Learning Predictive Categories Using Lifted Relational Neural Networks

Gustav Šourek[1], Suresh Manandhar[2], Filip Železný[1], Steven Schockaert[3], and Ondřej Kuželka[3]

[1] Czech Technical University, Prague, Czech Republic
{souregus,zelezny}@fel.cvut.cz
[2] Department of Computer Science, University of York, York, UK
suresh.manandhar@york.ac.uk
[3] School of CS & Informatics, Cardiff University, Cardiff, UK
{SchockaertS1,KuzelkaO}@cardiff.ac.uk

Abstract. Lifted relational neural networks (LRNNs) are a flexible neural-symbolic framework based on the idea of lifted modelling. In this paper we show how LRNNs can be easily used to specify declaratively and solve learning problems in which latent categories of entities, properties and relations need to be jointly induced.

1 Introduction

Lifted models, such as Markov logic networks (MLNs [13]), are first-order representations that define patterns from which specific (ground) models can be unfolded. For example, in a MLN we may express the pattern that *friends of smokers tend to be smokers*, which then constrains the probabilistic relationships between specific individuals in the derived ground Markov network. Inspired by this idea, in [16] we introduced a method that uses weighted relational rules for learning feed-forward neural networks, called *Lifted Relational Neural Networks* (LRNNs). This approach differs from standard neural networks in two important ways: (i) the network structure is derived from symbolic rules and thus has an intuitive interpretation, and (ii) the weights of the network are tied to the first-order rules and are thus shared among different neurons.

In this paper, we first show how LRNNs can be used to learn a latent category structure that is predictive in the sense that the properties of a given entity can be largely determined by the category to which that entity belongs, and dually, the entities satisfying a given property can be largely determined by the category to which that property belongs. This enables a form of transductive reasoning which is based on the idea that similar entities have similar properties. We then extend this model into a relational setting, in which entities not only have properties but can also be linked by arbitrary relations.

The proposed approach is similar in spirit to [7], which instead uses crisp clustering based on second-order MLNs. However, the use of LRNNs has several important advantages for learning latent concepts. Firstly, LRNNs do not need

to invoke costly EM algorithms and hence can be more efficient than latent variable probabilistic models. Secondly, the learnt soft clusters can naturally be interpreted as vector space embeddings of entities, properties and relations. Finally, the flexibility of LRNNs means that the considered form of transductive reasoning can be extended in a natural way to take into account various forms of prior domain knowledge, as well as alternative types of heuristic reasoning (e.g. reasoning by analogy, modelling persistence or periodic behaviour).

The remainder of this paper is structured as follows. In Section 2 we briefly describe the LRNN framework from [16], after which we introduce a technique to deal with recursive rules in LRNNs. We describe the predictive model for the non-relational setting in Section 3.1 and for the relational setting in Section 3.2. Next, in Section 4, we describe a simple model encoded as a LRNN which is based on similarity-based reasoning. In Section 5 we evaluate the method experimentally. Finally, we discuss related work in Section 6 and conclude the paper in Section 7

2 Lifted Relational Neural Networks

2.1 The Basic Framework

A lifted relational neural network (LRNN) $\mathcal{N}$ is a set of weighted definite first-order clauses[4]. Let $\mathcal{H}_{\mathcal{N}}$ be the least Herbrand model of the classical theory $\{\alpha : (\alpha, w) \in \mathcal{N}\}$, with $\mathcal{N}$ a LRNN. We define the *grounding of* $\mathcal{N}$ as $\overline{\mathcal{N}} = \{(h\theta \leftarrow b_1\theta \wedge \cdots \wedge b_k\theta, w) : (h \leftarrow b_1 \wedge \cdots \wedge b_k, w) \in \mathcal{N} \text{ and } \{h\theta, b_1\theta, \ldots, b_k\theta\} \subseteq \mathcal{H}_{\mathcal{N}}\}$.

Definition 1. *Let $\mathcal{N}$ be a LRNN, and let $\overline{\mathcal{N}}$ be its grounding. Let $g_\vee$, $g_\wedge$ and g_* be functions[5] from $\bigcup_{i=1}^{\infty} \mathbb{R}^i$ to $\mathbb{R}$. The* ground neural network *of $\mathcal{N}$ is a feed-forward neural network constructed as follows.*

- *For every ground atom h occurring in $\overline{\mathcal{N}}$, there is a neuron A_h with activation function $g_\vee$, called* atom neuron.
- *For every ground fact $(h, w) \in \overline{\mathcal{N}}$, there is a neuron $F_{(h,w)}$, called* fact neuron*, which has no input and always outputs the constant value w.*
- *For every ground rule $(h\theta \leftarrow b_1\theta \wedge \cdots \wedge b_k\theta, w) \in \overline{\mathcal{N}}$, there is a neuron $R_{h\theta \leftarrow b_1\theta \wedge \cdots \wedge b_k\theta}$ with activation function $g_\wedge$, called* rule neuron*. It has the atom neurons $A_{b_1\theta}, \ldots, A_{b_k\theta}$ as inputs, all with weight* 1*.*
- *For every rule $(h \leftarrow b_1 \wedge \cdots \wedge b_k, w) \in \mathcal{N}$ and every $h\theta \in \mathcal{H}_{\mathcal{N}}$, there is a neuron $Agg^{h\theta}_{(h \leftarrow b_1 \wedge \cdots \wedge b_k, w)}$ with activation function g_*, called* aggregation neuron*. Its inputs are all rule neurons $R_{h\theta' \leftarrow b_1\theta' \wedge \cdots \wedge b_k\theta'}$ where $h\theta = h\theta'$ with all weights equal to* 1*.*
- *Inputs of an atom neuron $A_{h\theta}$ are the aggregation neurons $Agg^{h\theta}_{(h \leftarrow b_1 \wedge \cdots \wedge b_k, w)}$ and fact neurons $F_{(h\theta, w)}$, with the input weights determined by the outputs of the aggregation and fact neurons.*

[4] Established notions such as "rule" are further used also for their weighted analogies.
[5] These represent aggregation operators that can take a variable number of arguments.

Depending on the used families of activation functions $g_\wedge$, $g_\vee$ and g_*, we can obtain neural networks with different behavior. In this paper we will use:

$$g_\wedge(b_1,\ldots,b_k) = sigm\Big(\sum_{i=1}^{k} b_i - k + b_0\Big) \quad g_\vee(b_1,\ldots,b_k) = sigm\Big(\sum_{i=1}^{k} b_i + b_0\Big)$$

$$g_*(b_1,\ldots,b_m) = \frac{1}{m}\sum_{i=1}^{m} b_i$$

Where *sigm* denotes the logistic sigmoid function $sigm(x) = \frac{1}{1+(e^{-x})}$, which also implies that $\forall i : 0 < b_i < 1$. Note that $g_\wedge$ and $g_\vee$ are closely related to the conjunction and disjunction from Łukasiewicz logic [6], which is in accordance with the intuition that $g_\wedge$ should only have a high output if all its inputs are high, while $g_\vee$ should be high as soon as one of the inputs is high.

2.2 Handling Recursion in LRNNs

As originally introduced in [16], LRNNs did not support recursive rules, in order to avoid the potential need to work with recurrent neural networks, since these are more difficult to train than feed-forward neural networks. However in general, recursive rules do not necessarily pose a problem to the LRNN framework as long as they do not induce directed cycles in the resulting ground neural networks. For instance, rules defining directed paths in acyclic graphs would not lead to directed cycles in the resulting ground neural networks, despite being recursive. One minor complication caused by allowing LRNNs to have recursion, even in the absence of directed cycles, is that weights may be shared among neurons that lie on a directed path from the input to the output of the network. This makes the computation of gradients more complicated than in the normal case. Although weights in such LRNNs, whose groundings are still feed-forward neural networks, can still be learned using Stochastic Gradient Descent (SGD).

In this paper we will use recursive rule sets that may potentially lead to recurrent neural networks. In order to maintain the feed-forward nature of the resulting ground neural networks, we modify the strategy for constructing ground networks as follows. First we construct the ground network exactly as described in Section 2. If this network contains directed cycles, we then proceed as follows. Let Q be a given ground query atom[6]. We find the respective atom neuron corresponding to Q in the ground network. If no such atom neuron exists, the output value for Q is 0. If there is such an atom neuron, we perform a breadth-first search from this atom neuron (traversing the connections between neurons in reverse, i.e. from output to input) and whenever we find an edge pointing *from* an already visited atom neuron, we delete it. The resulting ground neural network is then feed-forward. While this process enables us to stick with feed-forward neural networks, it comes at the price of a slightly less intuitive semantics, in

[6] In general LRNNs support non-ground query atoms but in this paper we will not need them. Therefore we assume only ground query atoms for simplicity.

which the inference and output for non-query atom neurons may also depend on the used queries. This is not problematic for any of the applications considered in this paper, as non-query atoms are not used within these application.

3 Learning Predictive Categories

In this section, we introduce a class of LRNN models that are aimed at learning predictive categories of entities, properties and relations. We first introduce a model for attribute-valued data in Section 3.1, which is extended to cope with relational data in Section 3.2.

3.1 Predictive Categories for Attribute-valued Data

Let a set of entities be given, and for each entity, a list of properties that it satisfies. The basic assumption underlying our model is that there exist some (possibly overlapping) categories, such that every entity can be described accurately enough by its soft membership to each of these categories. We furthermore assume that these categories can themselves be organised in a set of higher-level categories. The idea is that the category hierarchy should allow us to predict which properties a given entity has, where the properties associated with higher-level categories are typically (but not necessarily) inherited by their sub-categories. To improve the generalization ability of our method, we assume that a dual category structure exists for properties. The main task we consider is to learn these (latent) category structures from the given input data.

To encode the above described model in a LRNN, we proceed as follows. We use $\mathit{HasProperty}(e, p)$ to denote that the entity e has the property p. For every entity e and for each category c at the lowest level of the category hierarchy, we construct the following ground rule:

$$w_{ec} : \mathit{IsA}(e, c)$$

Note that weight w_{ec} intuitively reflects the soft membership of e to the category c; it will be determined when training the ground network. Similarly, for each category c_1 at a given level and each category c_2 one level above, we add the following ground rule:

$$w_{c_1 c_2} : \mathit{IsA}(c_1, c_2)$$

In the same way, ground rules are added that link each property to a property category at the lowest level, as well as ground rules that link property categories to higher-level categories. To encode the idea that entity categories should be predictive of properties, we add the following rule for each entity category c_e and each property category c_p:

$$w_{c_e c_p} : \mathit{HasProperty}(A, B) \leftarrow \mathit{IsA}(A, c_e), \mathit{IsA}(B, c_p).$$

The weights $w_{c_e c_p}$ encode which entity categories are related to which property categories, and will again be determined when training weights of the LRNN. To encode transitivity of the is-a relationship, we simply add the following rule:

$$w_{isa} : IsA(A, C) \leftarrow IsA(A, B), IsA(B, C).$$

Training examples are encoded as a set of facts of the form ($HasProperty(e, p)$, l) where $l \in \{0, 1\}$, 0 denoting a negative example and 1 a positive example. We train the model using SGD as described in [16]. In particular, in a LRNN, there is a neuron for any ground literal which is logically entailed by the rules and facts in the LRNN and the output of this neuron represents the truth value of this literal. Therefore if we want to train the weights of the LRNN, we just optimize the weights of the network w.r.t. a loss function such as the mean squared error, where the loss function is computed from the desired truth values of the query literals and the outputs obtained from the respective atom neurons

3.2 Predictive Categories for Relational Data

The model from Section 3.1 can be extended to cope with relational facts. Similar to our encoding of properties, we will use a reified representation of relational facts, writing e.g. $Relation(ParentOf, e_1, e_2)$ to denote that e_1 is the parent of e_2. In this way, we can induce predictive relation categories, similar to the entity and property categories considered in Section 3.1.

To this end, analogously as for entity and property categories, for every relation r and every (latent) relation category c we add the following ground rule:

$$w_{rc} : IsA(r, c)$$

For each relation category c_1 at a given level and each category c_2 one level above, we add the following ground rule:

$$w_{c_1 c_2} : IsA(c_1, c_2).$$

Note that a rule encoding transitivity of the IsA relation was already added in the first part of the model. Finally we encode that, like properties, relations among entities are typically determined by their categories. Specifically, for each triple consisting of a pair of (not necessarily distinct) entity clusters c_e, c'_e and a relation cluster c_r, we add the following ground rule:

$$w_{c_r c_e c'_e} : Relation(R, A, B) \leftarrow IsA(R, c_r), IsA(A, c_e), IsA(B, c'_e) \qquad (1)$$

The LRNNs defined in this way will be referred to as *fully-connected*, as they contain rules for every *relation-entity-entity* triple. Obviously, when a high number of clusters is used, the number of rules of the form (1) may be prohibitively high. To address this, we can limit the triples for which such rules are added. In particular, we will consider LRNNs which restrict such rules to those of the following form:

$$w_{c_r c_{2i} c_{2i+1}} : Relation(R, A, B) \leftarrow IsA(R, c_r), IsA(A, c_{2i}), IsA(B, c_{2i+1}) \qquad (2)$$

where c_1, c_2, ..., c_n are entity concepts. In fact the LRNNs with rules of this form can learn anything that can be learned by LRNNs with rules of the form (1) as long as they have enough rules.

In addition, to help the model learn symmetric and transitive relations (e.g. the "same-political-bloc" relation), we also add rules of the following form:

$$w_{c_r c_i' c_i'} : \mathit{Relation}(R, A, B) \leftarrow \mathit{IsA}(R, c_r), \mathit{IsA}(A, c_i'), \mathit{IsA}(B, c_i') \tag{3}$$

Note that we do not need to explicitly consider these in the fully-connected model, as they are a special case of (1).

4 Prediction Using Learned Similarities

In this section we describe a LRNN model based on similarity degrees, for the same predictive task that was considered in the previous section. While the similarity degrees could be obtained from any source, we will use similarity degrees that have been obtained from the model described in the previous section, by taking advantage of the fact that the cluster membership degrees can be interpreted as defining a vector-space embedding. Rather than using the membership degrees directly, we will use the weights of the respective ground $IsA(e, c)$ rules, which, unlike the membership degrees, may also be negative[7]. In particular, the similarity degree between two entities is defined as the cosine similarity between the vector representation of these entities, with the coordinates of these vectors the soft memberships of the entity in each of the categories.

For each pair of entities (e_1, e_2) with similarity degree s, we add the following ground fact:

$$1.0 : \mathit{Similar}(e_1, e_2, s)$$

We furthermore add rules which encode a learnable transformation of the similarities into a score which is useful for the given predictive task:

$$\begin{aligned}
&w_{-1} : && \mathit{Similar}(X, Y) \leftarrow \mathit{Similar}(X, Y, S), S \geq -1.0 \\
&w_{-0.9} : && \mathit{Similar}(X, Y) \leftarrow \mathit{Similar}(X, Y, S), S \geq -0.9 \\
&\ldots \\
&w_{0.9} : && \mathit{Similar}(X, Y) \leftarrow \mathit{Similar}(X, Y, S), S \geq \quad 0.9
\end{aligned}$$

Finally we add one rule of the following type for every relation r:

$$w_r : \mathit{Relation}(r, X, Y) \leftarrow \mathit{Relation}(r, V, W), \mathit{Similar}(X, V), \mathit{Similar}(Y, W).$$

Taking into account the aggregative nature of the used family of activation functions (cf. Section 2), these rules encode the intuition that in order to predict if

[7] The membership degrees are simply obtained as applying sigmoids on the respective weights in this particular case, so the two representations essentially bear the same information

X and Y are in relation r, we could check how similar on average the entities known to be in this relation are to X and Y.

Naturally, not all relations can be accurately predicted by a model like the one described in this section. However, this similarity based approach is quite natural, and serves as an important illustrative example of how other strategies could be encoded (e.g. interpolation/extrapolation or reasoning by analogy).

5 Evaluation

5.1 Evaluation of the Model from Section 3.1

To evaluate the potential of the model proposed in Section 3.1, we have used the Animals dataset[8], which describes 50 animals in terms of 85 Boolean features, such as *fish, large, smelly, strong*, and *timid*. This dataset was originally created in [11], and was used among others for evaluating a related learning task in [7]. For both entities and properties, we have used two levels of categories, with in both cases three categories at the lowest level and two categories at the highest level.

Recall that we can view the category membership degrees as defining a vector-space embedding. Figures 1 and 2 show the first two principal components of this embedding for a number of entities and properties. We can see, for instance, that sea mammals are clustered together, and that predators tend to be separated from herbivores. In Figure 2, we have highlighted two types of properties: colours and teeth types. Note that these do not form clusters (e.g. a cluster of colours) but they represent, as prototypes, different clusters of properties which tend to occur together. For instance, *blue* is surrounded by properties which typically hold for water mammals; white and red occur together with stripes, nocturnal, pads; gray occurs together with small and weak; etc. We also evaluated the predictive ability of this model. We randomly divided the facts from the dataset in two halves, trained the model on one half and tested it on the other one, obtaining AUC ROC of 0.77. We also performed an experiment with a 90-10 split, in order to be able to directly compare our results with those from [7]; we obtained the same AUC PR 0.8 as reported in [7] (and AUC ROC 0.86).

5.2 Evaluation of the Model from Section 3.2

In order to evaluate the relational method proposed in Section 3.2 we performed experiments with two relational datasets:[9] Nations and UMLS. These datasets have previously been used to evaluate statistical predicate invention methods in [7]. The Nations dataset contains a set of relations between pairs of nations and their features [15]. It consists of relations such as *ExportsTo* and *GivesEconomicAidTo*, as well as properties such as *Monarchy*. The dataset contains 14

[8] Downloaded from `https://alchemy.cs.washington.edu/data/animals/`.

[9] Downloaded from `https://alchemy.cs.washington.edu/data/nations/` and from `https://alchemy.cs.washington.edu/data/umls/`.

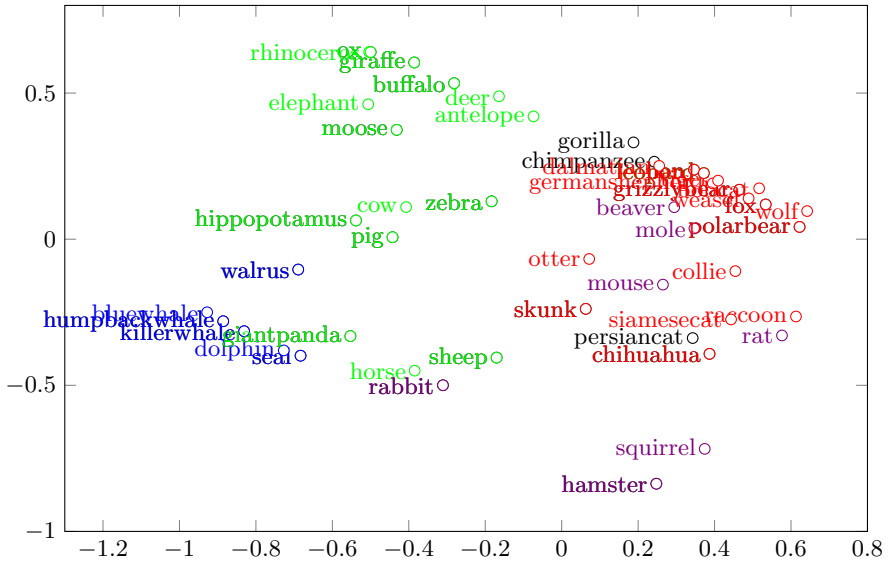

Fig. 1. Embedding of entities (animals, only a subset of entities is displayed). Several homogeneous groups of animals are highlighted: sea mammals (blue), large herbivores (green), rodents (violet), and other predators (red).

nations, 56 relations and 111 properties. There are 2565 true ground atoms. The UMLS dataset contains data from the Unified Medical Language System, which is a biomedical ontology [8]. It contains 49 relations and 135 biomedical entities. There are 6529 true ground atoms in this dataset.

Initial experiments have revealed two trends. First, accuracy consistently improved when we increased the size of the LRNNs (contrarily to our expectation that overfitting might be a problem when increasing the size). Second, for a fixed number of entity, property and relation categories, adding the layer of more general concepts helps, but it also increased memory consumption and runtime. Therefore, in the experiments, we created LRNNs as large as possible which still fitted in memory. A consequence of this strategy is that the LRNNs with more than one layer of categories had fewer categories in total than their single-layer counterparts. Similar effects also took place for fully-connected LRNNs when compared to LRNNs with isolated rules of the form (2) and (3); therefore we did not consider fully connected LRNNs in our experiments.

For the Nations dataset, the largest single-layer LRNN which fitted in 40GB of memory had 100 property categories, 100 entity categories and 50 relation categories. The cross-validated AUC ROC was 0.89 and AUC PR 0.74, which is within the standard error margin of the results obtained in [7]. The largest two-layer LRNN learned on this dataset had 20 property categories, 20 entity categories and 10 relation categories. Its cross-validated AUC ROC was 0.88 and AUC PR 0.7. For comparison, we also trained a single-layer LRNN with the

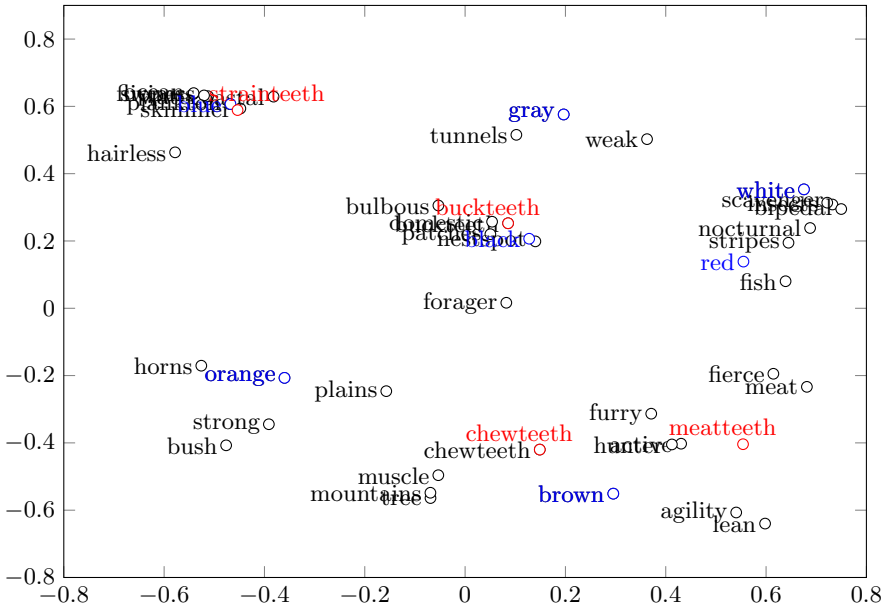

Fig. 2. Embedding of properties (only a subset of properties is displayed). Two representative groups of properties are shown in colour: colours (blue) and teeth-type (red).

exact same number of each type of categories, which achieved AUC ROC 0.86 and AUC PR 0.67, which agrees with the above described general trends.

The first two principal components of the embeddings of the states are displayed in Figure 3. When interpreting this embedding, note that this dataset relates to the political situation of 1950s.

For the UMLS dataset we used a LRNN with 100 entity categories and 50 relation categories. Due to the size of the dataset, consisting of a total of 893k ground facts and memory limitations, we only performed experiments with a largely subsampled training set, obtaining test AUC ROC 0.97 and AUC PR 0.76. This is a lower AUC PR than obtained by the method from [7], but it is close to the second-best method tested there and is better than the reported results for MLN structure learning.

5.3 Evaluation of the Model from Section 4

The evaluation of the model introduced in Section 4 primarily serves to estimate the usefulness of the embeddings learned by LRNNs. We have particulary focused on the embedding of countries, whose first two principal components are shown in Figure 3. First, we have split the nations dataset [15] into equally large training and testing parts. We trained the relational model described in Section 3.2, extracted the learned cluster membership degrees as vector embeddings, and

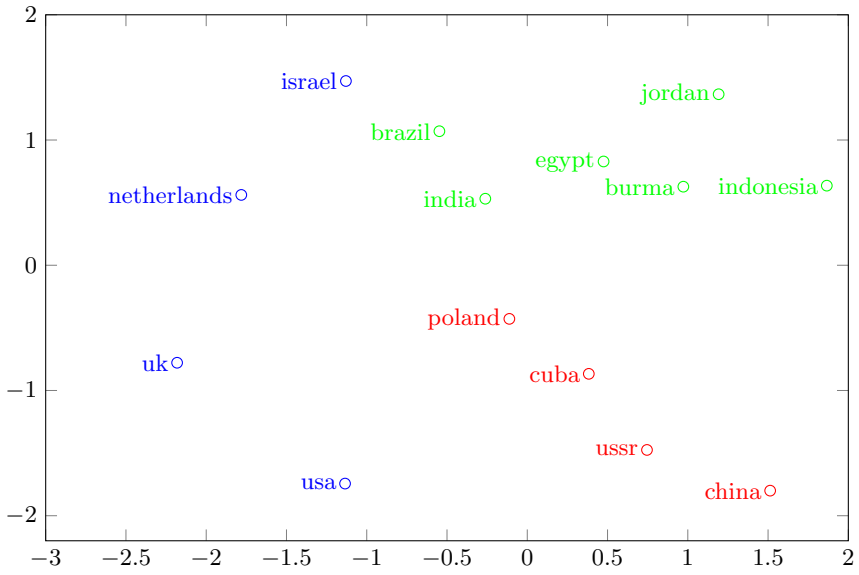

Fig. 3. Embeddings of countries as induced by learning from their geopolitical relations captured in the historical dataset [15]. A possible interpretation of the projection is displayed in colors, dividing them into communist (red), western (blue), and developing nations (green).

calculated their pairwise cosine similarities. We included these similarities as ground facts, together with all the true statements from the training part of the dataset. On top of these facts, we added the transformation and inference rules to form the model described in Section 4. We then trained this composite model on the same training part of the nations dataset that we used to obtain the embeddings, and evaluated its generalization ability on the remaining testing part. We obtained AUC ROC of 0.85 and 0.49 AUC PR, which is lower than the crossvalidated performance reported for the best models, but indirectly proves that the previously learned embeddings indeed carry useful information that may be subsequently reused for different predictive scenarios.

5.4 An Experiment with Real-Life Data from NELL

We have also evaluated the method on a real-life dataset. The main idea here was to analyse whether the LRNN models described in this paper could be used in an NLP pipeline to fill gaps in a knowledge base. To test this idea we downloaded a collection consisting of about 29k actors from NELL [10] with all their parental categories. For the experiments, we have subsampled the dataset to 2k actors. In the end, the number of different parental categories assigned to actors in this dataset turned out to be quite small. There were only 20 different categories such as *comedian* or *celebrity*, resulting into a dataset of 4k true ground facts, which

we completed with their negative complement under the closed world assumption for evaluation. In the experiments, we have tested the LRNN construct described in Section 3.1 and obtained a test-set AUC ROC 0.84 and AUC PR 0.43. This suggests that the LRNN method is indeed able to discover plausible properties of entities in datasets obtained from text. This could be quite useful for suggesting properties or relations in settings like NELL's where feedback from users is also used to validate the predictions.

6 Related Work

The proposed model essentially relies on the assumption that similar entities tend to have similar properties, for some similarity function which is learned implicitly in terms of category membership degrees. It is possible to augment this form of inference with other models of plausible reasoning, such as reasoning based on analogical (and other logical) proportions [9,12]. Moreover, as in [2], we could take into account externally obtained similarity degrees, using rules such as those in Section 4.

The model considered in this paper is related to statistical predicate invention [7] which relies on jointly clustering entities and relations. The dual representation of entity and property categories is also reminiscent of formal concept analysis [5]. LRNNs themselves are also related to the long stream of research in neural-symbolic integration [1], previous work on using neural networks for relational learning [3], and more recent approaches such as [14,4].

7 Conclusions and Future Work

We have illustrated how the declarative and flexible nature of LRNNs can be used for easy encoding of non-trivial learning scenarios. The models that we considered in this paper jointly learn predictive categories of entities, their properties and relations between them. The main strength of this approach lies in the ease with which the model can be extended to more complicated settings, which is mainly due to the declarative nature of LRNNs. It seems remarkable that such a declarative approach is able to obtain results which are close to the state-of-the-art method from [7], without tailoring any part of the learning method to this particular problem setting.

Our main direction for future work will focus on making LRNNs more scalable, which, as indicated by the performed experiments, should also lead to improved predictive performance.

Acknowledgments GS and FZ acknowledge support by project no. 17-26999S granted by the Czech Science Foundation. OK is supported by a grant from the Leverhulme Trust (RPG-2014-164). SS is supported by ERC Starting Grant 637277. Computational resources were provided by the CESNET LM2015042 and the CERIT Scientific Cloud LM2015085, provided under the programme "Projects of Large Research, Development, and Innovations Infrastructures".

References

1. Sebastian Bader and Pascal Hitzler. Dimensions of neural-symbolic integration-a structured survey. *arXiv preprint cs/0511042*, 2005.
2. Islam Beltagy, Cuong Chau, Gemma Boleda, Dan Garrette, Katrin Erk, and Raymond Mooney. Montague meets Markov: Deep semantics with probabilistic logical form. In *Proc. *SEM*, pages 11–21, 2013.
3. Hendrik Blockeel and Werner Uwents. Using neural networks for relational learning. In *ICML-2004 Workshop on Statistical Relational Learning and its Connection to Other Fields*, pages 23–28, 2004.
4. William W Cohen. Tensorlog: A differentiable deductive database. *arXiv preprint arXiv:1605.06523*, 2016.
5. Bernhard Ganter, Gerd Stumme, and Rudolf Wille. *Formal concept analysis: foundations and applications*, volume 3626. springer, 2005.
6. Petr Hájek. *Metamathematics of fuzzy logic*, volume 4. Springer Science & Business Media, 1998.
7. Stanley Kok and Pedro Domingos. Statistical predicate invention. In *Proceedings of the 24th International Conference on Machine Learning*, pages 433–440, 2007.
8. Alexa T McCray. An upper-level ontology for the biomedical domain. *Comparative and Functional Genomics*, 4(1):80–84, 2003.
9. Laurent Miclet, Sabri Bayoudh, and Arnaud Delhay. Analogical dissimilarity: definition, algorithms and two experiments in machine learning. *Journal of Artificial Intelligence Research*, 32:793–824, 2008.
10. Tom M. Mitchell, William W. Cohen, Estevam R. Hruschka Jr., Partha Pratim Talukdar, Justin Betteridge, Andrew Carlson, Bhavana Dalvi Mishra, Matthew Gardner, Bryan Kisiel, Jayant Krishnamurthy, Ni Lao, Kathryn Mazaitis, Thahir Mohamed, Ndapandula Nakashole, Emmanouil Antonios Platanios, Alan Ritter, Mehdi Samadi, Burr Settles, Richard C. Wang, Derry Tanti Wijaya, Abhinav Gupta, Xinlei Chen, Abulhair Saparov, Malcolm Greaves, and Joel Welling. Never-ending learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pages 2302–2310, 2015.
11. Daniel N Osherson, Joshua Stern, Ormond Wilkie, Michael Stob, and Edward E Smith. Default probability. *Cognitive Science*, 15(2):251–269, 1991.
12. Henri Prade and Gilles Richard. Reasoning with logical proportions. In *Twelfth International Conference on the Principles of Knowledge Representation and Reasoning*, 2010.
13. Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine learning*, 62(1-2):107–136, 2006.
14. Tim Rocktäschel and Sebastian Riedel. Learning knowledge base inference with neural theorem provers. In *NAACL Workshop on Automated Knowledge Base Construction (AKBC)*, 2016.
15. Rudolph J Rummel. *The dimensionality of nations project: attributes of nations and behavior of nations dyads, 1950-1965*. Number 5409. Inter-University Consortium for Political Research, 1976.
16. Gustav Šourek, Vojtěch Aschenbrenner, Filip Železný, and Ondřej Kuželka. Lifted relational neural networks. In *Proceedings of the NIPS Workshop on Cognitive Computation: Integrating Neural and Symbolic Approaches*, 2015.