Leveraging Representation and Inference through Deep Relational Learning

Maria Leonor Pacheco
Dept. of Computer Science
Purdue University
pachecog@purdue.edu

Ibrahim Dalal
Indian Institute of Technology
Hyderabad
cs14btech11041@iith.ac.in

Dan Goldwasser
Dept. of Computer Science
Purdue University
dgoldwas@purdue.edu

Abstract

We present DRaiL, a declarative framework for specifying deep relational models. Our framework provides an easy-to-use interface for defining complex models consisting of many interdependent variables, and experimenting with different design choices, learning algorithms and neural architectures. We demonstrate the importance of correctly modeling the interactions between learning, representation and inference by applying DRaiL to two challenging relational learning problems, combining textual and social information. Then, we introduce a relational zero-shot learning task and explore the deep component of DRaiL to work in this setting.

Dealing with structured data, such as conversational interactions (e.g., debates), and social and information networks, requires modeling the interactions between different data points and decisions defined over them. Traditionally, there are two approaches for doing this: (1) *graphical models*, that capture the interactions between variables using probabilistic inference; and (2) *end-to-end deep learning* techniques, in which these dependencies are represented in a latent high-dimensional space. Graphical models explicitly define how the model decomposes, resulting in an interpretable models which can incorporate constraints capturing relevant world knowledge. On the other hand, deep learning approaches capture structural dependencies in a latent space, resulting in expressive classifiers that are often difficult to interpret and constrain according to domain knowledge.

In an effort to combine these two trends, we present DRaiL, a declarative modeling language for defining deep structured prediction problems. A DRaiL program consists of a set of rules, each defining a factor template over a set of variables. Each rule is associated with a neural architecture used to learn its scoring function, and a feature representation definition, which describes how the model variables are represented. From a modeling perspective, DRaiL resembles other declarative languages such as Markov Logic Networks [Domingos *et al.*, 2016] and Probabilistic Soft Logic [Bach *et al.*, 2015]. It uses first order logic as a template language for defining factor graphs. In DRaiL, the scoring function for each factor can be learned using highly expressive models; unlike the other frameworks, which assume a fixed representation.

Previous efforts to combine deep learning with structured inference [Chen and Manning, 2014; Durrett and Klein, 2015; Andor *et al.*, 2016; Lample *et al.*, 2016; Kiperwasser and Goldberg, 2016] combine the two approaches in an ad hoc way, focusing on application-specific properties. For example, training a linear-chain CRF with neural nets as *emission* potentials, while using simple *transition* probabilities. We designed DRaiL to answer the need for a more general approach to study the interactions between inference and representation.

1 DRaiL Overview

A task in DRaiL is defined by specifying a finite set of *entities* and *predicates*. Decisions are defined using rule *templates*, formatted as horn clauses: $r_{LH} \Rightarrow r_{RH}$, where r_{LH} (*body*) is a conjunction

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

of observations and predicted values, and r_{RH} (*head*) is the output value to be predicted. Each rule template t is associated with a neural architecture, defined over a parameter set $\mathbf{w}^t$, to learn its scoring function, and a feature function ϕ^t, mapping the initial observations to an input vector for the neural net. The predicates in the horn clauses can correspond to hidden or observed information, and a specific input is defined by the instantiations -or *groundings*- of these elements. Each rule grounding is comprised of a feature representation $\phi^t(r_{LHi})$ and an output value r_{RHi}. The collection of all rule groundings represents our global decision, taking into account the consistency and dependencies between the rules using a set of constraints C. We define rule activations over Boolean variables r_i^t for each rule grounding, indicating whether they are active or not. The final prediction corresponds to the collection of heads in active rule groundings.

$$\mathbf{R}^* = \arg\max_{r_i^t} \quad \sum_i r_i^t \cdot \text{score}(\phi^t(r_{LHi}), \mathbf{w^t}, r_{RHi})$$
$$\text{subject to C}, \qquad \forall i; r_i^t \in \{0,1\} \tag{1}$$
$$\mathbf{y} = \bigcup_{r_i^t \in = \mathbf{R}^*} r_{RHi} \tag{2}$$

We include two different protocols to learn how to score rules: *local learning*, in which the scoring function corresponding to each rule template is optimized independently of the other templates, and *global learning*, which optimizes the parameters of all the scoring functions jointly. Neural models are specified using Pytorch, and different architectures can be used to represent different factors in both learning protocols. After scoring factors, we assign values to the output variables by running an inference procedure. Currently, we formulate inference as an Integer Linear Program (ILP). The details of the ILP formulation are outlined in appendix A.

Local Learning Each rule is treated as an independent learning problem, and their associated network parameters are optimized separately.

Global Learning Different rules are combined into a globally normalized model, which uses inference to ensure a globally consistent decision. We use the structured hinge loss and the error is back-propagated to update the parameters in all networks.

$$L_G = \sum_j \max_{\mathbf{y} \in Y}(\Delta(\mathbf{y}, \mathbf{y}_j) + \sum_{r_i^t \in \mathbf{R}^*} \text{score}(\phi^t(r_{LHi}), \mathbf{w}^t, r_{RHi})) - \sum_{r_i^t \in \mathbf{G}} \text{score}(\phi^t(r_{LHi}), \mathbf{w}^t, r_{RHi}) \tag{3}$$

Here $\mathbf{y}_j$, is the gold prediction for input instance j, $\mathbf{G}$ is the set of gold rule activations, and Y denotes all possible predictions.

2 Model and Evaluation

We design our experimental evaluation to demonstrate the modeling capabilities of DRAIL for three challenging relational problems. First, we define two typical collective classification tasks: predicting paper topics in a citation network, and predicting user stances on a set of known political issues in a debate social network. Then, we formulate a zero-shot learning task for predicting the stance of posts written in political debates.

2.1 Collective Classification

Datasets For the topic prediction task, we use *Citeseer-M10* [Lim and Buntine, 2016], consisting of 10,310 publications from 10 distinct areas, and 77,218 citation links. For the stance prediction task, we sampled 17,588 users, their profiles, stances and friends from *debate.org*. The friendship graph contains 139,428 edges. Users have stances (i.e. *pro* or *con*) in a number of political issues. For brevity, we focus on Marriage Equality. We represent these tasks using rules of the form:

1. **rule**: InGraph(a,g) ⇒ HasLabel(a,x)
2. **rule**: InGraph(a,g) & InGraph(b,g) & Edge(a,b) & HasLabel(b,y) ⇒ HasLabel(a,x)

The first rule is a simple local rule that maps a node to a label (e.g. Paper ⇒ Topic, User ⇒ Stance). The second rule exploits the relational dependency between nodes a and b (e.g. Citation ⇒ Topic, Friendship ⇒ Stance). We experimented with different initial representations, as well as ways to combine them using neural architectures and global inference.

Initial Representations and Neural Architectures We explored different initial representations to represent both textual and relational information. *DeepWalk* uses the network structure [Perozzi *et al.*, 2014], *Doc2Vec* embeds the text of papers and profiles [Le and Mikolov, 2014], and *TriDNR* combines text and network information through its coupled neural model [Pan *et al.*, 2016]. For all neural models, we use a feed-forward net with one hidden layer and 50 hidden units. An outline of this architecture can be observed in Appendix B, Figure 1.

As a baseline, we used the initial representations directly as inputs to the neural models. The best macro F1 scores were obtained by concatenating Doc2Vec and Deepwalk, 0.585 for predicting paper topics, and 0.556 for predicting user stances. For papers, we were able to improve these results using TriDNR, obtaining 0.690. More results can be observed in Appendix C, Table 3.

Then, we define the collective dependencies described above using DRAIL and experiment with different rule compositions. We write local and relational rules, and use inference to make a global decision. DRAIL allows us to combine different representations either in the same neural network or by exploiting inference. For example, we can write a single rule that takes both the document embedding and the node embedding as an input (D2V$_a$; DW$_a$) to represent a node a. On the other hand, we can write several rules for the same dependency. For example, using different neural networks to represent the edge (a, b): one for the text information (D2V$_{a;b}$), and one for the node embeddings (DW$_{a;b}$). We highlight this flexibility in our experiments and compare different compositions, using the same neural architectures as the baseline methods. These results can be observed in Table 1.

Paper ⇒ T	**Citation ⇒ T**	**Macro F1**	
		Inf	**Gl**
D2V$_a$; DW$_a$	D2V$_{a;b}$; DW$_{a;b}$	0.592	0.550
D2V$_a$ DW$_a$	D2V$_{a;b}$ DW$_{a;b}$	0.561	0.606
TDNR$_a$	TDNR$_{a;b}$	0.707	0.705
D2V$_a$ DW$_a$ TDNR$_a$	D2V$_{a;b}$ DW$_{a;b}$ TDNR$_{a;b}$	0.675	**0.716**

User ⇒ S	**Friendship ⇒ S**	**F1 (Gl)**
D2V$_a$	D2V$_{a;b}$	0.567
DWf$_a$	DW$_{a;b}$	0.572
D2V$_a$; DW$_a$	D2V$_{a;b}$; DW$_{a;b}$	0.563
D2V$_a$ DWf$_a$	D2V$_{a;b}$ DW$_{a;b}$	**0.583**

Table 1: Collective classification results

Additionally, we compare local learning with inference only at prediction time (Inf), and global learning (Gl). Due to space constraints, we limit the discussion of learning protocols to the citation network. All other results use global learning. Evaluation was done with 5-fold cross validation. Using inference, either at prediction time or during learning, consistently outperforms the local methods. We also observe that combining different networks for different representations when using global learning consistently improves performance.

2.2 Zero-Shot Debate Stance Prediction

Traditionally, relational learning approaches use symbolic representations of the data. In Section 2.1 we experimented with a distributed representation for input instances, in this section we suggest that the same can be done for the *output* space. Instead of using categories known a priori, we take a zero-shot learning approach, and learn a distributed representation for the output space as well. The learned representation allows us to capture the semantic relatedness between newly unobserved classes and the classes seen during training. We take advantage of DRAIL's neural architecture and define this process in a relational setting.

To help make the discussion concrete, consider the binary stance classification task of predicting the stance (i.e. *pro* or *con*) of a post written in an online debate, similar to the one we discussed in the previous section. In that case, our model learns to predict stances on a specific topic, such as Marriage Equality (ME) or Gun Control (GC). The decision rule for ME can be expressed as: `InDebate(p,d)` ⇒ `HasStance`$_{\texttt{ME}}$`(p,x)`. Predicting stances for GC would require learning a new rule from scratch: `InDebate(p,d)` ⇒ `HasStance`$_{\texttt{GC}}$`(p,x)`. Instead, we would like to leverage the fact that stances in these domains are highly correlated, and suggest a zero-shot set up, which can be expressed succinctly as: `InDebate(p,d) & Issue(d,t)` ⇒ `HasStance(p,x)`. In this setup, prediction depends on learning a good embedding for instances of the relation `Issue(d,t)`. To study these settings we consider an extreme case, in which each debate defines its own topic and the output space is defined by embedding the *title* of the debate.

There are some straight-forward structural constraints in this setup, as posts written by the same author in a single debate will have the same stance. Likewise, posts written by the instigator and posts written by the contender will have opposing stances. We represent the task as follows:

1. **rule**: InDebate(p,d) & HasTitle(d,t) ⇒ HasStance(p,x)
2. **rule**: InDebate(p,d) & IsAuthor(p,a) & HasTitle(d,t) ⇒ HasStance(p,x)
3. **constr:** InDebate(p,d) & InDebate(o,d) & SameAuthor(p,o) & HasStance(p,x) ⇒ HasStance(o,x)
4. **constr:** InDebate(p,d) & InDebate(o,d) & !SameAuthor(p,o) & HasStance(p,x) ⇒ !HasStance(o,x)

Dataset We collected 21,794 political debates from *debate.org*, containing a total of 162,037 posts. All debates have at least two posts, and posts contain between one and 25 sentences.

Initial Representations and Neural Architectures We used a very simple initial representation for sentences and titles, averaging their pre-trained word vectors [Mikolov *et al.*, 2013]. For users, we use a 1-hot representation of their profile attributes (e.g: gender, occupation, religion, etc). Since many users don't disclose all of their information, these features are very sparse. For the first rule (NN_1), we represent the post using a BiLSTM over its sentences, and the title using a feed-forward net. We concatenate the hidden layer of the title with the last hidden state of the BiLSTM. For the second rule (NN_2), we represent both the title and the user using feed-forward computations, and concatenate their hidden layers. We define a third architecture (NN_3) as an alternative representation for rule 2, where we concatenate the hidden layers of the title and user with the last hidden state of the post. In all cases, we add a softmax on top to predict the stance. These architectures are outlined in Appendix B, Figures 3, 2 and 4. When we combine the two rules using inference, we share the title parameters between the two networks.

Given that each one of the 21,794 debates has a different title, the standard leave-one-out zero-shot learning setup would be too expensive to compute. We divide the debates randomly into five folds and perform 5-fold cross validation.

Model	**Accuracy**	
		+ Inf
NN_1(Post & Title ⇒ S)	0.5611	0.6645
NN_2(User & Title ⇒ S)	0.5514	0.5759
NN_3 (Post & User & Title ⇒ S)	0.5791	0.5793

(a) Neural models with and without constraints

Model	**Protocol**	**Acc**
NN_1(Post & Title ⇒ S) NN_2(User & Title ⇒ S)	Inference	0.6721
	Global (Struct. Loss)	**0.7289**

(b) Combining different representations

Table 2: Zero-Shot Debate Stance Prediction

Table 2 summarizes our results. In Table 2a, we use different representations and architectures with and without inference. As expected, enforcing the debate structural constraints consistently improves results, and is particularly helpful in the "Post & Title ⇒ Stance" case. Our intuition is that some posts are easier to disambiguate than others, and inference allows them to dominate the prediction. However, the best results are obtained when we not only enforce the debate structural constraints, but also combine the two representations (i.e. post and users) using inference, as observed in Table 2b. Using different rules for different representations is better than combining representations using the neural architecture only, even when just using inference. The best results for this task are obtained when using global learning, demonstrating the advantage of having a globally normalized model using the structured loss.

3 Conclusion

This paper presents DRAIL[1], a declarative framework for learning dependencies in relational domains using deep learning architectures. We have demonstrated the flexibility of our framework, which can be used for quick prototyping and evaluation of the interplay between representation complexity and structural complexity. Our current work looks into learning representations using DRAIL through multitask learning, and by explicitly defining embedding objectives, as well as incorporating approximate inference algorithms into the framework to address computational constraints.

[1]Demo: `https://gitlab.com/purdueNlp/DRaiL_Public`

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. Globally normalized transition-based neural networks. In *ACL*, pages 2442–2452, 2016.

Stephen H. Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. Hinge-loss markov random fields and probabilistic soft logic. *Computing Research Repository*, arXiv:1505.04406, 2015.

Danqi Chen and Christopher Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750, 2014.

Pedro Domingos, Daniel Lowd, Stanley Kok, Aniruddh Nath, Hoifung Poon, Matthew Richardson, and Parag Singla. Unifying logical and statistical ai. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '16, pages 1–11, New York, NY, USA, 2016. ACM.

Greg Durrett and Dan Klein. Neural crf parsing. In *ACL*, pages 302–312, 2015.

Eliyahu Kiperwasser and Yoav Goldberg. Simple and accurate dependency parsing using bidirectional lstm feature representations. *TACL*, 2016.

Guillaume Lample, Miguel Ballesteros, Kazuya Kawakami, Sandeep Subramanian, and Chris Dyer. Neural architectures for named entity recognition. In *NAACL*, pages 1–10, 2016.

Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II–1188–II–1196. JMLR.org, 2014.

Kar Wai Lim and Wray L. Buntine. Bibliographic analysis with the citation network topic model. *CoRR*, abs/1609.06826, 2016.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013.*, 2013.

Shirui Pan, Jia Wu, Xingquan Zhu, Chengqi Zhang, and Yang Wang. Tri-party deep network representation. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 1895–1901. AAAI Press, 2016.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 701–710, New York, NY, USA, 2014. ACM.

A ILP Formulation

In this section we define our 1-0 ILP formulation. For each rule grounding i, we introduce rule variables r_i. Each r_i has an associated score $s_i \in \mathbb{R}$. The objective function can then be expressed as:

$$\begin{aligned} &\arg\max_{r_i} \textstyle\sum_i s_i \cdot r_i \\ &\text{subject to C} \qquad \forall i; r_i \in \{0,1\} \end{aligned} \tag{4}$$

Then, we introduce head variables h_j for each different head predicate j (and its negation $(\bar{h}_j)$) to indicate the activation of the variable. As in the case of rule variables, head variables can exclusively take values of 0 or 1 ($\forall j; h_j \in \{0,1\}$). Note that in the objective function, we do not assign any weight to head variables, as their values are entirely determined by constraints that ensure consistency.

In order to enforce consistency between variable assignments and dependencies among them, the following five types of constraints are taken into consideration.

negation constraints They ensure exclusive activation of a head predicate or its negation at the same time. For example, $h_{\texttt{HasLabel(a,b)}} + \bar{h}_{\texttt{HasLabel(a,b)}} = 1$.

implied constraints Each rule template defines the dependency between body and head. This dependency is reflected between the rule grounding variable and the head variables in the body. For example, consider a rule grounding i: `Edge(a,b) & HasLabel(b,c)` $\Rightarrow$ `HasLabel(a,d)`, where a and b are observed nodes, and c and d are labels. In this case, the constraint $r_i \leq h_{\texttt{HasLabel(b,c)}}$ is needed, as the whole rule is true only when the body is activated.

rule/head constraints One head predicate can be associated with multiple rule grounding variables. Let $r_i \in ruleset(j)$ denote the rule variables associated with the same head variable h_j, where $ruleset(j)$ is the set of rule groundings that share the same head predicate j. Activation of any rules in $ruleset(j)$ ensures the activation of the head variable, i.e. $h_j \geq r_i, \forall i \in ruleset(j)$. On the other hand, the activation of the head variable ensures the activation of at least one of its corresponding rule variables, i.e. $h_j \leq \sum_i r_i$.

binary/multi-class/multi-label constraints In many problems, we are facing multi-class or multi-label decisions. DRAIL allows us to express this and add suitable constraints. For instance, in the multi-class case, among all head variables $h_j \in decision(d)$ on the same entity (e.g. all possible labels assignments for a single node), only one of them is activated while the others remain inactive, i.e. $\sum_j h_j = 1$. Note that the constraints for binary predicates can be covered by the negation constraints mentioned above.

hard constraints from rule definitions Users can define hard constraints in the rule templates, which usually infuse prior knowledge and thus improve the prediction capacity. Rule groundings of these templates are dealt differently as the activation of such a rule depends on the activation of all body predicates. In this paper, we showed two examples of such constraints: 1) `InDebate(p,d) & InDebate(o,d) & SameAuthor(p,o) & HasStance(p,x)` $\Rightarrow$ `HasStance(o,x)`, to enforce authors to have consistent opinions across posts in a debate, and 2) `InDebate(p,d) & InDebate(o,d) & !SameAuthor(p,o) & HasStance(p,x)` $\Rightarrow$ `!HasStance(o,x)`, to enforce debating users to have opposing opinions in a debate.

B Neural Architectures

The following architectures were used in this paper. The feed-forward architecture on Figure 1 was used for all factors in the collective classification tasks. Figures 2, 3 and 4 show the architectures used for the Zero-Shot debate stance prediction task.

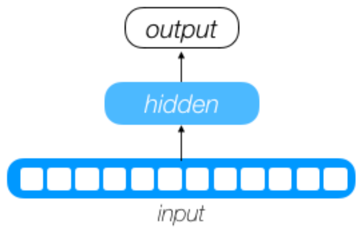

Figure 1: Feed-Forward Network

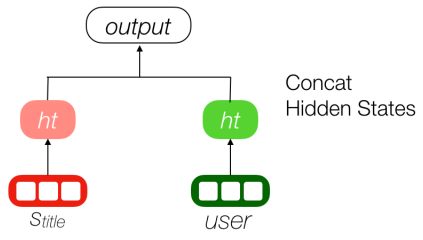

Figure 2: N2(User+Title ⇒ Stance)

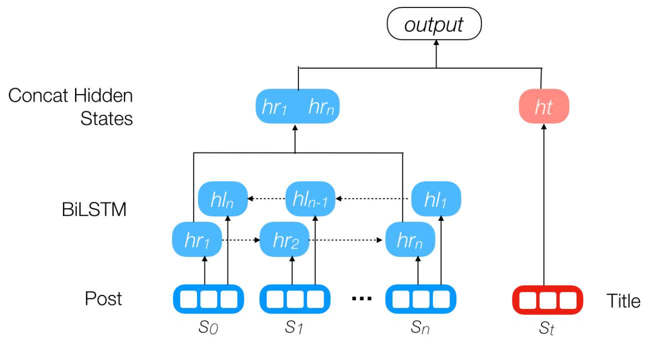

Figure 3: N1(Post+Title ⇒ Stance)

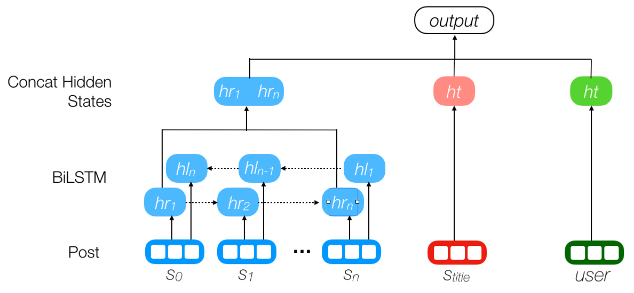

Figure 4: N3(Post+User+Title ⇒ Stance)

C Other Results

Node	Macro F1	
	Papers	**Users**
Doc2Vec	0.522	0.551
DeepWalk	0.394	0.551
Doc2Vec + DeepWalk	0.585	**0.556**
TriDNR	**0.690**	-

Table 3: Baseline representations for predicting paper topics and user stances, using neural architecture in Appendix B, Figure 1 and no inference