Contrastive Learning in Neural Tensor Networks using Asymmetric Examples

Mohammad Maminur Islam
Trinity College
maminur.islam@trincoll.edu

Somdeb Sarkhel
Adobe Research
sarkhel@adobe.com

Deepak Venugopal
University of Memphis
dvngopal@memphis.edu

***Abstract*—Neuro-Symbolic models combine the best of two worlds, knowledge representation capabilities of symbolic models and representation learning power of deep networks. In this paper, we develop a Neuro-Symbolic approach to infer unknown facts from relational data. A well-known approach is to use statistical relational models such as Markov Logic Networks (MLNs) to perform probabilistic inference. However, these approaches are known to be non-scalable and inaccurate for large, real-world problems. Therefore, given symbolic knowledge, we train a Neural Tensor Network (NTN) to learn representations for symmetries implied by the symbolic knowledge. Further, since the data is interconnected, predicting one fact can positively or negatively impact the prediction of other facts. Therefore, we train the NTN using open-world semantics over multiple possible worlds, learning to represent symmetries in each world. We evaluate our approach in several real-world benchmarks comparing with state-of-the-art relational learning methods, Neuro-Symbolic methods and purely symbolic methods clearly illustrating the generality, accuracy and scalability of our proposed approach.**

***Index Terms*—Neuro-Symbolic Models, Relational Learning, Symmetry Representation Learning**

I. Introduction

Neuro-symbolic learning [4] aims to combine neural networks with symbolic AI models. In the past, convergence between statistical and logical approaches have been explored in different forms such as statistical relational models [8] that combine probabilistic graphical models with first-order logic, or earlier attempts to combine neural networks with logic formalisms [35]. However, given the impact that deep neural networks have had in various domains such as computer vision, natural language understanding etc., it is natural to explore new connections between symbolic AI and deep neural networks. Particularly, by regularizing the representation learning power of deep networks with domain knowledge from symbolic models, we can aim to learn more scalable and generalizable models.

In this paper, we focus on the following problem - given relational data and knowledge about dependencies in the data specified in a knowledge base, we want to infer if an unobserved fact (or atom in first-order logic terminology) is likely to be true. In theory, for performing such inferences, we can use well-known statistical relational models such as Markov Logic Networks [5]. However, the main challenge is that on real-world problems, these methods scale very poorly and also have poor accuracy [15]. However, since MLNs are symbolic models based on first-order logic (FOL), they can encode rich domain knowledge that can significantly help learning in complex domains [34]. At the same time, deep learning-based methods such as Neural Tensor Networks (NTNs) [37] can be applied to add reasoning capabilities to knowledge bases (e.g. WordNet [23]). In particular, NTNs combine entity vectors using a bilinear tensor layer to represent relationships between entities. In this paper, we leverage the reasoning power of NTNs to develop a general, scalable framework for inference in relational data with FOL formulas describing possible logical dependencies within the data based on our understanding of the domain.

Symmetries are ubiquitous and essential for reasoning in the real-world, and symbolic knowledge can effectively encode symmetries. For example, if we have two FOL formulas $\texttt{Flu}(x) \Rightarrow \texttt{Fever}(x)$ and $\texttt{Flu}(x) \Rightarrow \texttt{Cough}(x)$, then given observations $\texttt{Flu}(A)$ and $\texttt{Flu}(B)$, the observation $\texttt{Fever}(A)$ is symmetrical to the observation $\texttt{Cough}(B)$, i.e., we can likely infer that A has a cough and B has a fever. To leverage such symmetries in the NTN, we train an NTN *contrastively* with asymmetric atoms given their dependencies in the knowledge base, thus forcing the NTN to learn different representations for asymmetric atoms. To do this, we assume semantics of MLNs to represent the relational data. That is, the FOL formulas represent cliques in a probabilistic graphical model. Learning symmetries from these models is well-known to be a hard problem and one that is often a bottleneck in *lifted inference* [41] for MLNs and other related models. Here, we develop a scalable approach where we learn an embedding for objects [12] shared across atoms such that objects that are symmetrical in the knowledge base are close to each other in the embedding-space. We then use NTNs to compose object embeddings into atom representations such that symmetrical atoms have similar representations and use these representations to make predictions about the atoms.

Further, to quantify uncertainty in our prediction, we need to explore multiple possible worlds (in MLN/FOL terminology). Specifically, in relational data, the predictions are related to each other and one prediction can affect the others. E.g.,

This research was sponsored by NSF IIS award #2008812 and NSF award #1934745. The opinions, findings, and results are solely the authors' and do not reflect those of the funding agencies.

978-1-6654-3902-2/21/$31.00 © 2021 IEEE

suppose we have a formula $\texttt{Flu}(x) \wedge \texttt{Samehouse}\ (x, y) \Rightarrow \texttt{Flu}(x)$, then if A and B stay in the same house, in a world where we predict flu for A, we are more likely to predict that B has a flu, but in a world where A does not have the flu, B is less likely to have the flu. Therefore instead of optimizing the NTN parameters over a single world, we learn using open-world semantics on predicted atoms. Specifically, we re-sample atoms based on their prior predictions to sample a new world on which we train the NTN. Thus, the NTN learns to represent symmetries between atoms over multiple possible worlds. Since we relate objects multiplicatively using multiple tensors we can compose object representations into different atom representations to represent symmetries over different possible worlds.

We evaluate our approach on knowledge base completion benchmarks and node classification benchmarks comparing with state-of-the-art relational learning approaches as well as general Neuro-symbolic methods and show that our approach performs better or on par with these methods. Further, we also show the versatility of our approach on other types of problems including image segmentation where we obtain better performance than U-Net which is a state-of-the-art image segmentation method and on text classification tasks. We also compare our approach with purely MLN-based approaches and show that our approach is orders of magnitude more scalable and accurate compared to these methods.

II. Related Work

Our work is broadly related to general Neuro-symbolic approaches as well as specialized relational learning methods.

General Models. While combining neural networks with symbolic systems has a long history [35], there has been a lot of recent interest under the broad umbrella of Neuro-symbolic learning [4]. Models for logical inference through neural nets have been proposed such as [30]. Tensor Logic Nets [33] similar to our work is inspired by NTNs but for logical inference. Modeling relational knowledge as graphs and using graph-based neural network learning algorithm methods have also been proposed. [19] provide a survey and taxonomy of Neuro-symbolic computing with graph (relational) networks. Well known models of this type include GMNNs [27] that uses GCNs [16] to combine Markov networks with deep learning. Similarly, GATs [42] use attention mechanism in graph-based knowledge representations.

In the realm of logic programming, [21] proposed DeepProbLog, combining neural networks with probabilistic logic. [44] proposed a differentiable semantic loss function that encodes symbolic logical knowledge into the neural network learning algorithm. Marra et al. [22] proposed Neural Markov Logic where they extended Markov logic with potential functions that exploit symmetries. Our work has the same flavor but is distinct in the sense that we do not learn Markov Logic Networks due to poor scalability learning and inference. Instead, we simply use its semantics (FOL representation) as a source of knowledge for the NTN. Very recently, in [47], the authors developed a combination of MLNs and graph-neural networks where they used a variational EM learning approach. In general, the core of our approach which relies on exploiting symmetries in knowledge has connections with other general approaches for learning such as exchangeable variable models [25]. Islam et al. [13] developed a CNN model that uses exchangeable variables.

Specialized Models. Our work is also related to knowledge graph embeddings which are more specialized relational learning problems. [2] is a classical method that proposed embeddings for knowledge graphs. Other well-known embeddings include ReScal [24] and ComplEx [40]. In particular, [14] proposed SimpleIE that learns embeddings using background knowledge. Distmult [45] also uses a bilinear formulation to relate entities for link prediction. RotatE [38] achieves state-of-the-art performance by using a novel adversarial negative sampling approach. Finally, our approach is similar in spirit to deep symmetry nets [7], where they used a CNN that learns features over symmetry groups capturing more broad invariances in object recognition for images. In our case, symmetries are more generally defined in the knowledge-base.

III. Background

A. First-Order Logic

The language of first-order logic (FOL) consists of quantifiers ($\forall$ and $\exists$), logical variables, constants, predicates, and logical connectives ($\vee$, $\wedge$, $\neg$, $\Rightarrow$, and $\Leftrightarrow$). A predicate is a relation that takes a specific number of arguments as input and outputs either TRUE (synonymous with 1) or FALSE (synonymous with 0). The arity of a predicate is the number of its arguments. We assume that each logical variable x has a finite domain of objects Δ_x that it can be substituted with (Herbrand semantics). A ground atom is a predicate where all its variables have been substituted by a constant (we use the terms constants and objects interchangeably) from its domain. For example, $\texttt{Friends}(Alice, Bob)$ is a ground atom obtained by substituting the variables in $\texttt{Friends}(x, y)$.

A first-order formula connects predicates using logical connectives. For example, $\neg\texttt{Friends}(x, y) \vee \texttt{Friends}(y, x)$. A grounding of a first order formula is one where all variables in the formula have been substituted by constants. For example, $\neg\texttt{Friends}(Alice, Bob) \vee \texttt{Friends}(Bob, Alice)$. Note that a ground formula evaluates to either True or False. A knowledge-base is a set of first-order formulas. We assume that our FOL formulas are in the standard conjunctive normal form, i.e., clauses connected by conjunctions. A possible world, denoted by ω, is a truth assignment to all possible ground atoms in the first-order KB.

Markov Logic. Markov logic networks (MLNs) add weights to FOL formulas to capture uncertainty. Each ground formula acts as a potential function (or clique) in an undirected graphical model that is parameterized by the weight attached to that formula. MLNs can represent very large graphs compactly due to the shared first-order structure over a large number of potential functions. That is, the number of potentials depends upon the number of groundings or ground formulas of a

first order formula. The number of grounding of a first-order formula depends upon depends upon the domain-size of the variables within the FOL formulas. Thus, the FOL formulas act as templates where depending on the number of objects specified in the domains for the variables in the formulas, different graphical models corresponding to the same MLN can be generated. In the typical MLN assumption, all groundings for a formula are parameterized by the same weight. A parameterized MLN represents a probability distribution over the set of possible worlds.

IV. Proposed Approach

Learning Task. Given a world ω, where a subset of atoms are fixed as evidence (or observed atoms) we want to learn a model that predicts if a non-observed atom is true. We also assume that we have domain knowledge about the problem-of-interest represented as a set of first-order logic (FOL) formulas.

A standard approach is to apply a standard statistical relational model (e.g. Markov logic) and use max-likelihood estimation for learning weights/parameters for the FOL formulas. However, the main problem is that the scalability and accuracy of such models is known to be very poor [15]. Instead, here, we train a more scalable and accurate discriminative deep neural network model that learns contrastive representations for asymmetric atoms based on the structure of the FOL formulas.

A. Overview

We motivate our approach with a simple example shown in Fig. 1. The example shows a graphical model assuming Markov Logic semantics. Here, each clique corresponds to a ground formula and the nodes correspond to ground atoms. Let the green cliques denote functions that are active, i.e., the formula is satisfied based on the assignment to the nodes and the red cliques denote the non-active functions. For different worlds (a 0/1 assignment to all nodes), the active/satisfied and inactive/unsatisfied formulas vary as shown in the example. Here, we assume that weights are the same for all the formulas and therefore, we are only interested in which formulas are satisfied (or unsatisfied) by a world. Based on this, in each world, we can identify symmetries among nodes, i.e., the nodes can be exchanged to create an isomorphic graph. However, note that in each world, symmetries change based on the assignments. For instance, for the worlds shown in the figure, the nodes X_2 and X_3 may be exchangeable/non-exchangeable in the graph and at the same time they may have uniform/non-uniform assignments. The main goal of our approach is to learn a discriminative model that assigns similar probabilities to atoms when they are symmetric and have uniform assignments across different worlds. To scale up learning, we use Neural Tensor Networks to learn the representation for atoms by combining embeddings for objects that are related in the atom. While in theory, we can also learn atom embeddings directly from the graph using approaches such as Graph Markov Nets [44], our approach scales up better when the graph has first-order structure. Specifically, we learn the object embeddings efficiently without explicitly constructing the graph. The NTN layer relates the embeddings multiplicatively and therefore can combine symmetry information encoded in the object embeddings non-linearly.

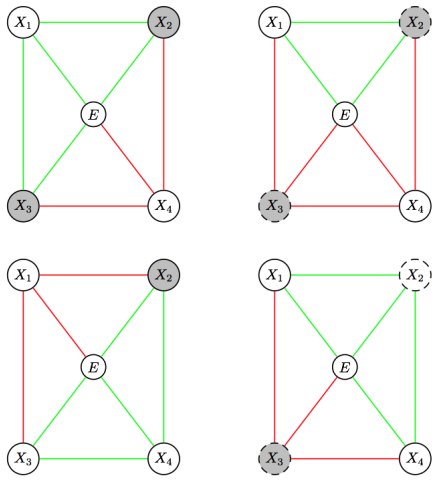


Fig. 1: Illustrating symmetries between X_2 and X_3 in 4 different worlds. world 1: X_2 and X_3 are exchangeable and have uniform assignments, world 2: not exchangeable but uniform assignments, world 3: exchangeable but non-uniform assignments and world 4: not exchangeable and non-uniform assignments. Our proposed approach learns to assign similar probabilities to X_1 and X_2 in world 1.

B. Contrastive Learning

For each predicate type, we learn an Neural Tensor Network (NTN) corresponding to that predicate where for every `True` atom X in ω corresponding to that predicate, we contrast it with $\bar{\mathbf{X}}$, a set of `False` atoms that are also asymmetric to X to train the model. The training objective is given by,

$$J(\Omega;\omega) = \min_{\Omega} \sum_{i=1}^{N} \sum_{\bar{X}\in\bar{\mathbf{X}}} max(0, 1 - (g(X) - g(\bar{X})) + \lambda||\Omega||_2^2 \quad (1)$$

where Ω represents the NTN parameters, N is the number of atoms in the training data ω, $\bar{X}$ is an atom of opposite assignment to X. $g()$ is the output of the NTN that scores each atom (larger value indicates that the atom is true).

NTN architecture. To learn the function $g()$, for each predicate type, the NTN uses a bilinear tensor layer to relate the objects in atoms corresponding to that predicate. For simplicity, assume that the predicate type for atom X is a binary predicate (where `R` is the type) and X^0 and X^1 are the objects in the atom. To make exposition easier to follow, we assume a single predicate type of binary arity unless

otherwise specified (we can extend the same easily to higher-arity atoms). In practice, we learn a separate NTN for each predicate type. The function $g()$ is defined as follows.

$$g(X) = u_R^\top f(v_{X^0}^\top W_R^{[1:k]} v_{X^1} + V_r \begin{bmatrix} v_{X^0} \\ v_{X^1} \end{bmatrix} + b_R) \quad (2)$$

where $v_{X^0} \in \mathbb{R}^d$ is a d-dimensional vector representation for object X^0, $W_R^{[1:k]} \in \mathbb{R}^{d \times d \times k}$ is a tensor, $f()$ is a tanh non-linearity applied element-wise and $b_R \in \mathbb{R}^k$ is the bias and $u_R \in \mathbb{R}^k$ is a weight vector. $v_{X^0}^\top W_R^{[1:k]} v_{X^1}$ results in a vector representation $v_X \in \mathbb{R}^k$, where each entry is computed by one tensor slice. The i-th dimension of v_X is computed using the i-th tensor slice. $v_{X^0}^\top W_R^{[i]} v_{X^1}$. $V_r \in \mathbb{R}^{k \times 2d}$ relates the concatenated object vectors.

Training. Minimizing the objective in Eq. (1) creates a separation between the `True` and `False` atoms. In theory, we can randomly sample atoms of opposite assignment to contrast with X. However, while this may classify atoms according to their truth value, it does not force the NTN to learn a representation that encodes symmetries between atoms. For example, Fig. 2 shows 2 possible representations for the same graph, both of which are accurate in classifying the atoms. However, we want the NTN to prefer the representation where symmetric atoms are grouped together.

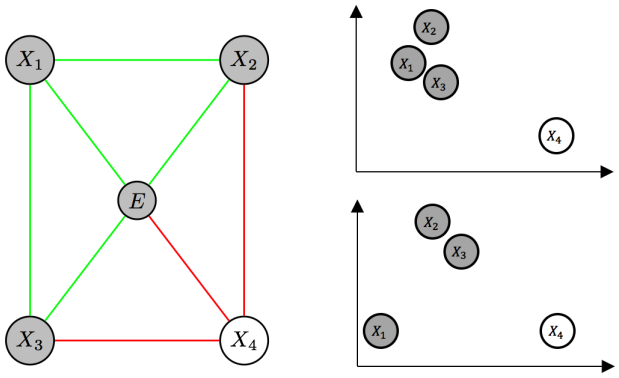


Fig. 2: For the world shown in the figure, the approaches on the right are both correct classifiers but the bottom one represents symmetries while the top one does not.

Thus, for every atom X, we want $\bar{X}$ to contain atoms that not only have the opposite assignment to X but are also asymmetric to X. This is non-trivial since we do not know symmetries between atoms apriori. Therefore, similar to weak supervision, we use a noisy approximation based on symmetries between objects.

Definition 1. *Given a world, atom X is exchangeable with X' in that world if for every ground formula the truth assignment for that formula is unchanged when all instances of X are replaced with X' and vice-versa.*

Definition 2. *Given a world, object O is exchangeable with O' in that world if for every ground formula the truth assignment for that formula is unchanged when all instances of o_1 are replaced with o_2 and vice-versa.*

Definition 3. *A monadic FOL formula is one where each atom in the formula is constrained to be a unary atom (also called singleton, e.g. $\mathtt{R}(x)$).*

Proposition 1. *Given a set of monadic FOL formulas, in a world ω, if object O is exchangeable with O' then this implies that every atom that contains O is exchangeable with an atom containing O'.*

Proof. Let X contain the object O and X' contain object O'. Assume that O is exchangeable with O' but X is not exchangeable with X'. This means that there exists at least one ground formula f such that exchanging X with X' flips the truth assignment to f. Since X and X' are unary atoms, this implies that exchanging O with O' in f flipped the truth assignment to f which is a contradiction since O and O' are exchangeable. □

From the given set of FOL formulas for the domain-of-interest, we construct a monadic approximation of the FOL formulas as follows. For every k-ary atom, $\mathtt{R}(x_1 \dots x_k)$, we substitute it with $\mathtt{R}(x_1) \vee \mathtt{R}(x_2) \dots \vee \mathtt{R}(x_k)$. We remove any redundant atoms that are repeated in the clause. Note that this approximation is similar to the ones used in [41]. Using this approximation, we vectorize objects such that exchangeable objects have similar vector representations. Specifically, we apply *Obj2Vec* [12] to learn a dense vector representation using word-embedding models. Here, for each ground formula f that is satisfied by the world, f acts as a positive instance for the model. We consider each object O in f and predict the *context* of O, where the context refers to other objects that occur in f. Importantly, when objects are exchangeable, they will have similar contexts. This allows us to learn a continuous approximation for exchangeable objects since objects with similar (but not identical) contexts will be represented by similar (though not identical) vectors in the embedding. Thus, we obtain vector representations for objects that also represent atoms if we assume monadic formulas. To learn atom representations w.r.t the original FOL formulas, we learn an NTN to combine the vectors of objects corresponding to an atom.

To train the NTN, for an atom $\mathtt{R}(X^0, X^1)$, we contrast it with atoms that are asymmetric with it in the monadic approximation. That is, we choose objects that are distant from X^0 and X^1 in the embedding. To ensure that our contrastive examples are taken from sufficiently diverse parts of the embedding space, we cluster the object embeddings into K clusters, $\mathbf{C} = \mathbf{C}_1 \dots \mathbf{C}_K$. Given that an atom X contains an object in cluster $\mathbf{C}_i$, we contrast it with atoms that have objects in $\mathbf{C} \setminus \mathbf{C}_i$. To do this, we sample all clusters other than $\mathbf{C}_i$ and pick atoms of opposite assignment to X that contain the sampled objects.

C. Symmetry Representation

From prior work on lifted inference, we can define three general forms of symmetry as follows.

Definition 4. *Uniform Assignment (*`UA`*) is a symmetry on a group of atoms where each of the atoms are constrained to have the same assignment. The group of atoms can therefore be replaced by a single atom. The MAP inference rule defined in lifted inference algorithms [32] illustrates such symmetries.*

Definition 5. *Count Symmetry (*`CS`*) is a symmetry on a group of atoms where configurations (assignments to atoms in the group) that have the same number of* `true` *atoms are symmetric to each other. The binomial rule defined in lifted inference algorithms such as PTP [9] illustrates such symmetries.*

Definition 6. *Variable-Value Symmetry (*`VS`*) is a symmetry on a group of atoms that constrains specific configurations (assignments to atoms in the group) to be equivalent to each other. Such symmetries are discovered using graph-based algorithms in [1].*

Suppose our model assumes a type of symmetry, clearly, in terms of bias of the model, `UA` models $>$ `CS` models $>$ `VS` models. That is, assuming `UA` makes the model simpler but may fit the data poorly since such symmetries are less representative of real-world data. Given the embeddings for objects in an atom, we can combine them using simpler methods such as additive compositions [3]. However, this approach will yield `UA` models. Specifically, let $\mathbf{C}_1$ and $\mathbf{C}_2$ represent clusters of objects in the embedding-space. Suppose $v_1 \in \mathbf{C}_1$ and $v_2 \in \mathbf{C}_2$. An additive composition $v_1 + v_2$ is likely to be similar to other compositions that can be formed from C_1 and C_2. Thus, any atom that includes objects from $\mathbf{C}_1$ and $\mathbf{C}_2$ is constrained to have a very similar representation which means that the model implicitly assumes *UA* symmetry among atoms.

On the other hand, consider the NTN multiplicative function in Eq. (2) that combines two object vectors. In this case, multiplying the vectors with each tensor slice followed by the non-linearity generates a binary value. Thus, the model can generate up to 2^k unique representations assuming k tensors. Specifically, the model can assign each atom that can be formed from clusters of objects $\mathbf{C}_1$ and $\mathbf{C}_2$ to one of the 2^k representations each of which can have a different assignment. Thus, there are 2^k possible configurations of atom assignments that can be generated by the model and the model can therefore represent *VS* symmetries. For example, the toy example in Fig. 3 (a) shows atoms arranged in a 2-D grid based on the values of their object embeddings. That is, each axis is an object's 1-D embedding. Note that the symmetries across different rows and columns are different (indicated by the true/false atom colors). The training accuracy results from our NTN-based approach is shown in Fig. 3 (b) for different number of tensor slices. As seen here, for smaller number of tensor slices the NTN could not accurately infer all the symmetries, but increasing the number of tensor slices results in a significant increase in accuracy (reduction of bias) since the model can assign each tensor-slice to detect different variants of the symmetries.

D. Open World Training

Eq. (1) which optimizes $J(\Omega; \omega)$ implicitly assumes a closed world, i.e., when we are predicting $g(X)$, the assignments to all atoms other than X in the world are known to us. However, since our training data is relational, it in fact consists of several interconnected atoms. In particular, predictions made for one atom can affect other predictions. Therefore, we train the model such that we optimize over *open world* atoms. Specifically,

Definition 7. *Given ω, the open-world atoms $\mathcal{O}$ is a subset of atoms in ω such that the possible worlds (denoted by $\mathcal{W}$) contain every possible configuration of assignments to atoms in $\mathcal{O}$ and $\forall\ X \notin \mathcal{O}$, the assignment to X remains unchanged from its assignment in ω.*

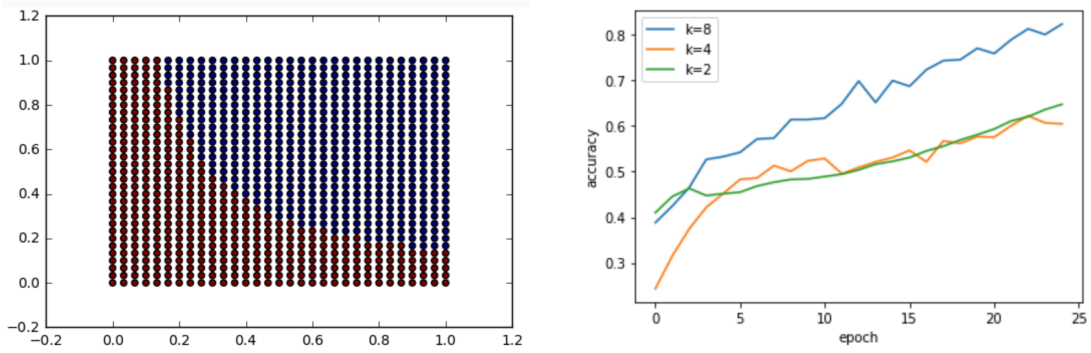
Therefore, to optimize over open-world atoms, we modify the training objective function as,

$$J^*(\Omega) = \min_{\Omega} \sum_{\omega' \in \mathcal{W}} J(\Omega; \omega') \tag{3}$$

However, optimizing Eq. (3) exactly is clearly infeasible since $|\mathcal{W}|$ is exponential in $|\mathcal{O}|$. To train our model tractably, we use an approach similar to stacking used to learn mixture models [36, 11]. Specifically, we learn a model that optimizes Eq. (3) over worlds sampled based on the "out-of-sample" prediction scores for the open world atoms. Specifically, we assume a fully factored mean-field distribution on the open world atoms.

$$P(\mathcal{O}|\Omega) = \prod_{X \in \mathcal{O}} g(X|\Omega) \tag{4}$$

where $g(X|\Omega)$ is the out-of-sample (normalized) score between 0 and 1 for an open-world atom X given the learned parameters Ω. That is, we assume that Ω is estimated without using X in training the model. Note that the distribution assigns higher probabilities for larger scores in predicting $\mathcal{O}$ which indicate that the atoms can be more accurately predicted using symmetries in the world. However, note that to compute the distribution, we need Ω which in turn requires learning the model over multiple worlds. Therefore, we use a coordinate descent approach, where we fix Ω to compute $P(\mathcal{O}|\Omega)$ and then re-estimate Ω from a new world sampled using $P(\mathcal{O}|\Omega)$. In theory, we can guarantee convergence to a local optima by rejection sampling, i.e., if the objective $J(\Omega; \omega')$ is larger than the previous value, we reject the sampled world ω' and re-sample a new world. However, in practice, we noticed that this wastes sampled worlds and in many cases, it may be better to explore more diverse worlds to capture their symmetries during learning. Therefore, we instead set a fixed number of iterations for stopping or stop if the world converges, i.e., the



(a) (b)

Fig. 3: Toy example where (a) shows atoms with true/false assignments shown by the red/blue colors. Each axis denotes an object's 1-D embedding. Each row and column has varying true/false assignments. (b) shows the training accuracy in our model for different number of tensor slices.

assignments to all the open world atoms reach a fixed point before these maximum iterations. Further, note that to compute the distribution $P(\mathcal{O}|\Omega)$, we need to learn $|\mathcal{O}|$ different models since for each atom $X \in \mathcal{O}$, we need to compute its out-of-sample score, which is computationally expensive. Therefore, similar to cross-validation, we split the world into m parts, train on $m-1$ parts and compute the scores for all the open world atoms on the remaining part.

Algorithms 1 and 2 summarize our approach in training and prediction. We start training from our initial world. In each iteration, we partition $\mathcal{O}$ into m parts. We mask the assignments to atoms in the test partition part to compute the object embeddings. Specifically, recall that to learn the embedding using Obj2Vec, our positive examples are object pairs that occur in satisfied ground formulas given a world. For a partial world, the masked atoms will have no assignment and we assume these to be false (closed world assumption) while learning the embeddings. We then train over all the atoms in the training partitions using the object embeddings. For each atom in the test partition, we predict the atom and sample it using its score. We repeat over all partitions to obtain a sample over $\mathcal{O}$. We combine this with the unchanged assignments from the original training data to obtain a new world. For prediction, we initially assign each atom in $\mathcal{O}$ a random assignment. In each iteration, we update the pre-trained embedding learned during training. Specifically, we use the current world to create new training instances of object pairs to update the pre-trained embedding. We then predict each instance in $\mathcal{O}$ and sample a new world from their scores. We average scores for the atom in all the sampled worlds as our final predicted score.

Algorithm 1: M-NTN Training

Input: FOL formulas F, training data ω, open world atoms $\mathcal{O}$
Output: Trained `M-NTN`

$\omega^{(0)} = \omega$
for *$i = 1$ to T iterations* **do**
  Partition $\mathcal{O}$ into m partitions
  repeat
    $\mathcal{E}$ = Compute object embeddings using $\omega^{(i)}$ where assignments to $\mathbf{T}$ are masked
    `// Train NTN Model using asymmetric examples`
    $\mathbf{C}$ = Cluster $\mathcal{E}$
    for *each atom $X \in \mathbf{P}$* **do**
      $\bar{\mathbf{C}}'$ = clusters that do not contain objects in X
      $\bar{\mathbf{X}}$ = Contrastive atoms for X containing objects sampled from $\bar{\mathbf{C}}'$
      Train `M-NTN` with X and $\bar{\mathbf{X}}$
    `// Sample test partition and update world`
    for *each X in $\mathbf{T}$* **do**
      Sample X from $g(X)$ and add to world $\omega^{(i+1)}$
  until *for each training partition $\mathbf{R}$ and test partition $\mathbf{T}$*;
  Copy assignments to non-open world atoms from $\omega^{(i)}$ to $\omega^{(i+1)}$

V. Experiments

A. Setup

We evaluated our approach (which we refer to as `MNTN`) using the following studies.

- Comparison using the standard publicly available WordNet and Freebase benchmarks [37] with several state of the art methods in knowledge base completion.
- Comparison using the standard publicly available Cora, Citeseer and Pubmed benchmarks [27] with several state-of-the-art Neuro-Symbolic models.
- An ablation study where we evaluated different parts of

Algorithm 2: M-NTN Prediction

Input: Test Database ω, Object embeddings $\mathcal{E}$, Trained NTN `M-NTN`, open world atoms $\mathcal{O}$
Output: Predictions for $\mathcal{O}$

$\omega^{(0)}$ = random assignment to $\mathcal{O}$ combined with other assignments in ω
for *t = 1 to T iterations* **do**
  Update object embeddings $\mathcal{E}$ based on assignments to $\mathcal{O}$
  for *each atom $X \in \mathcal{O}$* **do**
    $\mathbf{v}$ = embedding vectors for objects in X
    Predict $g(X)$ using $\mathbf{v}$ in `M-NTN`
    $\hat{X}^{(t)} = g(X)$
  for *each atom $X \in \mathcal{O}$* **do**
    Sample an assignment for X using $g(X)$
`// Average scores across T iterations`
for *each $X \in \mathcal{O}$* **do**
  for *t = 1 to T* **do**
    $\hat{X} = \hat{X} + \hat{X}^{(t)}$
  for *each $X \in \mathcal{O}$* **do**
    $\hat{P}(X) = \frac{1}{T}\ \hat{X}$

our system. In (`NTN-R`), we randomly sample negative examples, i.e., we do not specifically train on asymmetric atoms. (`NTN-CW`), we train on a single world, i.e. we make the closed world assumption. Additive Composition (`AC`), we implemented a linear composition of object vectors instead of using NTNs to combine the object vectors.

- To show applicability of our proposed approach in other diverse tasks, we implemented image segmentation using the publicly available TU Darmstadt database of images [20] and compare with a state-of-the-art deep learning architecture for segmentation, `UNET` [31]. We also compare with purely symbolic learning and inference using two MLN based learning and inference systems, `Tuffy` [26] and `Magician` [43] using publicly available text classification benchmarks.

1) Implementation: To learn the Obj2Vec embeddings, we used the Gensim [29] implementation of word2vec with the skip-gram model. We implemented the standard NTN architecture as specified in [37] using Tensorflow. All experiments were performed on an AWS cluster with 16 CPUs and 64 GB RAM and a GPU. For the NTN, we used 20 tensor slices after experimentation.

For Knowledge Base Completion, we use the *Freebase* dataset with 13 relations and 75043 entities and the *Wordnet* dataset with 11 relations and 38696 entities [37, 3]. For Object classification, we use the same benchmarks used in `GMNN` [27] For *Segmentation*, we use the TU Darmstadt database of images [20] to perform image segmentation into foreground/background pixels. We used the set of images corresponding to side-views of cows. For text processing, we used *WebKB*, *Yelp* and *Movielens* datasets. The WebKB task and dataset is defined in Alchemy [17] where we classify webpages according to a topic. For Yelp [28], the task is to classify if a review is fake or not. For Movielens [10], we predict movie ratings.

2) Relational Knowledge: .

- For *Knowledge Completion*, corresponding to each triplet, (h, r, t) we encode a formula $\texttt{Head\_Rel}(h, r) \wedge \texttt{Tail\_Rel}(t, r) \Rightarrow \texttt{Head\_Tail}(h, t)$ that specifies the head and tail relations. Further, we add a transitive formula corresponding to each of the three predicates. This is similar to the transitive relationship encoded in [14].
- For *Object Classification*, we encoded the homophily property of the form $\texttt{Class}(x, c) \wedge \texttt{Linked}(x, y) \Rightarrow \texttt{Class}(y, c)$. We also had formulas that connect words in each instance to the class for that instance similar to the bag-of-words formulas in WebKB specified in Alchemy [17].
- For the *Segmentation* task, we define regions in which the pixel lies along both the x-axis and the y-axis and add formulas of the form $\texttt{Region}(x_1, y_1, r) \wedge \texttt{Foreground}(x_1, y_1) \wedge \texttt{Region}(x_2, y_2, r) \Rightarrow \texttt{Foreground}(x_2, y_2)$ where $(x_1,\ y_1)$ and $(x_2,\ y_2)$ are pixel-coordinates. This rule encodes our knowledge that if two pixels are in the same region they are likely to be of the same type (foreground/background). Further, we add formulas that connect the pixel values (discretized to 10 levels for R, G and B channels) along 4 orientations (top, right, bottom, left) to the `Foreground` predicate.
- For text processing, the formulas are similar to *Object Classification* except the relational formulas connect webpages in WebKB, and users (who wrote the review) in Movielens and Yelp.

All the data and code for `MNTN` is available here[1].

B. Results (Knowledge Base Completion)

The comparison results with several state-of-the-art systems are shown in Table. I. The results shown for the other systems are from SimpleIE [14] over 6 different metrics. The methods include canonical Polyadic (CP) decomposition, TransE and its variants (STransE and TransR) [3], ComplEx [40], Distmult [45], NTNs (note that in the result here the NTN does not use our embedding), ER-MLP [6]. The Mean Reciprocal Rank (MRR) is defined in [14]. Here, for each test triple (h, r, t), we compute ranking for h based on the scores of other triples of the form (h', r, t) and similarly, we compute a similar ranking for t. We then take the mean of the inverse of these rankings which is more robust than a mean rank. The filtered MRR is the same as defined in [3]. The HIT@k computes the $\%$ of test triples with ranking $\leq k$. As seen from the results, in majority of the cases, `MNTN` outperformed most of the competing systems over most of the metrics.

[1] https://github.com/tushancse04/MNTN

	WN18					FB15k				
	MRR		HIT@			MRR		HIT@		
Model	Filter	Raw	1	3	10	Filter	Raw	1	3	10
CP	7.5	5.8	4.9	8.0	12.5	32.6	15.2	21.9	37.6	53.2
TransE	45.4	33.5	8.9	82.3	93.4	38.0	22.1	23.1	47.2	64.1
TransR	60.5	42.7	33.5	87.6	94.0	34.6	19.8	21.8	40.4	58.2
DistMult	82.2	**82.2**	82.2	91.4	91.4	91.4	24.2	54.6	73.3	82.4
NTN	53.0	–	–	–	66.1	25.0	–	–	–	41.4
STransE	65.7	46.9	–	–	93.4	54.3	25.2	–	–	79.7
ER-MLP	71.2	52.8	62.6	77.5	86.3	28.8	15.5	17.3	31.7	50.1
ComplEx	94.1	58.7	93.6	94.5	94.7	69.2	24.2	59.9	75.9	84.0
SimplE	94.2	58.8	**93.9**	94.4	94.7	72.7	23.9	66.0	77.3	83.8
RotatE	79.7		74.6	83.0	88.4	**94.9**		**94.4**	95.2	95.9
MNTN	**97.0**	74.5	92.4	**95.2**	**97.7**	94.3	**42.0**	88.5	**95.5**	**96.2**

TABLE I: Comparing the accuracy of different relational learning methods for Knowledge Base Completion for the Wordnet and Freebase benchmarks.

RotatE [38] performed slightly better on some metrics for FB15k and SimpleIE on one metric for WN18. Overall, MNTN was competetive with the best system in each case.

C. Results (Object Classification)

The comparison results with several state-of-the-art approaches including Neuro-symbolic models and purely symbolic (SRL) models are shown in Table II. Here, F1 score is the metric used in [44]. The results shown for the other systems are taken from the Graph Markov Neural Nets (GMNN) [44] paper. The other systems used for comparison are as follows. Semi-supervised Learning (SSL) using label propagation [48]. For statistical relational learning (SRL), MLNs along with Probabilistic Relational Model (PRM) [18] and Relational Markov Network(RMN) [39]. For Graph Neural Network based approaches, the methods include Graph Convolutional Networks (GCN) [16], Graph Attention Networks (GAT) [42] and Planetoid [46]. Three relational learning benchmarks were used, Cora, Citeseer and PubMed. On all three benchmarks MNTN had the best accuracy. The next best approach was GMNN but the difference in accuracy between MNTN and GMNN was significant in all three benchmarks. This shows that the use of symmetries indeed plays an important role in relational learning.

Category	Algorithm	Cora	Citeseer	Pubmed
SSL	LP	74.2	56.3	71.6
SRL	PRM	77.0	63.4	68.3
	RMN	71.3	68.0	70.7
	MLN	74.6	68.0	75.3
GNN	Planetoid	75.7	64.7	77.2
	GCN	81.5	70.3	79.0
	GAT	83.0	72.5	79.0
GMNN	Best results	83.7	73.6	81.9
MNTN	Best results	**89.4**	**82.2**	**91.0**

TABLE II: Comparing accuracy for Object Classification on three relational benchmarks. The F1 Score is shown in each case.

Task	MNTN	NTN-R	NTN-CW	AC
Freebase	**94.1**	91.2	90.8	68.2
Wordnet	**89**	86.6	85.6	62.4
Cora	**89.4**	84.1	85.3	68.0
CiteSeer	**82.2**	78.6	80.0	71.7
Pubmed	**91**	84.6	86.3.0	76.8

TABLE III: Ablation Study results for different benchmarks. The ROC-AUC score is shown for Freebase and Wordnet while for the remaining three benchmarks, the F1 score is shown as in [44].

D. Ablation Study

The results of the ablation study are shown in Table III. That is, we compared AC with As seen here AC, NTN-R, NTN-CW and MNTN to analyze the significance of each. As seen by the results, AC consistently performs the worst in terms of accuracy indicating that a simple additive composition of objects is not sufficient to represent symmetries and the value of MNTN is the use of multiplicative composition to represent symmetries. Further, the results for NTN-CW show that exploiting symmetries helps in improving accuracy compared to random negative samples NTN-R. However, training over several possible worlds adds the greatest improvement in performance showing that capturing symmetries over different configurations of the open world atoms adds most value in our system.

E. Results (Image Segmentation and Text Processing)

Results on the image segmentation and text classification tasks are shown in Table IV. The results once again show that MNTN outperformed the other methods including specialized methods such as U-Net for image segmentation. NTN-R and NTN-CW also showed good performance in this case. For

Task	MNTN	NTN-R	NTN-CW	AC	U-Net
Segmentation	**98.2**	97.3	97	77	90
WebKB	**94.3**	92.7	92.5	71	
Yelp	**92.3**	89.7	90.1	61	
Movielens	**94.8**	87	92.5	76	

TABLE IV: Results for Image Segmentation and Text Classification (ROC-AUC).

Task	Tuffy-1	Magician	Tuffy-2
WebKB	61	64	54
Yelp	56	53	53
Movielens	46	51	48

TABLE V: MLNs for Text Classification (ROC-AUC). Tuffy-1 denotes marginal inference results and Tuffy-2 MAP inference (for which the F1 score is mentioned since it produces binary classification).

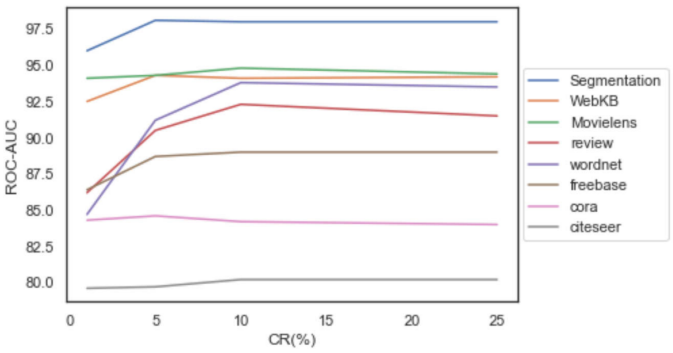


Fig. 4: Accuracy for varying compression ratios (CR). The ROC-AUC score is shown except for object classification where F1 score is shown. Larger CR implies a larger number of clusters are used to cluster the objects in the domain.

image segmentation since spatially distant pixels can be easily randomly sampled as contrasting examples, choosing the right negative examples seemed to have lesser effect here. In the text classification tasks, MNTN outperforms the other methods. The next best approach here was NTN-CW.

The results for the MLN based methods for text classification are also shown in Table V. As seen here, the results have much poorer accuracy than all the other methods. That is, purely symbolic methods such as MLN-based approximate inference methods perform poorly in terms of accuracy. Also, in terms of scalability, we could not process the full datasets here for these methods and sampled them down to approximately 50K atoms per dataset. Thus, MNTN is orders of magnitude more scalable and accurate than MLN methods. Thus, the case for using MLNs as a knowledge representation component and performing the inference and learning through deep neural networks is reinforced with these results.

F. Scalability

Table. VI shows the training time for MNTN and the number of atoms in each of the benchmarks. As seen here, even for large number of atoms, MNTN scales up quite easily. As seen by our results, even for the largest benchmark (Pubmed) which contains over a million atoms, the training time is under 2 hours. In comparison, purely symbolic methods such as MLN-based systems (Magician, Tuffy, etc.) cannot process more than around 50K atoms. Thus, using neural training algorithms with input from symbolic methods is orders of magnitude more scalable. Further, while we have not fully exploited the benefit of pre-trained models here, this is a direction we will explore in future to further improve scalability of our approach.

G. MNTN Representation

Fig. 4 shows the change in accuracy as we change the number of clusters in training the model. In Fig. 4, we show results for different CR(Compression Ratio) which is $\max_d \frac{c_d}{|\Delta_d|}$, where d represents a domain, Δ_d is the number of objects d and c_d is the number of clusters. As we increase CR, it implies that the number of clusters are correspondingly increased. This means that for a small CR, since the number of clusters are small, asymmetric atoms may be part of the same cluster. On the other hand, for a very large CR, since the number of clusters is large, symmetric atoms may be part of different clusters. As shown, increasing CR increases accuracy initially forcing similar representations for asymmetric atoms and results plateau out or may become worse since we may learn diverse representations for symmetrical atoms for large CR.

Finally, we illustrate the representation learned by MNTN with an example case. Fig. 5 shows a 2-D visualization using t-SNE of the atom embeddings from the *Yelp* data. Fig. 5 (a) represents the embeddings learned using AC and Fig. 5 (b) represents the embeddings learned using MNTN. As seen in the results, AC does not a very distinctive pattern to separate the atoms (of different assignments). On the other hand, MNTN seems to learn the atom representations such that there is better structure and separation between symmetric groups of atoms.

H. Hyperparameters

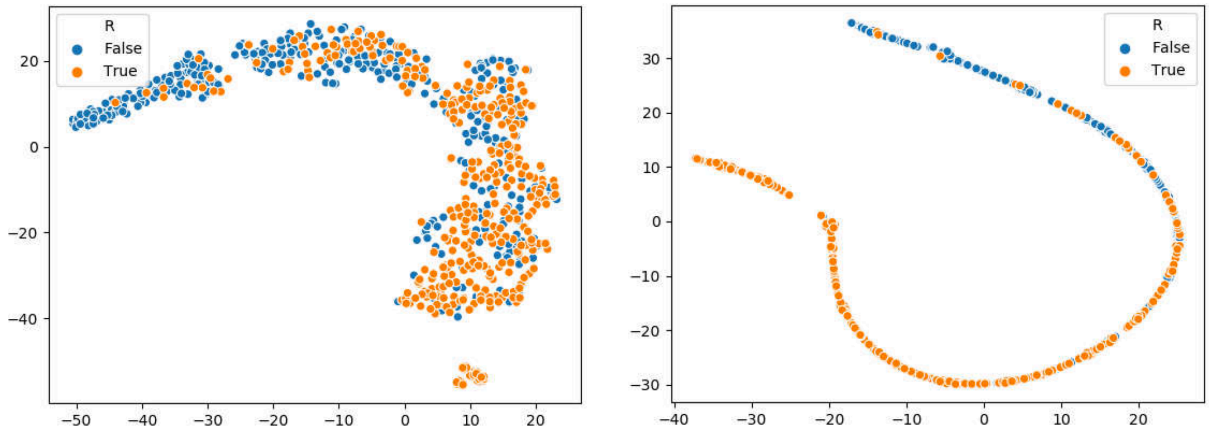
Table VII shows the accuracy for different settings in our architecture. Specifically, we vary the activation functions, epochs, dropout rate and the tensor slices. As seen here, a dropout rate of around 0.5 with 50 epochs gave us the best performance in most of the datasets. The tanh activations performed better than the ReLU activations in general. Compared to the other parameters, changing the tensor slices affected accuracy more significantly. This is expected since the tensors are responsible for learning to represent symmetries between atoms. As seen by these results, for a small number of tensors, the accuracy was lower and this increased with the number of tensor slices which indicates that the increased capacity due to the larger number of tensors allowed MNTN to represent symmetries better.

VI. Conclusion

We developed a Neuro-Symbolic model that learns to predict unknown facts from knowledge bases where the knowledge base also encodes first-order logic rules. Using the

	Cora	*Movielens*	*Segmentation*	*Wordnet*	*Freebase*	*Yelp*	*Citeseer*	*Pubmed*
#Atoms	250K	269K	445K	463K	567K	653K	700K	1.2M
Training Time (mins)	22	23	45	60	58	63	71	110

TABLE VI: Scalability of MNTN. Number of atoms in benchmarks and Training time is shown.



(a) AC (b) MNTN

Fig. 5: Illustrating T-SNE visualization of atom embeddings (with truth values shown by different colors) for *Yelp* data. (a) shows the pattern of embeddings learned for AC and (b) for MNTN. The representation for atoms learned by MNTN shows a more well defined smooth pattern.

	Activation		Epochs			Dropout			Tensor slices		
Benchmark	ReLU	tanh	10	20	50	0.3	0.5	0.7	4	8	16
WN18	94.3	97.0	93.9	96.8	97.0	97.0	96.8	88.1	86.5	96.2	97.0
FB15k	92.8	94.3	89.6	93.0	94.3	94.2	93.7	86.1	88.2	93.9	94.3
Cora	83.5	89.4	83.8	88.5	89.4	89.2	89.0	82.7	83.5	89.4	89.4
Citeseer	81.5	82.2	76.5	81.4	82.2	82.2	82.2	78.9	82.0	82.2	82.2
Pubmed	85.7	91.0	87.7	89.8	91.0	91.0	91.0	83.5	84.8	90.2	91.0
Movielens	94.4	94.8	88.4	93.0	94.5	94.5	94.1	87.3	87.8	94.4	94.4
Yelp	88.1	92.3	85.3	88.7	92.3	92.2	92.3	84.4	88.1	92.3	92.3
webkb	94.0	94.3	94.0	93.3	93.5	94.2	94.2	93.7	94.2	94.2	94.2

TABLE VII: Comparing different hyper-parameter settings. We show the ROC-AUC scores except for object classification where we show F1 scores.

semantics of MLNs for representing the knowledge base, we trained a Neural Tensor Network (NTN) to learn contrastive representations for asymmetric atoms. Since several predictions made by our model may be related to one another, to quantify uncertainty in our predictions, we trained the NTN over multiple possible worlds. In each world, the NTN tensors learn to encode symmetries specific to that world. Our empirical results on varied types of datasets and problems clearly illustrated that our approach is general, highly scalable to large real-world benchmarks and outperforms state-of-the-art relational learning methods as well as Neuro-Symbolic methods in several benchmark problems.

In future, we plan to extend our approach to generative Neuro-symbolic learning. Further, we will also extend our work to generate interpretable predictions based on the formulas specified by the knowledge base.

REFERENCES

[1] Ankit Anand, Aditya Grover, Mausam, and Parag Singla. Contextual symmetries in probabilistic graphical models. In *IJCAI*, pages 3560–3568, 2016.

[2] Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. Learning structured embeddings of knowledge bases. In *AAAI*, pages 301–306, 2011.

[3] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *NeurIPS*, pages 2787–2795, 2013.

[4] Artur S. d'Avila Garcez, Marco Gori, Luís C. Lamb, Luciano Serafini, Michael Spranger, and Son N. Tran. Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning. *FLAP*, 6(4):611–632, 2019.

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

[6] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In *KDD*, pages 601–610. ACM, 2014.

[7] Robert Gens and Pedro M. Domingos. Deep symmetry networks. In *NeurIPS*, pages 2537–2545, 2014.

[8] L. Getoor and B. Taskar, editors. *Introduction to Statistical Relational Learning*. MIT Press, 2007.

[9] V. Gogate and P. Domingos. Probabilistic Theorem Proving. In *UAI*, pages 256–265, 2011.

[10] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4):19:1–19:19, 2016.

[11] Mohammad Maminur Islam, Somdeb Sarkhel, and Deepak Venugopal. Learning mixtures of mlns. In *AAAI*, pages 6359–6366. AAAI Press, 2018.

[12] Mohammad Maminur Islam, Somdeb Sarkhel, and Deepak Venugopal. On lifted inference using neural embeddings. In *AAAI*, pages 7916–7923, 2019.

[13] Mohammad Maminur Islam, Somdeb Sarkhel, and Deepak Venugopal. Augmenting deep learning with relational knowledge from markov logic networks. In *IEEE BigData*, pages 54–63. IEEE, 2020.

[14] Seyed Mehran Kazemi and David Poole. Simple embedding for link prediction in knowledge graphs. In *NeurIPS*, pages 4289–4300, 2018.

[15] Tushar Khot, Niranjan Balasubramanian, Eric Gribkoff, Ashish Sabharwal, Peter Clark, and Oren Etzioni. Exploring markov logic networks for question answering. In *EMNLP*, pages 685–694, 2015.

[16] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

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

[18] Daphne Koller and Avi Pfeffer. Probabilistic frame-based systems. In *AAAI*, 1998.

[19] Luís C. Lamb, Artur S. d'Avila Garcez, Marco Gori, Marcelo O. R. Prates, Pedro H. C. Avelar, and Moshe Y. Vardi. Graph neural networks meet neural-symbolic computing: A survey and perspective. In *IJCAI*, pages 4877–4884, 2020.

[20] B. Leibe, A. Leonardis, and B. Schiele. Combined object categorization and segmentation with an implicit shape model. In *Statistical Learning in Computer Vision*, 2004.

[21] Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Deepproblog: Neural probabilistic logic programming. *CoRR*, abs/1805.10872, 2018.

[22] Giuseppe Marra and Ondrej Kuzelka. Neural markov logic networks. *CoRR*, abs/1905.13462, 2019.

[23] George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, 1995.

[24] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. Factorizing YAGO: scalable machine learning for linked data. In *WWW Conference*, pages 271–280, 2012.

[25] Mathias Niepert and Pedro M. Domingos. Exchangeable variable models. In *ICML*, pages 271–279, 2014.

[26] Feng Niu, Christopher Ré, AnHai Doan, and Jude W. Shavlik. Tuffy: Scaling up statistical inference in markov logic networks using an rdbms. *PVLDB*, 4(6):373–384, 2011.

[27] Meng Qu, Yoshua Bengio, and Jian Tang. GMNN: graph markov neural networks. In *ICML*, pages 5241–5250, 2019.

[28] Shebuti Rayana and Leman Akoglu. Yelp Dataset for Anomalous Reviews. Technical report, Stony Brook University, 2015. http://odds.cs.stonybrook.edu.

[29] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *LREC*, pages 45–50, 2010.

[30] Tim Rocktäschel and Sebastian Riedel. End-to-end differentiable proving. In *NIPS*, pages 3791–3803, 2017.

[31] O. Ronneberger, P.Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, volume 9351, pages 234–241, 2015.

[32] Somdeb Sarkhel, Deepak Venugopal, Parag Singla, and Vibhav Gogate. Lifted MAP inference for Markov Logic Networks. In *AISTATS*, 2014.

[33] Luciano Serafini and Artur S. d'Avila Garcez. Logic tensor networks: Deep learning and logical reasoning from data and knowledge. In *NeSy@HLAI*, volume 1768 of *CEUR Workshop Proceedings*, 2016.

[34] Anup Shakya, Vasile Rus, and Deepak Venugopal. Student strategy prediction using a neuro-symbolic approach. In *Fourteenth International Conference on Educational Data Mining*, 2021.

[35] P. Smolensky. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artif. Intell.*, 46(1-2):159–216, 1990.

[36] Padhraic Smyth and David Wolpert. Stacked density estimation. In *NeurIPS*, 1998.

[37] Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. Reasoning with neural tensor networks for knowledge base completion. In *NeurIPS*, pages 926–934, 2013.

[38] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complec space. In *ICLR*, 2019.

[39] B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. In *UAI*, 2002.

[40] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *ICML*, pages 2071–2080, 2016.

[41] Guy van den Broeck and Adnan Darwiche. On the complexity and approximation of binary evidence in lifted inference. In *NeurIPS*, pages 2868–2876, 2013.

[42] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *ICLR (Poster)*, 2018.

[43] D. Venugopal, S.Sarkhel, and V. Gogate. Magician. Technical report, The University of Memphis, Memphis, TN, 2016. https://github.com/dvngp/CD-Learn.

[44] Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van den Broeck. A semantic loss function for deep learning with symbolic knowledge. In *ICML*, pages 5498–5507, 2018.

[45] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. *ICLR*, 2015.

[46] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *ICML*, pages 40–48, 2016.

[47] Yuyu Zhang, Xinshi Chen, Yuan Yang, Arun Ramamurthy, Bo Li, Yuan Qi, and Le Song. Efficient probabilistic logic reasoning with graph neural networks. In *ICLR*, 2020.

[48] Dengyong Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *NeurIPS*, 2003.