphs. In: ICLR (2014)
40. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In: NIPS (2016)
41. Li, Y., Tarlow, D., Brockschmidt, M., Zemel, R.: Gated graph sequence neural networks. In: ICLR (2016)
42. Pham, T., Tran, T., Phung, D., Venkatesh, S.: Column networks for collective classification. In: AAAI (2017)
43. Hamilton, W.L., Ying, R., Leskovec, J.: Inductive representation learning on large graphs. In: NIPS (2017)
44. Chen, J., Zhu, J.: Stochastic training of graph convolutional networks. arXiv preprint arXiv:1710.10568 (2017)
45. Chen, J., Ma, T., Xiao, C.: FastGCN: fast learning with graph convolutional networks via importance sampling. In: ICLR (2018)