The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)

On Lifted Inference Using Neural Embeddings

Mohammad Maminur Islam,[1] **Somdeb Sarkhel,**[2] **Deepak Venugopal**[1]
[1]The University of Memphis [2]Adobe Research
mislam3@memphis.edu, sarkhel@adobe.com, dvngopal@memphis.edu.

Abstract

We present a dense representation for Markov Logic Networks (MLNs) called Obj2Vec that encodes symmetries in the MLN structure. Identifying symmetries is a key challenge for lifted inference algorithms and we leverage advances in neural networks to learn symmetries which are hard to specify using hand-crafted features. Specifically, we learn an embedding for MLN objects that predicts the context of an object, i.e., objects that appear along with it in formulas of the MLN, since common contexts indicate symmetry in the distribution. Importantly, our formulation leverages well-known skip-gram models that allow us to learn the embedding efficiently. Finally, to reduce the size of the ground MLN, we sample objects based on their learned embeddings. We integrate Obj2Vec with several inference algorithms, and show the scalability and accuracy of our approach compared to other state-of-the-art methods.

Introduction

Neural embeddings have been extremely successful as a general approach to learn efficient and effective representations for a variety of real-world domains including words (Mikolov et al. 2013; Pennington, Socher, and Manning 2014), knowledge graphs (Nickel et al. 2016), images (Kiela and Bottou 2014), etc. Inspired by these successes, in this paper, we present a novel representation for Markov Logic Networks (MLNs) (Domingos and Lowd 2009) using neural embeddings to represent symmetries in the model. Our main motivation for such a representation stems from the fact that over the last several years, it has been widely recognized that exploiting symmetries in MLNs (and in other statistical relational models such as PSL (Bach et al. 2017)) yields exponential improvements in the scalability of inference algorithms. Thus, several algorithms that are collectively referred to as *lifted inference* (Poole 2003) algorithms have been proposed that exploit symmetries in the MLN.

However, identifying symmetries in the MLN efficiently and effectively is non-trivial. Previous lifted inference methods have developed first-order rules to identify symmetries (de Salvo Braz 2007; Van den Broeck et al. 2011; Gogate and Domingos 2011). However, such rules can identify a relatively small subset of symmetries and are severely limited when the MLN is conditioned on evidence variables (Van den Broeck and Darwiche 2013). Consequently, more recent lifted inference algorithms try to exploit approximate symmetries using matrix factorization for binary evidence (Van den Broeck and Darwiche 2013) or clustering (Venugopal and Gogate 2014). Further, algorithm-specific scalable lifting methods include clustering for MAP (Sarkhel, Singla, and Gogate 2015) and approximate messaging in BP (Singla, Nath, and Domingos 2014). Other lifted inference techniques focus on detecting symmetries using the graph structure of the Markov network underlying the MLN have been proposed. Specifically, Bui et al. (Bui, Huynh, and Riedel 2013) connected automorphism groups in the Markov network structure with lifted variational inference. Similar approaches have been developed for MAP inference using ILPs (Apsel, Kersting, and Mladenov 2014) and marginal inference (Mladenov and Kersting 2015). However, these approaches are essentially "bottom-up" approaches meaning that they work by instantiating the MLN to create the Markov network and then detect symmetries. For practical problems in domains such as information extraction (Venugopal et al. 2014) or question answering (Khot et al. 2015; Venugopal and Rus 2016), creating the Markov network quickly becomes infeasible. Therefore, we develop a generic, scalable approach that learns subsymbolic vector representations for the MLN based on symmetries.

Our main contribution in this paper is Obj2Vec, a distributed representation for objects in the MLN. Specifically, if two objects are symmetrical, they are exchangeable in ground formulas of the MLN. Thus, one possible representation is to vectorize objects using ground formulas and learn a dense embedding from these vectors. However, learning from vectors that directly encode ground formulas is not scalable since the input representation is as big as the ground Markov network. Therefore, inspired by the successful *skip-gram* model, we propose a novel, more scalable approach that creates an embedding based on local context information for objects. Specifically, we train the neural network to predict objects based on surrounding objects in the ground formulas of the MLN. The embedding layer will then learn similar representations for objects that have similar contexts. Using this formulation, we can adapt skip-gram model ar-

chitectures (Mikolov et al. 2013) to perform domain lifting efficiently. To perform tractable inference using the Obj2Vec embedding, we sample from the embedding and create a smaller MLN with fewer *meta-objects* that represent groups of exchangeable objects.

We perform experiments on marginal and MAP inference algorithms implemented in two state-of-the-art systems Tuffy (Niu et al. 2011) and Magician (Venugopal, Sarkhel, and Gogate 2016), and compare our approach with other approaches for lifting, binary matrix factorization (Van den Broeck and Darwiche 2013) and clustering (Venugopal and Gogate 2014). Our results clearly show that our approach is more scalable and accurate for several benchmark inference problems.

Background

Markov Logic Networks

Markov logic networks (MLNs) are template models that define uncertain, relational knowledge as first-order formulas with weights. Weights encode uncertainty in a formula. ∞ weight formulas are hard constraints which should always be true. Similarly, formulas with $-\infty$ weights are always false. Thus, MLNs offer a flexible framework to mix hard and soft formulas. In MLNs, we assume that each argument of each predicate in the MLN is typed and can only be assigned to a finite set of constants. By extension, each logical variable in each formula is also typed. Thus, each variable corresponds to a specific domain of objects. A ground atom is an atom where each of its variables has been substituted by constants from their respective domains. A ground formula is a formula containing only ground atoms.

Given a set of constants that represent the domains of variables in the MLN, an MLN represents a factored probability distribution over the possible worlds, in the form of a Markov network. A world in an MLN is an assignment of 0/1 to all ground atoms of the MLN. Specifically, the distribution is given by,

$$\Pr(\omega) = \frac{1}{Z} \exp\left(\sum_i w_i N_i(\omega)\right) \tag{1}$$

where w_i is the weight of formula f_i, $N_i(\omega)$ is the number of groundings of formula f_i that evaluate to True given a world ω, and Z is the normalization constant. The two common inference problems over MLNs are marginal inference and MAP inference which are both intractable problems.

Skip-Gram Models

Skip-gram models are used to learn an embedding over words based on the context in which they appear in the training data (i.e., nearby words). Word2vec (Mikolov et al. 2013) is a popular model of this type, where we train a neural network based on pairs of words seen in the training data. Specifically, we use a sliding window of a fixed size and generate all pairs of words within that sliding window. Each word is first represented as a one-hot encoded vector. Then, for each pair of words, we consider one word of the pair as input and the other word as a target. That is, we learn to predict a word based on its context or nearby words. The hidden layer typically has a much smaller number of dimensions as compared to the input/output layers. Thus, the hidden-layer learns a low-dimensional embedding that is capable of mapping words to their contexts. Typically the hidden-layer output is used as features for other text processing tasks, as opposed to using hand-crafted features.

Related Work

Lifted inference is the predominant approach to improving the scalability of inference in relational models. Exact lifted inference approaches include (Poole 2003; de Salvo Braz 2007; Gogate and Domingos 2011; Van den Broeck et al. 2011). Approximate inference methods that exploit exact symmetries include (Singla and Domingos 2008; Niepert 2012; Venugopal and Gogate 2012; Sarkhel et al. 2014). Our approach is most closely related to pre-processing approaches that exploit approximate symmetries such as binary evidence processing (Van den Broeck and Darwiche 2013) and clustering methods that use specific count-based features (Venugopal and Gogate 2014). However, our approach is more general since it can be applied to all evidence types, and also does not use hand-coded features. The second line of research in lifted inference identifies symmetries on the Markov network structure using automorphism in graphs (Niepert 2012; Bui, Huynh, and Riedel 2013; Van den Broeck and Niepert 2015). Specifically, color passing algorithms are used to find exchangeable variables for marginal as well as MAP inference (Apsel, Kersting, and Mladenov 2014; Mladenov and Kersting 2015). However, such algorithms compute symmetries on the ground Markov network, and for large practical problems, this becomes infeasible. In contrast to detecting symmetries, Kopp et al. (Kopp, Singla, and Kautz 2015) used symmetry breaking techniques commonly used in the SAT community and applied it to lifted inference. Anand et al. (Anand et al. 2016) developed methods that detect contextual symmetries for probabilistic graphical models. These methods also require the ground Markov network structure in order to adapt them to MLNs.

Finally, recent approaches have been proposed that integrate advances in deep learning with relational models. Also, Rocktaschel and Riedel (Rocktäschel and Riedel 2017) developed subsymbolic representations and learning for logical inference operators. Specifically, they developed vector representations for logical symbols and used them within theorem proving. Our approach can be viewed as developing representations for probabilistic reasoning taking advantage of distributional symmetries. Note that while graph-based embeddings have been proposed for relational data previously (Bordes et al. 2011), in our case, the graph structure is enormous and constructing the ground Markov network is infeasible which is our primary motivation in developing new representations that are scalable to learn and take advantage of symmetries in the model. To the best of our knowledge, ours is the first work to connect lifted inference with neural embeddings.

Obj2Vec

We next describe our model for embedding MLN objects. First, we define what symmetry means for two objects. Note that previous works have defined symmetry in terms of *orbits* in the automorphism groups of variables in the Markov network underlying the MLN (Bui, Huynh, and Riedel 2013). Here, we characterize symmetry based on the exchangeability of objects in ground formulas.

Definition 1. *Given an MLN $\mathcal{M}$ and evidence database $\mathcal{D}$, let X and Y be two objects in the same domain Δ. X and Y are exchangeable (denoted as $X \sim Y$) if we can exchange X and Y in all groundings of the MLN in which they occur without changing the truth assignment of the groundings (according to $\mathcal{D}$).*

The above definition can now be used to define the distance between two objects in a domain as,

Definition 2. *Given an MLN $\mathcal{M}$ and evidence database $\mathcal{D}$, let X and Y be two objects in the same domain Δ. $\delta(X, Y) = \sum_f \mathbb{I}(f, X, Y)$, where $\mathbb{I}(f, X, Y) = 0$ if the truth assignment to f does not change when we exchange X and Y, and 1 otherwise.*

Instead of using a threshold function to represent the difference between mismatches in the ground formula assignment, we now define a continuous approximation of the δ function as a sigmoid function. This allows us to represent approximate symmetries between domain objects. Specifically,

Definition 3. *Given an MLN $\mathcal{M}$ and evidence database $\mathcal{D}$, let X and Y be two objects in the same domain Δ. $\delta(X, Y) = \frac{1}{1+e^{-\beta m}}$, where m is the number of ground formulas that have a difference in assignments before and after exchanging the objects X and Y, and β is a hyper-parameter that controls the shape of the sigmoid function.*

We can now search for an optimal set of approximately exchangeable objects in Δ by finding a subset that minimizes δ. Specifically,

$$\arg \min_{\Delta' \subseteq \Delta} \sum_{X \in \Delta', Y \in \mathcal{N}(X)} \delta(X, Y)$$

where $\mathcal{N}(X)$ is the set of objects in Δ that X exchanges with, and with the constraints that $\cup_X \mathcal{N}(X) = \Delta$ and $|\Delta'| \leq \alpha$, for some constant α.

However, the above approach is computationally infeasible since the number of subsets is exponential. Alternatively, we can cast the above problem as a clustering problem that heuristically find α clusters of approximately exchangeable objects with $\delta(X, Y)$ as the distance function. However, note that even finding a heuristic solution using such a clustering formulation is difficult due to the *curse of dimensionality*. Specifically, as the number of ground formulas increase, the number of clustering dimensions increase accordingly.

To avoid the curse of dimensionality, we instead learn a dense neural representation that yields a more compact, distributed representation for the objects. However, the key challenge here is, *how can we encode or vectorize an object to learn the subsymbolic representation scalably?*. One approach is to design an encoding over the ground formulas. Specifically, the vector encoding for an object X, v_X is a vector that specifies whether a ground formula is `True`/`False`/`unknown`. For groundings where X does not appear, the vector component has a value `unknown`. However, such an encoding leads to an extremely large input layer, since the size of the encoding is equal to the size of the ground Markov network, and is thus not scalable for MLNs. Therefore, we develop a more scalable representation inspired by the *skip-gram* model which is widely used in word embeddings.

Learning Scalable Embeddings

The main idea in our approach is to train a neural network that predicts objects from other objects. Specifically, borrowing terminology from skip-gram models, we seek to predict the *context* of an object. The hidden layer of the neural network learns to represent the input object vectors in a reduced dimension such that objects that appear in similar contexts are placed close together in the embedded space.

Definition 4. *Given an MLN $\mathcal{M}$ and evidence $\mathcal{D}$, suppose f is a ground formula satisfied by $\mathcal{D}$ and where X occurs, the context for object X in f, $\mathcal{C}(f, X)$, is the set of objects (other than X) that occur in f.*

When defining the context, it is important to note the dichotomy between satisfied formulas and unsatisfied formulas. Specifically, we define the context of an object only when the object appears in satisfied ground formulas. In general, there can be ground formulas with an unknown truth value (since not every ground atom is specified as evidence). In such a case, we do not have sufficient information to state whether the context of an object is valid (according to the data) or not. To determine the context of objects in such a ground formula, we first need to infer the most likely truth assignment to the formula, which in turn requires us to perform inference (e.g. MAP), or treat unknown atoms as missing and run an EM algorithm, both of which are computationally expensive. Therefore, we ignore such formulas when defining context of an object. For example, let us assume a simple MLN formula, $\mathtt{R}(x) \wedge \mathtt{S}(x, y)$ with evidence $\mathtt{R}(X_1)$, $\mathtt{S}(X_1, Y_1)$, $\mathtt{R}(X_2)$, $\mathtt{S}(X_2, Y_1)$, $\mathtt{S}(X_3, Y_1)$. In this case, $\mathtt{R}(X_3) \wedge \mathtt{S}(X_3, Y_1)$ has an unknown truth value, so the available data cannot assert that Y_1 is in the context of X_3 for $\mathtt{R}(X_3) \wedge \mathtt{S}(X_3, Y_1)$, but it can assert that Y_1 is in the context of X_1 and X_2 for $\mathtt{R}(X_1) \wedge \mathtt{S}(X_1, Y_1)$ and $\mathtt{R}(X_2) \wedge \mathtt{S}(X_2, Y_1)$ respectively.

Proposition 1. *Suppose $X, Y \in \Delta$ are exchangeable, then X and Y have common contexts.*

Proof. (Sketch) If X and Y are exchangeable, then, to guarantee that the MLN structure does not change, for every formula f where X occurs, we need to find a corresponding formula f' where Y occurs, such that we replace occurrences of X in f to Y, and occurrences of Y in f' to X, and the modified formulas are equivalent to the original formulas given $\mathcal{D}$. To guarantee this, f and f' must share the exact same structure and truth assignment in $\mathcal{D}$, which implies that they have the same contexts. □

Thus, our task in learning the embedding is to ensure that objects with common contexts lie close to each other in the embedding. To do this, we vectorize an object X as a one-hot encoding represented y vector v_X. Specifically, the encoding is as large as the maximum domain size, and each object in the domain is represented as a 0/1 value in this vector. v_X will set the vector component corresponding to X to 1 and the other components to 0. Note that, the size of the input encoding in this case equal to the maximum domain size which is orders of magnitude lesser than the number of ground formulas.

To learn the embedding, we assume a canonical form for all the formulas in the MLN. Specifically, we assume that each formula is a clause, and that there is an ordering over the predicates. That is, in each formula, predicates appear according to a pre-specified ordering. This is needed to give a sequential ordering to the objects in the formulas. To learn the embedding, for every ground formula satisfied by the evidence, we predict an object given the surrounding objects or context. Specifically, let $f_1 \dots f_K$ denote the ground formulas, and let $\mathbf{O}_i$ represent the sequence of objects in f_i, and the o_{ij} represents the j-th object in the i-th formula. The learning algorithm seeks to maximize,

$$\sum_{i=1}^{K} \sum_{j=1}^{|\mathbf{O}_i|} \sum_{-c \le k \le c; k \neq 0} \log P(o_{ij+k} | o_{ij})$$

Specifically, c defines a sliding window-size over the objects in $\mathbf{O}_i$. Here, $P(o_{ij+k}|o_{ij})$ is defined as a softmax function proportional to $\exp(v'^{\top}_{o_{ij+k}} v_{o_{ij}})$, where v_o refers to the input vector representation for object o, and v'_o refers to its output vector representation. An example of specifying the training data to learn the embedding is shown below.

Example 1. *Consider a simple formula* $\mathtt{R}(x) \wedge \mathtt{S}(x, y)$. *Let* $\Delta_x = \{X_1, X_2, X_3\}$ *and* $\Delta_y = \{Y_1, Y_2\}$. *Assume a closed world with the evidence database* $\mathtt{R}(X_1)$, $\mathtt{R}(X_2)$, $\mathtt{R}(X_3)$, $\mathtt{S}(X_1, Y_1)$, $\mathtt{S}(X_2, Y_1)$, $\mathtt{S}(X_3, Y_2)$. *The training instances include,* v_{X_1}, v_{Y_1}; v_{X_2}, v_{Y_1}; v_{X_3}, v_{Y_2};. *That is, given* X_1 *or* X_2 *at the input layer, we predict* Y_1 *at the output layer, and given* X_3 *as input we predict* Y_2 *as the output layer. This means that the hidden layer in the model will derive features such that* X_1 *and* X_2 *will make common predictions at the output layer. At the same time, since* X_3 *has a different context, it needs to predict* Y_2 *at the output layer, and therefore, the hidden layer encoding for* X_3 *will be different from that of* X_1 *and* X_2.

Context in Partially Satisfied Formulas. Defining context of an object only in satisfied ground formulas has limitations when the evidence is very sparse. In such cases, very few ground formulas may be satisfied. For example, consider an MLN $\mathtt{R}(x) \wedge \mathtt{S}(x, y) \wedge \mathtt{T}(y)$, with evidence $\mathtt{R}(X_1)$, $\mathtt{S}(X_1, Y_1)$, $\mathtt{R}(X_2)$, $\mathtt{S}(X_2, Y_1)$, $\mathtt{S}(X_2, Y_2)$. Since none of the ground formulas are satisfied by the evidence, we are unable to detect any symmetries in this case. To account for sparse evidences, we make a *closed world* assumption, where unknown atoms are assumed false in partially satisfied formulas, and then compute the contexts as before.

Representing Joint Symmetries. Our embedding approach implicitly propagates symmetries across domains, since we are learning symmetries jointly over all objects. Specifically, suppose $P(Y_1|X_1) \approx P(Y_2|X_1) \geq \epsilon$, i.e., we can predict Y_1 and Y_2 from X_1 with accuracy ϵ, where $Y_1, Y_2 \in \Delta_1$, and $X_1 \in \Delta_2$. Then, as $\epsilon \to 1$, the embedded vector distance $|v'_{Y_1} - v'_{Y_2}|$ diminishes since a compact embedding will try to represent Y_1 and Y_2 using a similar representation.

Sampling the Embedding

We reduce the size of the ground Markov network by removing objects that are sufficiently close (in the embedding) to an object that we retain in the MLN. Note that one approach to doing this is to formulate a clustering problem using the embedded vectors as features, use K-Means to cluster the objects, and then sample the clusters. However, K-Means is known to get stuck in local optima, and as we observed in our experiments, it often creates imbalanced clusters. This is problematic for MLNs. Specifically, in MLNs, a reduction in the number of objects changes the structure of the underlying Markov network, and thus the distribution changes accordingly. There are some special cases where we can bound this change in the absence of evidence (Sarkhel, Singla, and Gogate 2015). However, bounding this change in the presence of evidence is known to be very challenging, and is still an open problem. Intuitively though, sampling from imbalanced clusters leads to some *meta-atoms* (atoms in the reduced MLN) representing a large cluster of atoms, and others representing a small cluster of atoms of the original distribution. Since we project the meta-atom results to all atoms in its cluster (marginal probability or MAP assignment), inference error tends to get amplified on larger clusters. Therefore, a heuristic that we use to minimize this error is to sample objects from more balanced groups.

Given a domain, Δ_i, let $\hat{\Delta}_i \subseteq \Delta_i$, and let $d(X_j, X_k)$ denote the distance between X_j, $X_k \in \Delta_i$ in the embedding. Let $\mathcal{N}_K(X, \hat{\Delta}_i)$ denote the K closest neighbors of X in $\hat{\Delta}_i$. We want to search for $\hat{\Delta}_i$ by minimizing,

$$\arg \min_{\hat{\Delta}_i \subseteq \Delta_i} \sum_{X \in \hat{\Delta}_i} \sum_{Y \in \mathcal{N}_K(X, \Delta_i \setminus \hat{\Delta}_i)} d(X, Y) \qquad (2)$$

with the constraint that $|\hat{\Delta}_i| \leq \alpha$. To do this, we start with an empty $\hat{\Delta}_i$. In each iteration, we choose X that minimizes $\sum_{Y \in \mathcal{N}_K(X, \Delta_i \setminus \hat{\Delta}_i \setminus X)} d(X, Y)$, remove it from Δ_i and add it to $\hat{\Delta}_i$. We also remove K nearest neighbors of X from Δ_i. Specifically K is chosen to be equal to $\frac{\Delta_i}{\alpha}$ to ensure balanced clusters. This neighborhood is now represented by X.

We stop adding elements to $\hat{\Delta}_i$ when $|\hat{\Delta}_i| \geq \alpha$. If there are objects that remain in Δ_i, we sample one of them to represent all remaining objects and add it to $\hat{\Delta}_i$. Since we remove neighbors greedily, we may add an object that is close to a few objects, and miss objects that may be symmetrical to many objects. To avoid these local optima, we borrow from stochastic local search techniques such as MaxWalkSAT (Kautz, Selman, and Jiang 1997). Specifically, we intersperse random walks in the solution-space along with the

Algorithm 1: Obj2vec Lifting

Input: MLN structure $\mathcal{M}$, Evidence $\mathcal{D}$, Inference algorithm $\mathcal{I}$, embedding dimensions m, object bound α

Output: Inference results R

// Encoding

for *each ground formula* f *obtained from* $\mathcal{D}$ **do**
  if f *is satisfied by* $\mathcal{D}$ **then**
    for *sliding window* c **do**
      (v_i, v_o) = Create vectorized object pairs of input object and predicted context object
      Add (v_i, v_o) to training data $\mathcal{T}$

// Embedding

H = Learn an embedding using skip-gram model architecture and training data $\mathcal{T}$

// Sampling Objects

for *each domain* **do**
  Δ_i = original domain
  $\hat{\Delta}_i = \{\}$
  while $|\hat{\Delta}_i| \leq \alpha$ **do**
    X^t = With probability p, choose from Δ_i greedily
    X^t = With probability $1-p$, choose Δ_i uniformly at random
    $\hat{\Delta}_i = X^t \cup \hat{\Delta}_i$ Remove X^t and its neighborhood from Δ_i

// Inference

Convert $\mathcal{D}$ to $\mathcal{D}'$ using new domains $\hat{\Delta}_1, \ldots \hat{\Delta}_n$

R' = Run $\mathcal{I}$ on $\mathcal{M}$ conditioning on $\mathcal{D}'$

R = Project R on original atoms

return R

greedy iterations. That is, with probability $1-p$, we select a random X, and with probability p, we choose X greedily. Thus, in each iteration, we remove a cluster of neighbors that are symmetric with the chosen object added to $\hat{\Delta}_i$. We denote the neighborhood of $\bar{X}^t$ by $\mathcal{H}(\bar{X}^t)$. All inference results obtained on $\bar{X}^t$ are assumed to hold for $\mathcal{H}(\bar{X}^t)$.

Given the new domains $\hat{\Delta}_1, \ldots \hat{\Delta}_k$, we change the evidence database $\mathcal{D}$, by removing all evidences that contain objects that are not in the new domains. We map the results obtained from inference using this new evidence database to inference results on the original MLN as follows. The inference results (marginal probability, MAP assignment, etc.) obtained after conditioning on the changed evidence, for an atom grounded with objects $X_1 \ldots X_k$ is assumed to hold for all atoms in the original MLN grounded with objects $\mathcal{H}(X_1) \times \ldots \times \mathcal{H}(X_k)$. Our complete approach is summarized in Algorithm 1.

Experiments

Setup

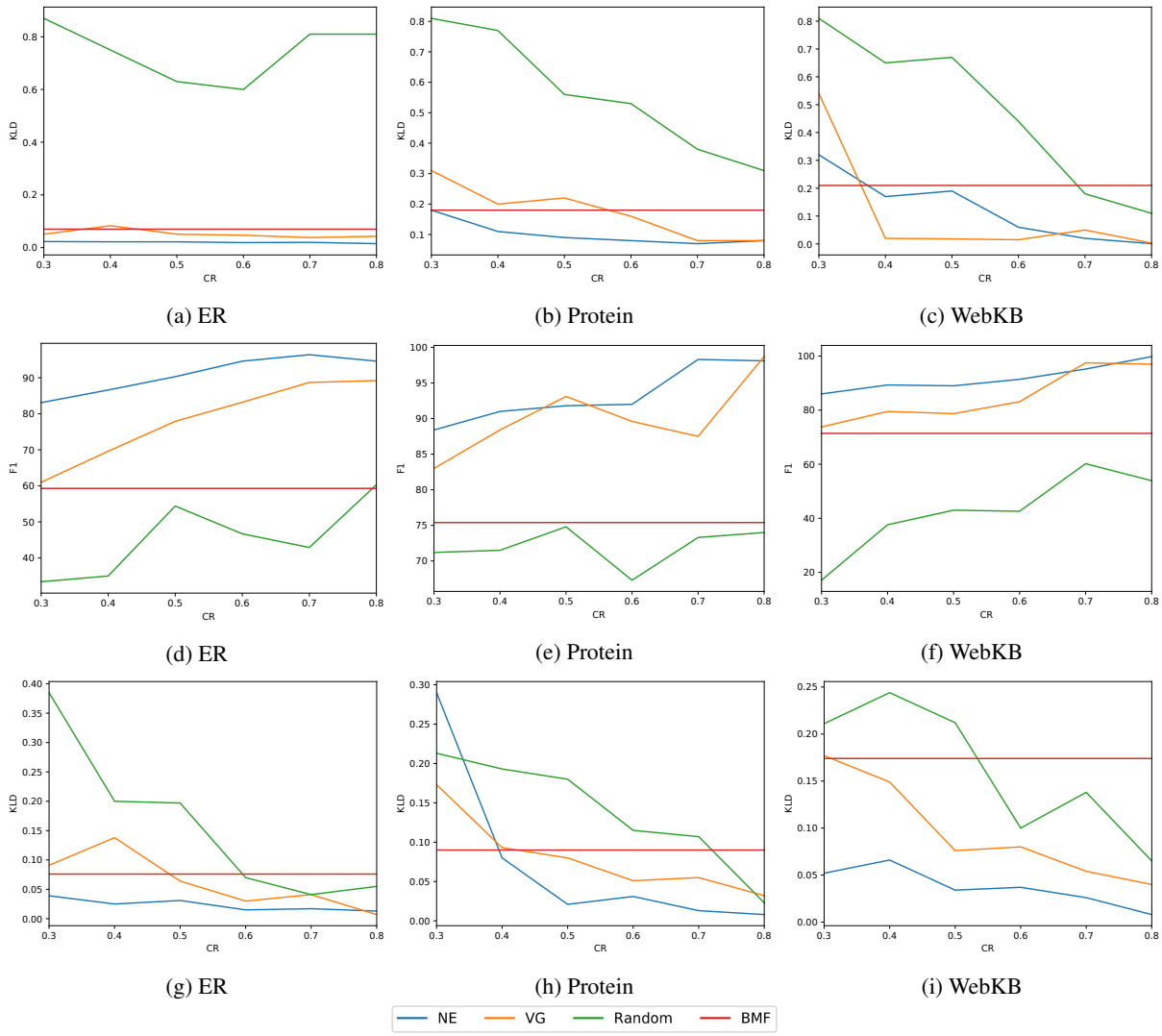
We conducted our experiments on three problems, Webpage classification (Webkb), Entity Resolution (ER) and Protein Interaction (Protein), all of which are publicly available in Alchemy (Kok et al. 2006). We implemented Obj2vec using the Gensim package (Řehůřek and Sojka 2010). For the sampler, we set $p = 0.01$ to insert random walks into the sampling. For performing inference, we integrated Obj2vec with two state-of-the-art inference systems, Tuffy (Niu et al. 2011) and Magician (Venugopal, Sarkhel, and Gogate 2016). Tuffy uses databases to perform efficient grounding, while Magician uses approximate counting within inference to scale up. In Tuffy, we performed marginal inference using MCSAT and MAP inference using MaxWalkSAT. In Magician, we used Gibbs sampling with approximate counting for marginal inference. Note that we also tried to perform inference with algorithms implemented in Alchemy, but due to problems in grounding the MLN, Alchemy could not work with our benchmarks. Note that we assumed the objects come from a fixed finite domain (as is done in most cases for MLNs), so this means that there are no unknown objects during testing (otherwise Obj2Vec may not be able to embed these objects during testing).

We compared our Obj2Vec-based pre-processing approach which we refer to as NE, with two other pre-processing based approaches that learn approximate symmetries. Venugopal and Gogate's (Venugopal and Gogate 2014) approach (VG) available in Magician compresses the MLN using K-Means clustering, with features based on counts of atoms satisfied by the evidence. Binary Matrix Factorization (BMF) (Van den Broeck and Darwiche 2013) pre-processes binary evidence. Specifically, we implemented BMF using the NIMFA library in python and smoothed binary evidences using a low-rank approximation. Finally, we added a baseline method that reduces the evidence database by randomly sampling the evidence atoms in the evidence, which we refer to as Random.

Results

Accuracy For each of the benchmarks, we learned the weights using Magician (Venugopal, Sarkhel, and Gogate 2016) (since learning is more efficient here than Tuffy). We then used around 10% of the benchmark data as test data. We needed to do this to ensure that the baseline inference results that we obtained were reliable using existing inference systems since for larger datasets, inference in Tuffy does not scale up. For Obj2vec, we set the hidden layer to have 300 neurons (a typical size recommended for word embeddings (Mikolov et al. 2013)) (results on varying this presented later).

To compare different approaches, we had to use a common measure of *compression* in each of the pre-processors, i.e., by how much do they reduce the size of the MLN. For a fair comparison, we would need all the reduction in MLN size to be approximately the same across all methods. We defined this reduction in terms of compression ratios (CR). Specifically, CR is the average of the ratios of original-domain-size and new-domain-size (after pre-processing) across all domains. A larger CR means that we are utilizing more objects to approximate the original domain. For VG we can achieve this CR through the parameter that controls the number of clusters. For NE, we con-



(a) ER (b) Protein (c) WebKB

(d) ER (e) Protein (f) WebKB

(g) ER (h) Protein (i) WebKB

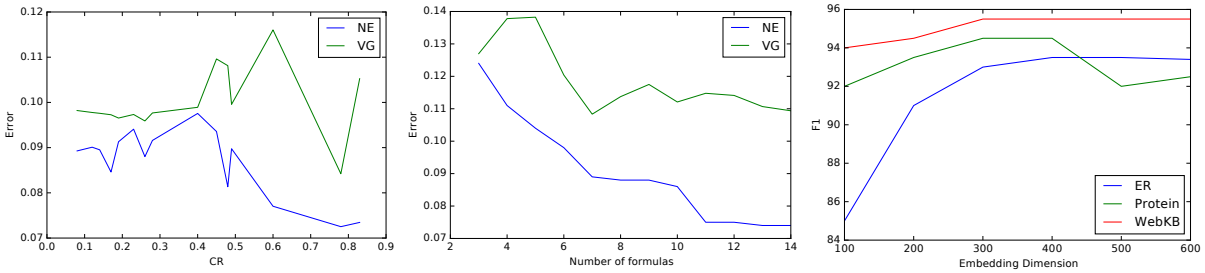
Figure 1: Inference results. (a) - (c) Marginal Inference in Tuffy (d) - (f) MAP Inference in Tuffy (g) - (i) Marginal Inference in Magician.

trol this by setting the α value during sampling to achieve the required CR. For Random, we control the sample size to achieve the required CR. For BMF, achieving the right CR was not possible since changing rank does not change the CR. Therefore for BMF, we varied rank from 20 to 100 (30 is the fault value in NIMFA), which is similar to the ranks used in (Van den Broeck and Darwiche 2013), and report the best results across the ranks.

Using the pre-processed evidence obtained using Obj2vec, VG, and BMF, we performed marginal inference and MAP inference using Tuffy and Magician. For MAP inference, we computed the F1-score based on the atoms on which we correctly obtained the MAP assignment. Specifically, the gold standard is the set of atoms set to true in the MAP assignment, when we perform inference with the original evidence. Based on this gold standard, we compute the precision and recall of MAP inference after pre-processing with Obj2vec and VG respectively. For marginal inference, we computed the average error on the marginal probabilities computed for the query variables, where the error is measured w.r.t the marginal probabilities obtained when we perform inference using the full evidence (no pre-processing).

Fig. 1 shows our results for all benchmarks, inference algorithms, and lifting approaches. It is evident that our approach consistently outperforms every other approach on all benchmarks for both marginal and MAP inference.

Exact Inference We evaluated our approach with a synthetic MLN on which exact inference is tractable. Specifically, we constructed m formulas of the form $\mathtt{R}(x) \vee \mathtt{S}_1(x,y) \vee \mathtt{T}_1(x,z)$; $\ldots$ $\mathtt{R}(x) \vee \mathtt{S}_m(x,y) \vee \mathtt{T}_m(x,z)$, with weights ranging from -1 to 1. We set the domain of x to have 50 objects, and the domains of y and z to have 100 objects each. We only set evidence on $\mathtt{S}_1 \ldots \mathtt{S}_m$ and $\mathtt{T}_1 \ldots \mathtt{T}_m$ atoms, and considered $\mathtt{R}$ as the query predicate. Using the lifted inference rules in PTP (Gogate and Domingos 2011), we



(a) Exact (b) Exact (c) Embedding

Figure 2: (a) Error on synthetic MLN using exact inference baseline for different compression ratios (CR), for BMF we obtained an error of around 0.1 (b) Error on synthetic MLN using exact inference baseline, varying with the number of formulas in the synthetic MLN. (c) Variation of MAP Inference Accuracy with Embedding size

Benchmark	Orig	BMF	VG	Rand	NE
ER	28;56.1	4.7;15.2	12.2;15.3	21.5;47	7.3;12.7
Protein	21.9;61	5;15.5	10.4;19.8	18.3;43.3	6.1;14.2
WebKB	41.3;112.6	5.3;23.4	17;22	32.8;79.9	7.8;31.4

Table 1: Comparison of running times (in minutes) for MAP inference using Tuffy and Marginal inference using Magician. The results are reported in the format (MAP;Marginal) in each column.

could compute the exact marginal probabilities for the query atoms. We computed the average absolute error between marginals obtained before pre-processing and marginals obtained after pre-processing for the query atoms (for $m = 10$). Fig. 2(a) shows our result for different values of CR. As seen here, with increased CR, the marginals approach the true marginal probabilities much faster in NE as compared to VG, and the accuracy obtained is better than BMF.

Increasing Context. For the same synthetic MLN (described above), we changed the context by varying m, i.e., changing the number of formulas. As the number of formulas increase, the context for the domain objects corresponding to x will increase. This leads to our embedding being much better and learning better representations for the objects as opposed to VG (which uses hand-coded features), leading to increasingly better performance as seen in Fig. 2(b).

Embedding Dimension Fig. 2(c) shows the accuracy of MAP inference (in terms of F1) as we change the number of neurons in the hidden layer, i.e., we change the embedding dimensions. As seen here, for WebKB and Protein benchmarks, the results were relatively stable as we varied the dimensions. For ER, changing the dimensions had a more significant effect. As with other applications of neural networks, choosing the right embedding size is challenging in our case as well. In the future, we plan to integrate our approach while learning the MLN, in which case, we can optimize the embedding size based on training or validation sets.

Running Time We measured the total time required to generate the embedding and reduce the objects, plus the time for inference on the reduced MLN. We measured the running time for Tuffy (MAP) and Magician (marginal). Table. 1 shows our results where we averaged the running times for VG, NE and Random across CRs ranging from 0.1 to 0.8. The pre-processing time is much smaller (less than 10%) than the time taken for inference in each of our benchmarks. As seen in these results, NE is more scalable than VG while generating more complex symmetry features. BMF was the fastest algorithm, but the accuracy (as seen in the previous section) was lower than NE and VG.

Conclusion

In this paper, we proposed a novel subsymbolic representation for MLNs that is based on symmetries in the underlying model. The main motivation for this representation was that leveraging symmetries is crucial to scaling up inference in MLNs. We proposed an efficient way to learn the symmetry-based representation by predicting objects in the context of other objects in the MLN akin to *skip-gram* based word embeddings. Our formulation leveraged efficient implementations for learning the embeddings scalably. Our experiments that used the object embeddings within inference algorithms showed our approach to be more scalable and accurate as compared to state-of-the-art systems. Future work includes leveraging neural network architectures to perform scalable discriminative learning (Islam, Sarkhel, and Venugopal 2018; Farabi, Sarkhel, and Venugopal 2018).

References

Anand, A.; Grover, A.; Mausam; and Singla, P. 2016. Contextual Symmetries in Probabilistic Graphical Models. In *International Joint Conference in Artificial intelligence*, 3560–3568.

Apsel, U.; Kersting, K.; and Mladenov, M. 2014. Lifting Relational MAP-LPs Using Cluster Signatures. In *Uncertainty in Artificial Intelligence*, 2403–2409.

Bach, S. H.; Broecheler, M.; Huang, B.; and Getoor, L. 2017. Hinge-loss Markov Random Fields and Probabilistic Soft Logic. *J. Mach. Learn. Res.* 18(1):3846–3912.

Bordes, A.; Weston, J.; Collobert, R.; and Bengio, Y. 2011. Learning Structured Embeddings of Knowledge Bases. In *AAAI Conference in Artificial Intelligence*, 301–306.

Bui, H. H.; Huynh, T. N.; and Riedel, S. 2013. Automorphism Groups of Graphical Models and Lifted Variational Inference. In *Uncertainty in Artificial Intelligence*, 132–141.

de Salvo Braz, R. 2007. *Lifted First-Order Probabilistic Inference*. Ph.D. Dissertation, University of Illinois, Urbana-Champaign, IL.

Domingos, P., and Lowd, D. 2009. *Markov Logic: An Interface Layer for Artificial Intelligence*. San Rafael, CA: Morgan & Claypool.

Farabi, K. M. A.; Sarkhel, S.; and Venugopal, D. 2018. Efficient Weight Learning in High-Dimensional Untied MLNs. In *International Conference on Artificial Intelligence and Statistics*, 1637–1645.

Gogate, V., and Domingos, P. 2011. Probabilistic Theorem Proving. In *Uncertainty in Artificial Intelligence*, 256–265.

Islam, M. M.; Sarkhel, S.; and Venugopal, D. 2018. Learning Mixtures of MLNs. In *AAAI Conference on Artificial Intelligence*, 6359–6366.

Kautz, H.; Selman, B.; and Jiang, Y. 1997. A General Stochastic Approach to Solving Problems with Hard and Soft Constraints. In Gu, D.; Du, J.; and Pardalos, P., eds., *The Satisfiability Problem: Theory and Applications*. New York, NY: American Mathematical Society. 573–586.

Khot, T.; Balasubramanian, N.; Gribkoff, E.; Sabharwal, A.; Clark, P.; and Etzioni, O. 2015. Exploring Markov Logic Networks for Question Answering. In *Empirical Methods in Natural Language Processing*, 685–694.

Kiela, D., and Bottou, L. 2014. Learning Image Embeddings using Convolutional Neural Networks for Improved Multi-Modal Semantics. In *Empirical Methods in Natural Language Processing*, 36–45.

Kok, S.; Sumner, M.; Richardson, M.; Singla, P.; Poon, H.; and Domingos, P. 2006. The Alchemy System for Statistical Relational AI. Technical report, Department of Computer Science and Engineering, University of Washington, Seattle, WA. http://alchemy.cs.washington.edu.

Kopp, T.; Singla, P.; and Kautz, H. A. 2015. Lifted Symmetry Detection and Breaking for MAP Inference. In *Neural Information Processing Systems*, 1315–1323.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Neural Information Processing Systems*, 3111–3119.

Mladenov, M., and Kersting, K. 2015. Equitable Partitions of Concave Free Energies. In *Uncertainty in Artificial Intelligence*, 602–611.

Nickel, M.; Murphy, K.; Tresp, V.; and Gabrilovich, E. 2016. A Review of Relational Machine Learning for Knowledge Graphs. *Proceedings of the IEEE* 104(1):11–33.

Niepert, M. 2012. Markov Chains on Orbits of Permutation Groups. In *Uncertainity In Artificial Intelligence*, 624–633.

Niu, F.; Ré, C.; Doan, A.; and Shavlik, J. W. 2011. Tuffy: Scaling up Statistical Inference in Markov Logic Networks using an RDBMS. *PVLDB* 4(6):373–384.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global Vectors for Word Representation. In *Empricial Methods in Natural Language Processing*.

Poole, D. 2003. First-Order Probabilistic Inference. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, 985–991.

Řehůřek, R., and Sojka, P. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, 45–50.

Rocktäschel, T., and Riedel, S. 2017. End-to-end Differentiable Proving. In *Neural information Processing Systems*, 3791–3803.

Sarkhel, S.; Venugopal, D.; Singla, P.; and Gogate, V. 2014. Lifted MAP inference for Markov Logic Networks. In *International Conference on Artificial Intelligence and Statistics*, 859–867.

Sarkhel, S.; Singla, P.; and Gogate, V. 2015. Fast Lifted MAP Inference via Partitioning. In *Neural Information processing Systems*, 3222–3230.

Singla, P., and Domingos, P. 2008. Lifted First-Order Belief Propagation. In *AAAI Conference in Artificial Intelligence*, 1094–1099.

Singla, P.; Nath, A.; and Domingos, P. 2014. Approximate Lifting Techniques for Belief Propagation. In *AAAI Conference in Artificial Intelligence*, 2497–2504.

Van den Broeck, G., and Darwiche, A. 2013. On the Complexity and Approximation of Binary Evidence in Lifted Inference. In *Advances in Neural Information Processing Systems 26*, 2868–2876.

Van den Broeck, G., and Niepert, M. 2015. Lifted Probabilistic Inference for Asymmetric Graphical Models. In *AAAI Conference in Artificial Intelligence*, 3599–3605.

Van den Broeck, G.; Taghipour, N.; Meert, W.; Davis, J.; and De Raedt, L. 2011. Lifted Probabilistic Inference by First-Order Knowledge Compilation. In *International Joint Conference on Artificial Intelligence*, 2178–2185.

Venugopal, D., and Gogate, V. 2012. On Lifting the Gibbs Sampling Algorithm. In *Proceedings of the 26th Annual Conference on Neural Information Processing Systems (NIPS)*, 1664–1672.

Venugopal, D., and Gogate, V. 2014. Evidence-based Clustering for Scalable Inference in Markov Logic. In *European Conference on Machine Learning*.

Venugopal, D., and Rus, V. 2016. Joint Inference for Mode Identification in Tutorial Dialogues. In *26th International Conference on Computational Linguistics*, 2000–2011.

Venugopal, D.; Chen, C.; Gogate, V.; and Ng, V. 2014. Relieving the Computational Bottleneck: Joint Inference for Event Extraction with High-Dimensional Features. In *Empirical Methods in Natural Language Processing*, 831–843.

Venugopal, D.; Sarkhel, S.; and Gogate, V. 2016. Magician: Scalable Inference and Learning in Markov logic using Approximate Symmetries. Technical report, Department of Computer Science, The University of Memphis. https://github.com/dvngp/CD-Learn.