Neural Markov Logic Networks

Giuseppe Marra[1] **Ondřej Kuželka**[2]

[1]Department of Computer Science, KU Leuven, Leuven, Belgium
[2]Faculty of Electrical Engineering, Czech Technical University in Prague, Prague, Czech Republic

Abstract

We introduce neural Markov logic networks (NMLNs), a statistical relational learning system that borrows ideas from Markov logic. Like Markov logic networks (MLNs), NMLNs are an exponential-family model for modelling distributions over possible worlds, but unlike MLNs, they do not rely on explicitly specified first-order logic rules. Instead, NMLNs learn an implicit representation of such rules as a neural network that acts as a potential function on fragments of the relational structure. Similarly to many neural symbolic methods, NMLNs can exploit embeddings of constants but, unlike them, NMLNs work well also in their absence. This is extremely important for predicting in settings other than the transductive one. We showcase the potential of NMLNs on knowledge-base completion, triple classification and on generation of molecular (graph) data.

1 INTRODUCTION

Statistical relational models are typically learned from one or more examples of relational structures that typically consist of a large number of ground atoms. Examples of such structures are social networks, protein-protein interaction networks etc. A challenging task is to learn a probability distribution over such relational structures from one or few examples. One possible approach is based on the assumption that the relational structure has repeated regularities; this assumption is implicitly or explicitly used in most works on statistical relational learning. Statistics about these regularities can be computed for small substructures of the training examples and used to construct a distribution over the whole relational structures. Together with the maximum-entropy principle, this leads to exponential-family distributions such as Markov logic networks [Richardson and Domingos, 2006]. In classical MLNs, however, either domain experts are required to design some useful statistics about the domain of interest by hand (i.e. logical rules) or they need to be learned by structure learning based on combinatorial search. Recently, many authors have tried to improve relational learning by integrating it with neural computation [Rocktäschel and Riedel, 2017, Kazemi and Poole, 2018, Sourek et al., 2018]. However, these hybrid approaches usually relax (or drop) the goal of modeling the joint probability distribution, preventing them from being applied to more complex learning and reasoning tasks.

In this paper, we propose neural Markov logic networks (NMLN). Here, the statistics (or features), which are used to model the probability distribution, are not known in advance, but are modelled as neural networks trained together with the probability distribution model. NMLNs overcome several limitations of existing approaches. In particular, *(i)* they can be used as an out-of-the-box tool in heterogeneous domains; *(ii)* they allow expressing and learning joint probability distributions of complete relational structures.

The main contributions presented in this paper are as follows: *(i)* we introduce a new class of potential functions exploiting symmetries of relational structures; *(ii)* we introduce a new statistical relational model called neural Markov logic networks *(iii)* we identify subclasses of NMLNs that allow for faster inference; *(iv)* we showcase the model's effectiveness on three diverse problems: generative modelling of small molecules, knowledge-base completion and triple classification.

The paper is structured as follows. In Section 2, we introduce preliminary concepts. In Section 3, we introduce the NMLN model. Section 4 focuses on a central component of NMLNs which are relational potential functions. In Section 5 we discuss inference in NMLNs. In Section 6, we show the results of the experiments we conducted. In Section 7, we position the proposed model in the literature. Finally, Section 8 concludes the paper.

Accepted for the 37th Conference on Uncertainty in Artificial Intelligence (UAI 2021).

2 PRELIMINARIES

We consider a function-free first-order logic language $\mathscr{L}$, which is built from a set of constants $\mathscr{C}_{\mathscr{L}}$ and predicates $\mathscr{R}_{\mathscr{L}} = \bigcup_i \mathscr{R}_i$, where $\mathscr{R}_i$ contains the predicates of arity i. For $c_1, c_2, \ldots, c_m \in \mathscr{C}_{\mathscr{L}}$ and $R \in \mathscr{R}_m$, we call $R(c_1, c_2, \ldots, c_m)$ a *ground atom*. We define *possible world* ω to be the pair $(\mathscr{C}, \mathscr{A})$, where $\mathscr{C} \subseteq \mathscr{C}_{\mathscr{L}}$, $\mathscr{A}$ is a subset of the set of all ground atoms that can be built from the constants in $\mathscr{C}$ and any relation in $\mathscr{R}_{\mathscr{L}}$. We define $\Omega_{\mathscr{L}}$ to be the set of all possible worlds over $\mathscr{L}$. Intuitively, a given possible world defines a set of *true* facts one can state using the constants (entities) and the relations of the language $\mathscr{L}$.

Definition 1 (Fragments)**.** *Let $\omega = (\mathscr{C}, \mathscr{A})$ be a possible world. A fragment $\omega\langle\mathscr{S}\rangle$ is defined as the restriction of ω to the constants in $\mathscr{S}$. It is a pair $\omega\langle S\rangle = (\mathscr{S}, \mathscr{B})$, with $\mathscr{S}$ the constants of the restriction and $\mathscr{B}$ a set of ground atoms only using constants from $\mathscr{S}$.*

Example 2.1. Given a language based on the set of constants $\mathscr{C}_{\mathscr{L}} = \{Alice, Bob, Eve\}$ and a set of relations $\mathscr{R}_{\mathscr{L}} = \{sm(x), fr(x,y)\}$, consider a possible world on this language $\omega = (\mathscr{C}_{\mathscr{L}}, \{sm(Alice), fr(Alice, Bob), fr(Bob, Eve)\})$.
Then, for instance, the fragment induced by the set of constants $\mathscr{S} = \{Alice, Bob\}$ is $\omega\langle S\rangle = (\mathscr{S}, \{sm(Alice), fr(Alice, Bob)\})$.

The set of all fragments of ω that are induced by size-k subsets of constants will be denoted by $\Gamma_k(\omega)$. Similarly, $\Gamma_k(\mathscr{L})$ will denote the set of all possible fragments in a given first-order language $\mathscr{L}$.

3 THE MODEL

In this section we introduce *neural Markov logic networks* (NMLNs), an exponential-family model for relational data that is based on potential functions represented by neural networks.

3.1 NEURAL MARKOV LOGIC NETWORKS

We need two classes of potential functions: fragment potentials and global potentials, which are defined on fragments and possible worlds, respectively.

Definition 2 (Fragment Potential)**.** *Given a first-order logic language $\mathscr{L}$, a fragment potential function ϕ is any parametric function $\phi(\gamma; \mathbf{w}, \mathbf{W}_e)$ from $\Gamma_k(\mathscr{L})$ to $\mathbb{R}$ with parameter vectors $\mathbf{w}$ and $\mathbf{W}_e$.*

We explain the role of the parameter vectors $\mathbf{w}$ and $\mathbf{W}_e$ of fragment potential functions farther in the paper. For now, they are just some parameters.

Definition 3 (Global potential)**.** *Given a parametric fragment potential function $\phi(\gamma; \mathbf{w}, \mathbf{W}_e)$, we define the parametric global potential function:*

$$\Phi(\omega; \mathbf{w}, \mathbf{W}_e) = \frac{1}{|\Gamma_k(\omega)|} \sum_{\gamma \in \Gamma_k(\omega)} \phi(\gamma; \mathbf{w}, \mathbf{W}_e).$$

With these two definition we can now introduce neural Markov logic networks.

Definition 4 (Neural Markov Logic Network)**.** *Given a set of fragment potential functions $\phi_1, \ldots, \phi_m$, and the respective global potential functions $\Phi_1, \ldots, \Phi_m$, a neural Markov logic network (NMLN) is the parametric exponential-family distribution over possible worlds from a given $\Omega_{\mathscr{L}}$:*

$$P(\omega) = \frac{1}{Z} \exp\left(\sum_i \beta_i \Phi_i(\omega; \mathbf{w}_i, \mathbf{W}_e)\right),$$

where β_i, $\mathbf{w}_i$ and $\mathbf{W}_e$ are parameters and $Z = \sum_{\omega \in \Omega_{\mathscr{L}}} \exp\left(\sum_i \beta_i \Phi_i(\omega; \mathbf{w}_i, \mathbf{W}_e)\right)$ is the normalization constant (partition function).

Neural Markov logic networks are a fairly standard exponential-family model (their main strength lies in the flexibility of their potential functions which we explain in the next section), so one can rely on standard maximum-likelihood estimation to learn them from data.

Learning When given only one training example $\widehat{\omega}$, NMLNs can be learned by maximizing the log-likelihood:

$$\max_{\mathbf{w}_i, \mathbf{W}_e, \beta_i} \left\{ \sum_{i=1}^{m} \beta_i \Phi_i(\widehat{\omega}; \mathbf{w}_i, \mathbf{W}_e) - \log Z \right\}. \tag{1}$$

The maximization of the log-likelihood is carried out by a gradient-based method (see Appendix **??**). When multiple training examples on domains of the same sizes are available then the maximum likelihood generalizes straightforwardly. For the more general case where domains of training and test data differ, it is more natural to view the learning problem as min-max entropy optimization. We discuss this in more detail in Appendix **??**.

Expressivity The potential functions in NMLNs play similar role as rules in classical MLNs. In fact, as we show in Appendix B, any MLN without existential quantifiers can be straightforwardly represented as an NMLN.[1] Standard first-order logic rules could be added simply as other potentials to NMLNs.

[1] By using max-pooling in the definition of global potentials, one can obtain even richer class of NMLN models that can represent any "Quantified MLN" [Gutiérrez-Basulto et al., 2018]. These models are not considered in this paper.

4 RELATIONAL POTENTIALS

In this section we delve into details of fragment potential functions, their properties and their representation.

4.1 SYMMETRIC FRAGMENT POTENTIALS

We start by introducing symmetric fragment potentials which are fragment potential function that treat any two fragments isomorphic to each other in the same way. Symmetric potentials are useful to model relational structures, such as molecules, where the same molecule may be represented by many isomorphic relational structures. Symmetric fragment potentials can be seen as analogical to formulas in MLNs that do not contain constant symbols. For instance, the constant-free formula $sm(x) \wedge fr(x,y) \Rightarrow sm(y)$, which can be used in an MLN, applies to all domain elements in the same way. Likewise, a symmetric fragment potential should not distinguish structures that are isomorphic, i.e. differ just by renaming of constants, such as $(\{Alice, Bob\}, \{fr(Alice, Bob), sm(Alice)\})$ and $(\{Alice, Eve\}, \{fr(Alice, Eve), sm(Alice)\})$. Next we formalize this intuition and define symmetric fragment potentials.

Definition 5 (Symmetric fragment potentials)**.** *Two fragments γ and γ' are* isomorphic *if γ can be obtained from γ' by renaming (some of) the constants (renaming here refers to an injective mapping). A fragment potential function ϕ is symmetric if $\phi(\gamma, \mathbf{w}) = \phi(\gamma', \mathbf{w})$ whenever γ and γ' are isomorphic.*

Note that symmetric fragment potentials do not depend on the parameter vector $\mathbf{W}_e$.

If the fragment potential ϕ is symmetric then the global potential Φ must be symmetric as well, i.e. if ω and ω' are isomorphic possible worlds then $\Phi(\omega; \mathbf{w}) = \Phi(\omega'; \mathbf{w})$. Hence, an NMLN is symmetric if its potential functions are symmetric. A symmetric NMLN gives the same probability to any two isomorphic worlds.

4.1.1 Representing Symmetric Fragment Potentials

Once we have defined symmetric fragment potentials, we still need to represent them. To this end, we use the concept of *fragment anonymization*. Given a fragment $\gamma = (\mathscr{S}, \mathscr{B})$, its anonymizations $\text{Anon}(\gamma)$ is a list of $|\mathscr{S}|!$ fragments obtained as follows. First, we construct the set of all bijective mappings from the set $\mathscr{S}$ to $\{1, 2, \ldots, |\mathscr{S}|\}$. This is the set of *anonymization functions* and we denote it by $\text{AnonF}(\gamma)$. Then, we obtain each of the elements of $\text{Anon}(\gamma)$ by taking one function from $\text{AnonF}(\gamma)$ and applying it to γ. Note that $\text{Anon}(\gamma)$ may contain several identical elements.

Example 4.1. Consider again the fragment $\gamma = (\mathscr{S}, \{sm(Alice), fr(Alice, Bob)\})$. The set of anonymization functions is $\text{AnonF}(\gamma) = \{\{Alice \mapsto 1, Bob \mapsto 2\}, \{Alice \mapsto 2, Bob \mapsto 1\}\}$ and the respective list of anonymizations is then the list: $\text{Anon}(\gamma) = (\gamma', \gamma'')$ where $\gamma' = (\{1,2\}, \{sm(1), fr(1,2)\})$ and $\gamma'' = (\{1,2\}, \{sm(2), fr(2,1)\})$.

We use anonymizations to define symmetric fragment potentials starting from not necessarily symmetric functions. Specifically, for a given function $\phi'(\gamma; \mathbf{w})$ on fragments over the language $\mathscr{L}_0$ with $\mathscr{C}_{\mathscr{L}_0} = \{1, 2, \ldots, k\}$, we define the symmetric fragment potential as

$$\phi(\gamma; \mathbf{w}) = \sum_{\gamma' \in \text{Anon}(\gamma)} \phi'(\gamma'; \mathbf{w}), \tag{2}$$

Clearly, any potential computed as above must be symmetric (and, vice versa, any symmetric potential can be represented in this way).

4.1.2 Neural Net Representations of Potentials

Anonymizations of fragments can also be represented using binary vectors. All possible ground atoms that we can construct from the available relations (from $\mathscr{R}_{\mathscr{L}}$) and the constants from $\{1, 2, \ldots, |\mathscr{S}|\}$ can be ordered (e.g. lexicographically) and then used to define the binary-vector representation.

Example 4.2. Let $\mathscr{R}_{\mathscr{L}} = \{sm(x), fr(x,y)\}$. Consider the fragment γ from Example 4.1. If we order the possible ground atoms lexicographically as: $fr(1,1)$, $fr(1,2)$, $fr(2,1)$, $fr(2,2)$, $sm(1)$, $sm(2)$, its two anonymizations can be represented by the binary vectors $(0,1,0,0,1,0)$ and $(0,0,1,0,0,1)$.

From now on we will treat anonymizations and their binary-vector representations interchangeably as long as there is no risk of confusion. Representing anonymizations as binary vectors allows us to represent the functions ϕ' above using standard *feedforward neural networks*; the parameters $\mathbf{w}$ are then the weights of the neural network. These networks take, as input, a binary-vector representation of the current anonymization and return a real value as output. When functions ϕ' are represented as neural networks, Equation 2 is actually defining a sharing scheme of the weights for the fragment potential ϕ. This scheme is imposing, by construction, an invariance property w.r.t. isomorphisms of fragments.

Relation to CNNs One can get a nice intuition about the properties of this class of functions when comparing them with Convolutional Neural Networks (CNN). While a CNN computes the same set of features for an input and its spatial translation (i.e. translation invariance), a symmetric fragment potential computes the same set of features for symmetric fragments.

4.2 GENERAL FRAGMENT POTENTIALS

Now we explain how to represent general non-symmetric potentials that will allow us to learn vector-space embeddings of constants from the domain, which is also a key feature of many existing transductive models, like NTP [Rocktäschel and Riedel, 2017]. In what follows $\mathbf{W}_e$ will denote the embedding parameters and $\mathbf{W}_e(c_1,\ldots,c_k)$ will denote the concatentation of the embedding vectors of c_1, $\ldots$, c_k.

Consider a potential $\phi'(\gamma;\mathbf{w},\mathbf{W}_e)$ on fragments over the language $\mathscr{L}_0$ with $\mathscr{C}_{\mathscr{L}_0} = \{1,2,\ldots,k\}$ where $\mathbf{w}$ and $\mathbf{w}_e$ are some parameter vectors. As was the case for the symmetric potentials, using the binary-vector representation of fragments in $\mathscr{L}_0$, ϕ' can be represented, for instance, as a feedforward neural network. We can then write the general potential function as

$$\phi(\gamma;\mathbf{w},\mathbf{W}_e) = \sum_{\pi\in\text{AnonF}(\gamma)} \phi'(\pi(\gamma);\mathbf{w},\mathbf{W}_e(\pi^{-1}(1),\ldots,\pi^{-1}(k)))$$

Again it is not difficult to show that any symmetric or non-symmetric fragment potential function can be represented in this way. We may notice that when $\mathbf{W}_e(c)$ gives the same vector for all constants, the potential will also be symmetric, which is not the case in general. As we show in Section 6.2, the addition of embedding of constants helps improving the prediction capability of our model in transductive settings.

5 INFERENCE

We use Gibbs Sampling (GS) for inference in NMLNs. Gibbs Sampling requires a large number of steps before converging to the target distribution. However, when we use it for learning inside SGD to approximate gradients (Appendix **??**), we run it only for a limited number of steps [Hinton, 2002].

Handling Determinism Gibbs sampling cannot effectively handle distributions with determinism. In normal Markov logic networks, sampling from such distributions may be tackled by an algorithm called MC-SAT [Poon and Domingos, 2006]. However, MC-SAT requires an explicit logical encoding of the deterministic constraints, which is not available in NMLNs where deterministic constraints are implicitly encoded by the potential functions.[2] Our solution is to simply avoid learning distributions with determinism by adding noise during training. We set a parameter $\pi_n \in [0,1]$ and, at the beginning of each training epoch, each ground atom of the input (training) possible worlds is inverted with probability π_n. This added noise also prevents the model from perfectly fitting training data, acting as a regularizer [Bishop, 1995].

5.1 FASTER INFERENCE FOR $k \leq 3$

The performance of Gibbs sampling can be improved using the idea of blocking [Jensen et al., 1995] in combination with the massive parallelism available through GPUs. We describe such a method for NMLNs with fragments of size $k \leq 3$. For simplicity we assume that all the relations are unary or binary (although it is not difficult to generalize the method to higher arities). The description below applies to one pass of Gibbs sampling over all possible ground atoms.

Case $k = 1$: This is the most trivial case. Any two atoms $U(c)$, $U'(c')$ and $R(c,c')$ are independent when $c \neq c'$, where U, U' and R are some relations (in fact, all $R(c,c')$ where $c \neq c'$ have probability 0.5 in this case). Hence, we can run in parallel one GS for each domain element c – each of these parallel GS runs over the unary atoms of the form $U(c)$ and reflexive binary atoms $R(c,c)$ for the given constant c over all relations.

Case $k = 2$: The sets of unary and reflexive atoms that were independent for $k = 1$ are no longer independent. Therefore we first run GS sequentially for all unary and reflexive atoms for the $k = 2$ case. However, conditioned on these unary and reflexive binary atoms, the atoms in any collection $R_1(c_1,c'_1)$, $\ldots$, $R_n(c_n,c'_n)$ are independent provided $\{c_i,c'_i\} \neq \{c_j,c'_j\}$ for all $i \neq j$. Therefore we can now create one GS chain for every 2-fragment and run these GS in parallel.[3] We note that similar ideas were exploited in lifted sampling algorithms [Venugopal and Gogate, 2012].

Case $k = 3$: We first sample unary and reflexive atoms as we did for $k = 2$. Conditioned on these, the atoms in any collection $R_1(c_1,c'_1)$, $\ldots$, $R_n(c_n,c'_n)$ are independent provided $\{c_i,c'_i\} \cap \{c_j,c'_j\} = \emptyset$ for all $i \neq j$ (compare this to $k = 2$). This gives us a recipe for selecting atoms that can be sampled in parallel. First, we construct a complete graph of size n and identify the constants from the domain with its vertices. We then find an edge-coloring of this graph with the minimum number of colors. When n is even then $n-1$ colors are sufficient, when it is odd then we need n colors (finding the coloring is trivial). We then partition the set of pairs of constants based on the colors of their respective

[2] In fact, only constraints that are almost deterministic, i.e. having very large weights, can occur in NMLNs but, at least for Gibbs sampling, the effect is the same. Such distributions would naturally be learned in our framework on most datasets.

[3] We can further increase scalability of NMLNs for $k = 2$ for transductive-learning problems such as knowledge graph completion by exploiting *negative-based sampling*. Here, when using GS to estimate the gradients while training, instead of running it on all pairs of constants, we use only those pairs for which there are at least some relations and only a sub-sample of the rest of pairs (and estimate gradients, accounting for the subsampling rate).

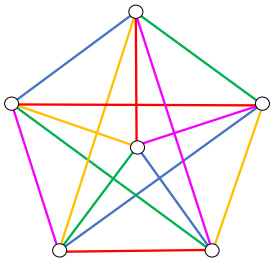


Figure 1: **Faster Inference: k=3.** Partitioning of the atoms according to a edge-coloring scheme. Nodes represent constants, while edges represent binary atoms. Constants connected by edges of the same colour belongs to independent fragments, which can be sampled in parallel.

edges. GS can be run in parallel for atoms in the different 2-fragments corresponding to edges that have the same color (see Figure 1), conditioned on the unary and reflexive binary atoms. This brings an $O(n)$ speed-up (if the parallel GS chains fit in the GPU).[4]

6 EXPERIMENTS

In this section, we report experiments done with NMLNs on three diverse tasks: on the one hand molecular (graph) generation and, on the other, knowledge base completion and triple classification. The aim of these experiments is to show that NMLNs can be used as an *out-of-the-box tool for statistical relational learning*.

We implemented[5] neural Markov logic networks in Tensorflow. In Appendix **??**, we provide detailed information about neural network architectures, hyperparameters grids and selected hyperparameters .

6.1 GRAPH GENERATION

By modeling the joint probability distribution of a relational structure and by learning the potentials as neural networks, NMLNs are a valid candidate for generative tasks in non-euclidean settings, which are receiving an increasing interest recently [You et al., 2018, Li et al., 2018]. To generate a set of relational structures, we can just collect samples generated by Gibbs sampling during training of an NMLN and return top-n most frequently occurring ones (or, alternatively, top n from the last N sampled ones to allow time for NMLNs to converge). In this section, we describe a molecule generation task. We used as training data the ChEMBL molecule database [Gaulton et al., 2016]. We restricted the dataset to molecules with 8 heavy atoms (with a total of 1073 training molecules). We used the RDKit framework [6] to get a FOL representation of the molecules from their SMILES encoding. We show a more detailed description of the training data and generation setting in Appendix **??**.

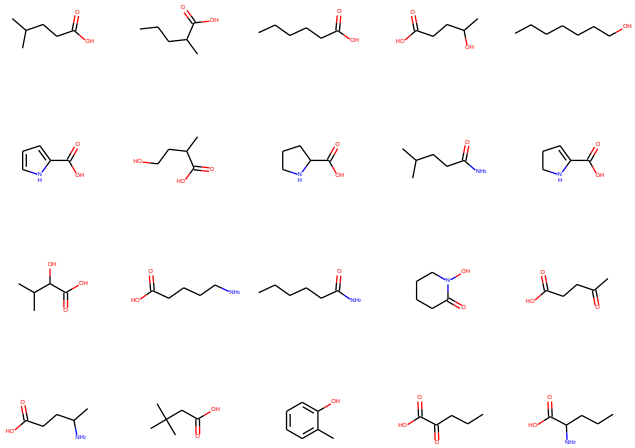


Figure 2: **Molecules generation.** A sample of generated molecules by a NMLN.

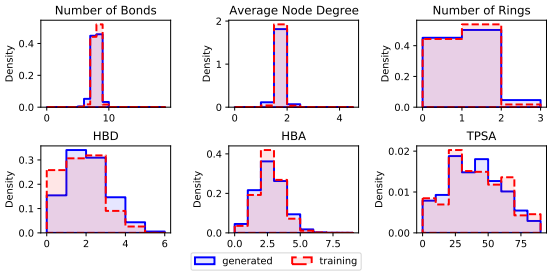


Figure 3: **Molecules generation.** Comparison of chemical properties of generated and real molecules.

In Figure 2, we show the set of top-20 sampled molecules. The first three molecules turn out to be isomers of *hexanoic acid*, the fourth is known as *4-hydroxyvaleric acid*, the fifth is the alcohol called *heptanol* etc.

Furthermore, in Figure 3, we compare the normalized count of some statistics on the training and generated molecules, as it has been recently done in Li et al. [2018]. These statistics represent both general structural properties as well as chemical functional properties of molecules.

In order to measure the generalization and novelty of the approach, we counted how many of the most frequently generated molecules are contained in the training set. We use this knowledge as an indication of the generalization

[4] Another speed-up for transductive problems such as knowledge-graph completion, where we only care about ranking individual atoms by their marginal probabilities, can be achieved as follows. We sample subsets of the domain of size $m < n$. For each of the samples we learn an NMLN and predict marginal probabilities of the respective atoms using Gibbs sampling. At the end, we average the marginal probabilites.

[5] `https://github.com/GiuseppeMarra/nmln/tree/uai2021`

[6] `https://rdkit.org/`

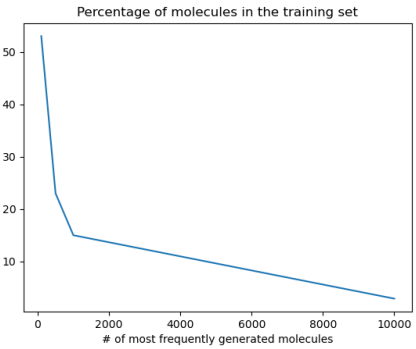


Figure 4: **Molecules generation.** Percentage of generated molecules in the training set when increasing the number of generated molecules.

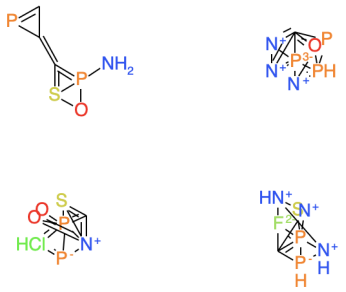


Figure 5: **Molecules generation.** A sample of generated molecules by a standard MLN.

capability of the model in terms of the novelty (w.r.t. the training set) of the generated molecules. The results are shown in Figure 4. It is shown that 53 out of the 100 most frequently generated molecules are indeed in the training set, which is a good sign, because we expect training molecules to be very likely. However, for larger number of generated molecules, NMLNs generate lots of new molecules, that have never been observed during training process.

However, these numbers still do not tell us whether the novel generated molecules (i.e. the ones not in the trainig set) are meaningful or not. Since we reject generated chemically invalid molecules during training (since they can be just checked with an automatic tool like RDkit, we get a method reminiscent of rejection sampling), all the generated molecules are chemically valid and thus we check whether they are known in larger databases than the one we used for training. We selected 100[7] of the novel generated molecules and we checked in the ChemSpider dataset[8] if they are indexed or not. **83** out of 100 molecules are indeed existing molecules, which are shown in Table **??** of the supplemental material. We still don't know if the remaining 17 molecules are simply not indexed in ChemSpider, are impossible for more complex chemical reasons (not checkable with RDkit) or they just represent completely novel molecules. However, it is rather impressive that most of these generated molecules actually exist and were not present in the training data.

Comparison with MLN. The strength of NMLN in generative tasks can be also evaluated when compared with the same task solved in a symbolic way using Markov Logic Network. We could expect that in order to encode the distribution of molecules, a MLN would need a prohibitively large number of rules. On the contrary, an approximation of such rules can be represented rather compactly in the neural potentials of NMLNs. We learned the structure of a MLN using Alchemy[9]. The setting of the experiment and the learned rules are available in Appendix **??**. As expected, MLNs were not able to learn a set of rules capable of correctly encoding the distribution. In fact, all the sampled molecules were rejected by RDkit, because they always violate the maximum admissible number of bonds of at least one atom of the molecule (e.g. a carbon atom with 6 bonds). We show some of the generated "molecules" in Figure 5. It is evident how there are too many non-carbon atoms w.r.t. training molecules. Moreover, the molecules are too densely connected.

6.2 KNOWLEDGE BASE COMPLETION

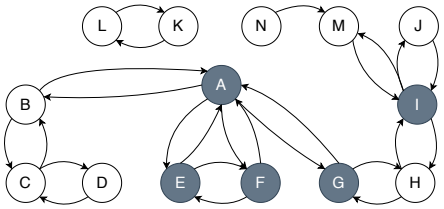
In Knowledge Base Completion (KBC), we are provided with an incomplete KB and asked to complete the missing part. The KBC task is inherently in the transductive setting and the data are provided in a positive-only fashion: we cannot distinguish between unknown and false facts. Kuželka and Davis [2019] studied KBC tasks under the missing-completely-at-random assumption and showed consistency of learning MLNs by maximum-likelihood where both missing and false facts are treated in the same way as *false*. Their arguments can be modified to give similar consistency guarantees for NMLNs.

Smokers. The "Smokers" dataset [Richardson and Domingos, 2006] is a classical example in statistical relational learning literature. Here, two relations are defined on a set of constants representing people: the unary predicate *smokes* identifies those people who smoke, while the binary predicate *friendOf* indicates that two people are friends. This dataset is often used to show how a statistical relational learning algorithm can model a distribution by finding a correlation of smoking habits of friends. For example, in MLNs, one typically uses weighted logical rules such as: $\forall x \, \forall y \, \texttt{friendOf}(x,y) \rightarrow \texttt{smokes}(x) \leftrightarrow \texttt{smokes}(y)$. We trained a NMLN on the small smokers dataset. Since no

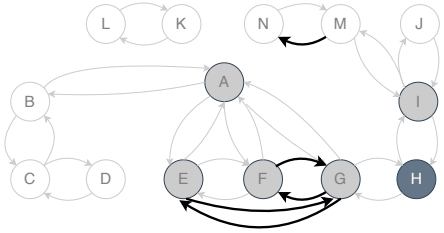
[7] Chemical databases limit the queries one can do for free.

[8] `http://www.chemspider.com`

[9] `https://alchemy.cs.washington.edu/`



(a) The training KB.



(b) The completed KB.

Figure 6: **Knowledge Base Completion in the Smokers dataset.** A grey circle means that the predicate `smokes` is *True* and a white one that it is unknown. Links represent the relation `friendOf`. The given world is shown in 6a and the completed one in 6b. The NMLN learnt that `friendOf` is symmetric, that a friend of at least two smokers is also a smoker, and that two smokers, who are friends of the same person, are also friends.

prior knowledge about the type of rules that are relevant was used by NMLNs, the model itself had to identify which statistics are mostly informative of the provided data by learning the potential functions. We started here with the Smokers dataset in order to (i) illustrate the Knowledge Base Completion task and (ii) to provide some basic intuitions about what kind of rules the model could have learned. In Figure 6, we show the KB before and after completion. In Figure 6b, we highlight only new facts whose marginal probability after training is significantly higher than the others.

Nations, Kinship and UMLS We use the Nations, Kinship and Unified Medical Language System (UMLS) KBs from [Kok and Domingos, 2007]. Nations contains 56 binary predicates, 14 constants and 2565 true facts, Kinship contains 26 predicates, 104 constants and 10686 true facts, and UMLS contains 49 predicates, 135 constants and 6529 true facts. These datasets have been exploited to test KBC performances in Neural Theorem Provers [Rocktäschel and Riedel, 2017]. Greedy-NTPs [Minervini et al., 2020a] and CTP [Minervini et al., 2020b] were recently introduced and the authors showed that their models were able to outperform original NTPs as well as other models proposed in the literature for tackling KBC tasks. In this section, we show how we can use NMLNs to tackle a KBC task on the Nations, Kinship and UMLS datasets.

We followed the evaluation procedure in [Minervini et al., 2020a]. In particular, we took a test fact and corrupted its first and second argument in all possible ways such that the corrupted fact is not in the original KB. Subsequently, we predicted a ranking of every test fact and its corruptions to calculate MRR and HITS@m (H@m). The ranking is based on marginal probabilities estimated by running Gibbs sampling on the NMLN. We compare the original Neural Therem Prover (*NTP*) model [Rocktäschel and Riedel, 2017], Greedy NTP (*G-NTP*) [Minervini et al., 2020a], CTP [Minervini et al., 2020b], *NMLNs* with $k = 3$ and no embeddings (*K3*) and NMLNs with $k = 2$ and embeddings of domain elements (*K2E*). In Table 1, we report the results of the KBC task. NMLN *K2E* outperforms competitors by a large gap on almost all datasets and metrics. Moreover, we can make two observations. Embeddings seem to play a fundamental role in the sparser datasets (i.e. Kinship and UMLS), where the relational knowledge is limited. However, both on Nations and Kinship, *NMLN-K3* still performs better than differentiable provers, even if it cannot exploit embeddings to perform reasoning and it has to rely only on the relational structure of fragments to make predictions. This is a clear signal that, in many cases, the relational structure already contains a lot of information and that NMLNs are better in modeling and exploiting these relational regularities.

Table 1: MRR and HITS@m on Nations, Kinships and UMLS. *K3* and *K2E* are two variants of neural Markov logic networks.

Dataset	Metric	Model				
		NTP	**GNTP**	**CTP**	**K3**	**K2E**
Nations	MRR	0.61	0.65	0.70	**0.81**	0.78
	H@1	0.45	0.49	0.56	**0.71**	0.66
	H@3	0.73	0.78	0.81	**0.89**	0.86
	H@10	0.87	0.98	0.99	0.98	**0.99**
Kinship	MRR	0.35	0.75	0.76	0.82	**0.84**
	H@1	0.24	0.64	0.64	0.73	**0.76**
	H@3	0.37	0.85	0.85	0.88	**0.90**
	H@10	0.57	0.95	0.95	0.96	**0.97**
UMLS	MRR	0.80	0.85	0.85	0.50	**0.92**
	H@1	0.70	0.76	0.75	0.39	**0.89**
	H@3	0.88	**0.94**	**0.94**	0.54	**0.94**
	H@10	0.95	**0.98**	0.98	0.71	0.97

6.3 TRIPLE CLASSIFICATION

In triple classification, one is asked to predict if a given triple belongs or not to the knowledge base. Even though there exists an entire class of methods specifically developed for this task, we wanted to show that our method is general enough to be also applicable to this large scale problems. We performed experiments on WordNet [Miller, 1995]

and FreeBase [Bollacker et al., 2008], which are standard benchmarks for large knowledge graph reasoning tasks. We used the splits WN11 and FB13 provided in Socher et al. [2013]. Interestingly, NMLN with $k = 2$ achieves an accuracy of $(74.4, 84.7)$ in WN11 and FB13, respectively. This compares similarly or favourably w.r.t. standard knowledge graph embeddings methods, like SE [Bordes et al., 2011] $(53.0, 75.2)$ or TransE [Bordes et al., 2012]$(75.9, 81.5)$. However, it is outperformed by newer methods, like TransD [Ji et al., 2015] $(86.4, 89.1)$ and DistMult-HRS [Zhang et al., 2018] $(88.9, 89.0)$. This is likely due to the fact that these two datasets are extremely sparse and very few pairs of constants are related by more that one relation. Unlike knowledge graph embedding methods, which are tuned for the specific task of predicting head or tail entity of a triple, our model is general and learns a joint probability distribution. The fact that it can still perform similarly to state-of-the-art methods on this specialized task is in fact rather surprising.

7 RELATED WORK

NMLNs as SRL NMLNs are an SRL framework inspired by Markov Logic Networks [Richardson and Domingos, 2006]. From certain perspective, NMLNs can be seen as MLNs in which first-order logic rules are replaced by neural networks (for an explicit mapping from MLNs to NMLNs, refer to Appendix **??**). While this may seem as a rather small modification, the gradient based learning of NMLNs' potentials allows more efficient learning than the usual combinatorial structure-learning.

An alternative approach to improve performance of structure learning in MLNs is represented by the gradient boosted MLNs [Khot et al., 2015]. However, as also noted in [Khot et al., 2015], the boosting approach has not been extended to the generative learning setting where one optimizes likelihood rather than pseudo-likelihood. In contrast, NMLNs are generative and trained by optimizing likelihood. Finally, unlike NMLNs, standard MLNs do not support embeddings of domain elements.

NMLNs as NeSy NMLNs integrate logical representations with neural computation, which is the domain of interest of Neural Symbolic Artificial Intelligence – NeSy [Besold et al., 2017, De Raedt et al., 2020]. Lippi and Frasconi [2009] was an early attempt to integrate MLNs with neural components. Here, an MLN was exploited to describe a conditional distribution over ground atoms, given some features of the constants. In particular, the MLN was reparametrized by a neural network evaluated on input features. A similar approach is the one in Marra et al. [2019], where a continuous relaxation of the logical potentials allows for a faster inference in specific tasks. A related approach in the domain of logic programming is provided in Manhaeve et al. [2018], where the probabilistic logic programming language ProbLog [De Raedt et al., 2007] is extended to allow probabilities of atoms to be predicted by neural networks and to exploit differentiable algebraic generalizations of decision diagrams to train these networks. A common pattern in these approaches is that neural networks are "simply" exploited to parameterize a known relational model. Compared to NMLN, they still rely on combinatorial structure learning (or rules hand-crafted by experts). Recently, there has been a renaissance of ILP methods in which neural computing is used to improve the search process [Ellis et al., 2018, Rocktäschel and Riedel, 2017, Minervini et al., 2020a, Sourek et al., 2018]. These typically use templates or program sketches to reduce the size of the search space and gradient based methods to guide the search process. Unlike NMLN, none of these systems can model joint probability distributions over relational structures.

NMLNs as KGE NMLNs are also related to the many different knowledge graph embedding methods [Wang et al., 2017] as they can also exploit embeddings of domain elements and (implicitly) also relations. NMLNs and KGEs are most similar when $k = 2$ and there are no unary and reflexive binary atoms. In this case, the NMLN still explicitly models the probabilistic relationship between different relations on the same pairs of constants, which KGE methods do not capture explicitly. Moreover, KGE methods cannot model unary relations (i.e. attributes) of the entities. Somewhat surprisingly, as noted by Kazemi and Poole [2018], this problem is less well understood than link-prediction in KGEs. Furthermore, NMLNs can naturally incorporate both models of attributes and links, as demonstrated, e.g., on the small Smokers dataset and in the molecular generation experiment. Moreover, if we properly fix the neural architecture of NMLNs, many existing KGE methods can be explicitly modelled as NMLNs. So NMLNs are more flexible and can solve tasks that KGE methods cannot. On the other hand, KGEs are very fast and optimized for the KG completion tasks, as we also observed in our experiments (cf Section 6.3).

8 CONCLUSIONS

We have introduced neural Markov logic networks, a statistical relational learning model combining representation learning power of neural networks with principled handling of uncertainty in the maximum-entropy framework. The proposed model works remarkably well both in small and large domains despite the fact that it actually solves a much harder problem (modelling joint probability distributions) than specialized models such as various knowledge graph embedding methods.

Acknowledgements

This research has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No [694980] SYNTH: Synthesising Inductive Data Models)", the KU Leuven Research Fund (C14/18/062) and the Research Foundation - Flanders (G097720N). OK was supported by Czech Science Foundation project "Generative Relational Models" (20-19104Y). Part of this work was done before and was supported by the OP VVV project *CZ.02.1.01/0.0/0.0/16_019/0000765* "Research Center for Informatics".

References

Tarek R Besold, Artur d'Avila Garcez, Sebastian Bader, Howard Bowman, Pedro Domingos, Pascal Hitzler, Kai-Uwe Kühnberger, Luis C Lamb, Daniel Lowd, Priscila Machado Vieira Lima, et al. Neural-symbolic learning and reasoning: A survey and interpretation. *arXiv preprint arXiv:1711.03902*, 2017.

Chris M Bishop. Training with noise is equivalent to tikhonov regularization. *Neural computation*, 7(1):108–116, 1995.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250, 2008.

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. Learning structured embeddings of knowledge bases. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. Joint learning of words and meaning representations for open-text semantic parsing. In *Artificial Intelligence and Statistics*, pages 127–135, 2012.

Luc De Raedt, Angelika Kimmig, and Hannu Toivonen. Problog: A probabilistic prolog and its application in link discovery. In *IJCAI*, volume 7, pages 2462–2467. Hyderabad, 2007.

Luc De Raedt, Sebastijan Dumančić, Robin Manhaeve, and Giuseppe Marra. From statistical relational to neural symbolic artificial intelligence. In *IJCAI 2020*, 2020.

Kevin Ellis, Lucas Morales, Mathias Sablé-Meyer, Armando Solar-Lezama, and Josh Tenenbaum. Learning libraries of subroutines for neurally-guided bayesian program induction. In *NeurIPS*, 2018.

Anna Gaulton, Anne Hersey, Michał Nowotka, A Patrícia Bento, Jon Chambers, David Mendez, Prudence Mutowo, Francis Atkinson, Louisa J Bellis, Elena Cibrián-Uhalte, et al. The chembl database in 2017. *Nucleic acids research*, 45(D1):D945–D954, 2016.

Víctor Gutiérrez-Basulto, Jean Christoph Jung, and Ondrej Kuzelka. Quantified markov logic networks. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixteenth International Conference, KR 2018, Tempe, Arizona, 30 October - 2 November 2018.*, pages 602–612, 2018.

Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14 (8):1771–1800, 2002.

Claus S Jensen, Uffe Kjærulff, and Augustine Kong. Blocking gibbs sampling in very large probabilistic expert systems. *International Journal of Human-Computer Studies*, 42(6):647–666, 1995.

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)*, pages 687–696, 2015.

Seyed Mehran Kazemi and David Poole. Relnn: A deep neural model for relational learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18)*, pages 6367–6375, 2018.

Tushar Khot, Sriraam Natarajan, Kristian Kersting, and Jude Shavlik. Gradient-based boosting for statistical relational learning: the markov logic network and missing data cases. *Machine Learning*, 100(1):75–100, 2015.

Stanley Kok and Pedro Domingos. Statistical predicate invention. In *Proceedings of the 24th international conference on Machine learning*, pages 433–440. ACM, 2007.

Ondřej Kuželka and Jesse Davis. Markov logic networks for knowledge base completion: A theoretical analysis under the MCAR assumption. In *Proceedings of the Thirty-Fifth Conference on Uncertainty in Artificial Intelligence, UAI*, 2019.

Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*, 2018.

Marco Lippi and Paolo Frasconi. Prediction of protein β-residue contacts by markov logic networks with grounding-specific weights. *Bioinformatics*, 25(18):2326–2333, 2009.

Robin Manhaeve, Sebastijan Dumančić, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Deepproblog: Neural probabilistic logic programming. In *NeurIPS*, 2018.

Giuseppe Marra, Francesco Giannini, Michelangelo Diligenti, and Marco Gori. Integrating learning and reasoning with deep logic models. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 517–532. Springer, 2019.

George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

Pasquale Minervini, Matko Bošnjak, Tim Rocktäschel, Sebastian Riedel, and Edward Grefenstette. Differentiable reasoning on large knowledge bases and natural language. In *AAAI*, 2020a.

Pasquale Minervini, Sebastian Riedel, Pontus Stenetorp, Edward Grefenstette, and Tim Rocktäschel. Learning reasoning strategies in end-to-end differentiable proving. *arXiv preprint arXiv:2007.06477*, 2020b.

Hoifung Poon and Pedro Domingos. Sound and efficient inference with probabilistic and deterministic dependencies. In *AAAI*, volume 6, pages 458–463, 2006.

Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine learning*, 62(1-2):107–136, 2006.

Tim Rocktäschel and Sebastian Riedel. End-to-end differentiable proving. In *Advances in Neural Information Processing Systems*, pages 3788–3800, 2017.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems*, pages 926–934, 2013.

Gustav Sourek, Vojtech Aschenbrenner, Filip Zelezný, Steven Schockaert, and Ondrej Kuzelka. Lifted relational neural networks: Efficient learning of latent relational structures. *J. Artif. Intell. Res.*, 62:69–100, 2018.

Deepak Venugopal and Vibhav Gogate. On lifting the gibbs sampling algorithm. In *NeurIPS*, 2012.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, 2017.

Jiaxuan You, Rex Ying, Xiang Ren, William L Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. *arXiv preprint arXiv:1802.08773*, 2018.

Zhao Zhang, Fuzhen Zhuang, Meng Qu, Fen Lin, and Qing He. Knowledge graph embedding with hierarchical relation structure. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3198–3207, 2018.