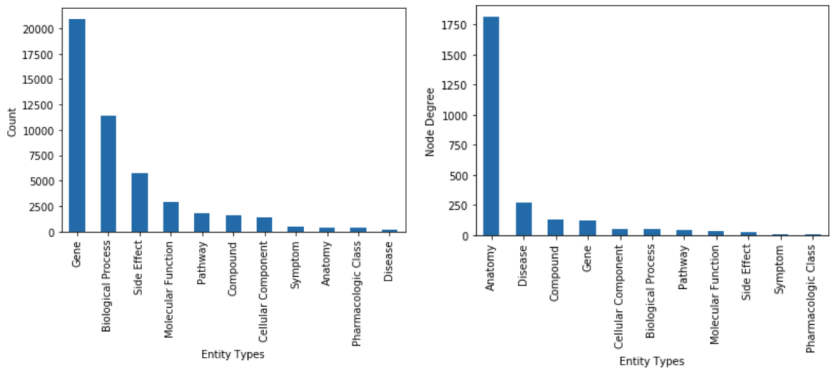
A Dataset Hetionet



(a) **(b)**

Fig. 1. Comparison of the (a) total counts and (b) the average node degrees according to each entity type in Hetionet.

B Experimental Details

MINERVA+ and MINERVA

For MINERVA+ and MINERVA, we tune the models over the hyperparameters λ which balances the two rewards, b which defines when the additional reward is added, the entity and relation embedding size, the number of LSTM layers, the hidden layer size for the LSTM and MLP (see Equation 2), the learning rate, and β which is the entropy regularization constant. The hyperparameter search space is given in Table 1. The path length is set to 3, the number of rollouts is given by 30 for training, and the number of test rollouts at inference time is set to 100. All experiments were conducted on a machine with 16 CPU cores and 32 GB RAM. Training our method on the Hetionet dataset takes around 10 minutes, testing about 10 seconds.

Table 1. Hyperparameter search space for MINERVA+/MINERVA.

Hyperparameter	Values
Embedding size	{64, 128, 256}
Hidden layer size	{128, 256, 512}
Number LSTM layers	{1, 2}
λ	$\{1, \ldots, 5\}$
b	$\{1, \mathbb{1}_{\{e_{T+1}=e_d\}}\}$
β	[0.01, 0.1]
Learning rate	[0.0001, 0.01]

AnyBURL

We use the implementation by the authors of the most recent version of AnyBURL[1] [3]. The rules are learned for 500 seconds, and the maximum length of rules is set to 3. Since AnyBURL does not need extensive hyperparameter tuning to achieve good results, we kept the default values of all other hyperparameters.

LibKGE

LibKGE[2] is a library for training, evaluation, and hyperparameter optimization of knowledge graph embedding models. We do a random search for the models TransE, DistMult, ComplEx, ConvE, and RESCAL across the hyperparameter search space given in Table 2.

[1] `http://web.informatik.uni-mannheim.de/AnyBURL/`
[2] `https://github.com/uma-pi1/kge`

Table 2. Hyperparameter search space for LibKGE. Models with specific hyperparameters are indicated in parenthesis.

Hyperparameter	Values
Embedding size	$\{64, 128, 256\}$
Training type	Negative sampling
Reciprocal	{True, False}
Number subject samples	$\{1, \dots, 1000\}$, log scale
Number object samples	$\{1, \dots, 1000\}$, log scale
Loss	Cross-entropy
L_p-norm (TransE)	{1, 2}
Optimizer	{Adam, Adagrad}
Learning rate	$[10^{-4}, 1]$, log scale
LR scheduler patience	$\{1, \dots, 10\}$
L_p regularization	{None, L_2}
Entity emb. weight	$[10^{-20}, 10^{-5}]$
Relation emb. weight	$[10^{-20}, 10^{-5}]$
Frequency weighting	{True, False}
Embedding normalization (TransE)	
Entity	{True, False}
Relation	{True, False}
Dropout	
Entity embedding	[0.0, 0.5]
Relation embedding	[0.0, 0.5]
Feature map (ConvE)	[0.0, 0.5]
Projection (ConvE)	[0.0, 0.5]

R-GCN

We used the implementation provided by the python package Pytorch Geometric[3]. The hyperparameters are tuned according to the ranges specified in Table 3.

Table 3. Hyperparameter search space for R-GCN.

Hyperparameter	Values
Number of conv. layers	$\{1, 2, 3\}$
Embedding size	$\{8, 16, 32\}$
Learning rate	$[0.1, 0.001]$

[3] `https://github.com/rusty1s/pytorch_geometric`

CompGCN

For CompGCN, we use the implementation by the authors[4]. The hyperparameters of CompGCN are tuned in the ranges specified in Table 4.

Table 4. Hyperparameter search space for CompGCN.

Hyperparameter	Values
Embedding size	{64, 128, 256}
GCN layer size	{64, 128, 256}
Number of GCN layers	{1, 2}
Scoring function	{TransE, ConvE, Distmult}
Composition operation	Circular-correlation
Dropout rate	0.1
Learning rate	0.001

pLogicNet

For the experiments on pLogicNet, we use the implementation by the authors[5] and report the results of pLogicNet*, which yields better results than plain pLogicNet. Table 5 shows the hyperparameter search space.

Table 5. Hyperparameter search space for pLogicNet.

Hyperparameter	Values
Embedding size	{500, 1000}
Number negative samples	{256, 512}
α	{0.5, 1}
γ	{6, 12}
λ	{1, 50, 100}
τ_{rule}	{0.1, 0.6}

[4] `https://github.com/malllabiisc/CompGCN`
[5] `https://github.com/DeepGraphLearning/pLogicNet`

C Experiments on Other Datasets

All metrics are filtered [1] and evaluated for tail-sided predictions. For AnyBURL, we apply either only cyclic rules for inference (AnyBURL (cyclic)) or include also acyclic rules of length 1 (AnyBURL).

Table 6. Comparison with baseline methods on FB15k-237.

Method	Hits@1	Hits@3	Hits@10	MRR
AnyBURL[a]	0.354	0.479	0.611	0.432
AnyBURL (cyclic)	0.292	0.397	0.539	0.362
TransE[b]	0.321	0.461	0.613	0.418
DistMult[b]	0.340	0.486	0.634	0.439
ComplEx[b]	0.346	0.493	0.639	0.445
ConvE[b]	0.342	0.481	0.630	0.439
RESCAL[b]	0.354	0.498	0.644	0.452
COMPGCN	0.356	0.496	0.638	0.452
Neural LP [2]	0.166	0.248	0.348	0.227
pLogicNet[c]	0.327	0.475	0.631	0.429
MINERVA [2]	0.217	0.329	0.456	0.293
MINERVA+	0.238	0.325	0.438	0.302
MINERVA+ (pruned)	0.256	0.351	0.458	0.321

Best model hyperparameters: [a] [3], [b] [5], [c] [4].

Table 7. Comparison with baseline methods on WN18RR.

Method	Hits@1	Hits@3	Hits@10	MRR
AnyBURL[a]	0.490	0.547	0.609	0.527
AnyBURL (cyclic)	0.445	0.510	0.560	0.482
TransE[b]	0.064	0.387	0.555	0.245
DistMult[b]	0.443	0.495	0.556	0.481
ComplEx[b]	0.455	0.513	0.565	0.494
ConvE[b]	0.426	0.475	0.532	0.461
RESCAL[b]	0.453	0.498	0.536	0.483
COMPGCN	0.460	0.512	0.564	0.497
Neural LP [2]	0.376	0.468	0.657	0.463
pLogicNet[c]	0.403	0.455	0.568	0.451
MINERVA [2]	0.413	0.456	0.513	0.448
MINERVA+	0.430	0.475	0.526	0.402
MINERVA+ (pruned)	0.446	0.495	0.545	0.478

Best model hyperparameters: [a] [3], [b] [5], [c] [4].

References

1. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: The Twenty-Seventh Conference on Neural Information Processing Systems (2013)
2. Das, R., Dhuliawala, S., Zaheer, M., Vilnis, L., Durugkar, I., Krishnamurthy, A., Smola, A., McCallum, A.: Go for a walk and arrive at the answer: reasoning over paths in knowledge bases using reinforcement learning. In: The Sixth International Conference on Learing Representations (2018)
3. Meilicke, C., Chekol, M.W., Fink, M., Stuckenschmidt, H.: Reinforced anytime bottom up rule learning for knowledge graph completion. Preprint arXiv:2004.04412 (2020)
4. Qu, M., Tang, J.: Probabilistic logic neural networks for reasoning. In: The Thirty-Third Conference on Neural Information Processing Systems (2019)
5. Ruffinelli, D., Broscheit, S., Gemulla, R.: You can teach an old dog new tricks! On training knowledge graph embeddings. In: The Ninth International Conference on Learning Representations (2020)