Modeling Content and Context with Deep Relational Learning

Maria Leonor Pacheco and Dan Goldwasser

Department of Computer Science
Purdue University
West Lafayette, IN 47907
{pachecog, dgoldwas}@purdue.edu

Abstract

Building models for realistic natural language tasks requires dealing with long texts and accounting for complicated structural dependencies. Neural-symbolic representations have emerged as a way to combine the reasoning capabilities of symbolic methods, with the expressiveness of neural networks. However, most of the existing frameworks for combining neural and symbolic representations have been designed for classic relational learning tasks that work over a universe of symbolic entities and relations. In this paper, we present DRaiL, an open-source declarative framework for specifying deep relational models, designed to support a variety of NLP scenarios. Our framework supports easy integration with expressive language encoders, and provides an interface to study the interactions between representation, inference and learning.

1 Introduction

Understanding natural language interactions in realistic settings requires models that can deal with noisy textual inputs, reason about the dependencies between different textual elements, and leverage the dependencies between textual content and the context from which it emerges. Work in linguistics and anthropology has defined context as a frame that surrounds a focal communicative event and provides resources for its interpretation (Gumperz, 1992; Duranti and Goodwin, 1992).

As a motivating example, consider the interactions in the debate network described in Figure 1. Given a debate claim (t_1), and two consecutive posts debating it (p_1, p_2), we define a textual inference task, determining whether a pair of text elements hold the same stance in the debate (denoted using the relation $\texttt{Agree}(\texttt{X}, \texttt{Y})$). This task is similar to other textual inference tasks (Bowman et al., 2015) that have been successfully approached using complex neural representations (Peters et al., 2018; Devlin et al., 2019). In addition, we can leverage the dependencies between these decisions. For example, assuming that one post agrees with the debate claim ($\texttt{Agree}(\texttt{t}_1, \texttt{p}_2)$), and the other one does not ($\neg\texttt{Agree}(\texttt{t}_1, \texttt{p}_1)$), the disagreement between the two posts can be inferred: $\neg\texttt{Agree}(\texttt{t}_1, \texttt{p}_1) \wedge \texttt{Agree}(\texttt{t}_1, \texttt{p}_2) \rightarrow \neg \texttt{Agree}(\texttt{p}_1, \texttt{p}_2)$. Finally, we consider the *social context* of the text. The disagreement between the posts can reflect a difference in the perspectives their authors hold on the issue. This information might not be directly observed, but it can be inferred using the authors' social interactions and behavior, given the principle of social homophily (McPherson et al., 2001), stating that people with strong social ties are likely to hold similar views and authors' perspectives can be captured by representing their social interactions. Exploiting this information requires models that can align the social representation with the linguistic one.

Motivated by these challenges, we introduce DRaiL[1], a Deep Relational Learning framework, which uses a combined neuro-symbolic representation for modeling the interaction between multiple decisions in relational domains. Similar to other neuro-symbolic approaches (Mao et al., 2019; Cohen et al., 2020), our goal is to exploit the complementary strengths of the two modeling paradigms. Symbolic representations, used by logic-based systems and by probabilistic graphical models (Richardson and Domingos, 2006; Bach et al., 2017), are interpretable, and allow domain experts to directly inject knowledge and constrain the learning problem. Neural models capture dependencies using the network architecture and are better equipped to deal with noisy data, such as text. However, they are often difficult to interpret and constrain according to domain knowledge.

[1] https://gitlab.com/purdueNlp/DRaiL/.

Transactions of the Association for Computational Linguistics, vol. 9, pp. 100–119, 2021. https://doi.org/10.1162/tacl_a_00357
Action Editor: Hoifung Poon. Submission batch: 6/2020; Revision batch: 10/2020; Published 3/2021.

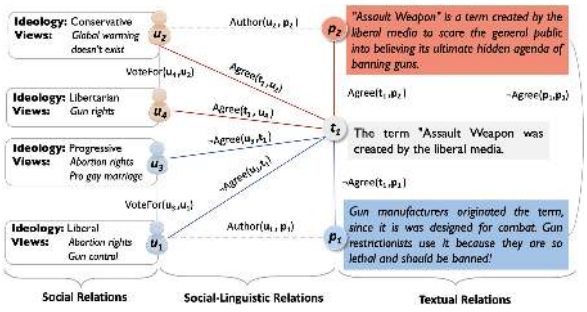


Figure 1: Example debate.

Our main design goal in DRAIL is to provide a generalized tool, specifically designed for NLP tasks. Existing approaches designed for classic relational learning tasks (Cohen et al., 2020), such as knowledge graph completion, are not equipped to deal with the complex linguistic input, whereas others are designed for very specific NLP settings such as word-based quantitative reasoning problems (Manhaeve et al., 2018) or aligning images with text (Mao et al., 2019). We discuss the differences between DRAIL and these approaches in Section 2. The examples in this paper focus on modelings various argumentation mining tasks and their social and political context, but the same principles can be applied to wide array of NLP tasks with different contextualizing information, such as images that appear next to the text, or prosody when analyzing transcribed speech, to name a few examples.

DRAIL uses a declarative language for defining deep relational models. Similar to other declarative languages (Richardson and Domingos, 2006; Bach et al., 2017), it allows users to inject their knowledge by specifying dependencies between decisions using first-order logic rules, which are later compiled into a factor graph with neural potentials. In addition to probabilistic inference, DRAIL also models dependencies using a *distributed knowledge representation*, denoted RELNETS, which provides a shared representation space for entities and their relations, trained using a relational multi-task learning approach. This provides a mechanism for explaining symbols, and aligning representations from different modalities. Following our running example, ideological standpoints, such as `Liberal` or `Conservative`, are discrete entities embedded in the same space as textual entities and social entities. These entities are initially associated with users, however using RELNETS this information will propagate to texts reflecting these ideologies, by exploiting the relations that bridge social and linguistic information (see Figure 1).

To demonstrate DRAIL's modeling approach, we introduce the task of *open-domain stance prediction with social context*, which combines social network analysis and textual inference over complex opinionated texts, as shown in Figure 1. We complement our evaluation of DRAIL with two additional tasks, issue-specific stance prediction, where we identify the views expressed in debate forums with respect to a set of fixed issues (Walker et al., 2012), and argumentation mining (Stab and Gurevych, 2017), a document-level discourse analysis task.

2 Related Work

In this section, we survey several lines of work dealing with symbolic, neural, and hybrid representations for relational learning.

2.1 Languages for Graphical Models

Several high-level languages for specifying graphical models have been suggested. BLOG (Milch et al., 2005) and CHURCH (Goodman et al., 2008) were suggested for generative models. For discriminative models, we have Markov Logic Networks (MLNs) (Richardson and Domingos, 2006) and Probabilistic Soft Logic (PSL) (Bach et al., 2017). Both PSL and MLNs combine logic and probabilistic graphical models in a single representation, where each formula is associated with a weight, and the probability distribution over possible assignments is derived from the weights of the formulas that are satisfied by such assignments. Like DRAIL, PSL uses formulas in clausal form (specifically collections of horn clauses). The main difference between DRAIL and these languages is that, in addition to graphical models, it uses distributed knowledge representations to represent dependencies. Other discriminative methods include FACTORIE (McCallum et al., 2009), an imperative language to define factor graphs, Constrained Conditional Models (CCMs) (Rizzolo and Roth, 2010; Kordjamshidi et al., 2015) an interface to enhance linear classifiers with declarative constraints, and ProPPR (Wang et al., 2013) a probabilistic logic for large databases that approximates local groundings using a variant of personalized PageRank.

2.2 Node Embedding and Graph Neural Nets

A recent alternative to graphical models is to use neural nets to represent and learn over relational data, represented as a graph. Similar to DRaiL's RelNets, the learned node representation can be trained by several different prediction tasks. However, unlike DRaiL, these methods do not use probabilistic inference to ensure consistency.

Node embeddings approaches (Perozzi et al., 2014; Tang et al., 2015; Pan et al., 2016; Grover and Leskovec, Grover and Leskovec, 2016; Tu et al., 2017) learn a feature representation for nodes capturing graph adjacency information, such that the similarity in the embedding space of any two nodes is proportional to their graph distance and overlap in neighboring nodes. Some frameworks (Pan et al., 2016; Xiao et al., 2017; Tu et al., 2017) allow nodes to have textual properties, which provide an initial feature representation when learning to represent the graph relations. When dealing with multi-relational data, such as knowledge graphs, both the nodes and the edge types are embedded (Bordes et al., 2013; Wang et al., 2014; Trouillon et al., 2016; Sun et al., 2019). Finally, these methods learn to represent nodes and relations based on pair-wise node relations, without representing the broader graph context in which they appear. Graph neural nets (Kipf and Welling, 2017; Hamilton et al., 2017; Veličković et al., 2017) create contextualized node representations by recursively aggregating neighboring nodes.

2.3 Hybrid Neural-Symbolic Approaches

Several recent systems explore ways to combine neural and symbolic representations in a unified way. We group them into five categories.

Lifted rules to specify compositional nets. These systems use an end-to-end approach and learn relational dependencies in a latent space. Lifted Relational Neural Networks (LRNNs) (Sourek et al., 2018) and RelNNs (Kazemi and Poole, 2018) are two examples. These systems map observed ground atoms, facts, and rules to specific neurons in a network and define composition functions directly over them. While they provide for a modular abstraction of the relational inputs, they assume all inputs are symbolic and do not leverage expressive encoders.

Differentiable inference. These systems identify classes of logical queries that can be compiled into differentiable functions in a neural network infrastructure. In this space we have Tensor Logic Networks (TLNs) (Donadello et al., 2017) and TensorLog (Cohen et al., 2020). Symbols are represented as row vectors in a parameter matrix. The focus is on implementing reasoning using a series of numeric functions.

Rule induction from data. These systems are designed for inducing rules from symbolic knowledge bases, which is not in the scope of our framework. In this space we find Neural Theorem Provers (NTPs) (Rocktäschel and Riedel, 2017), Neural Logic Programming (Yang et al., 2017), DRUM (Sadeghian et al., 2019) and Neural Logic Machines (NLMs) (Dong et al., 2019). NTPs use a declarative interface to specify rules that add inductive bias and perform soft proofs. The other approaches work directly over the database.

Deep classifiers and probabilistic inference. These systems propose ways to integrate probabilistic inference and neural networks for diverse learning scenarios. DeepProbLog (Manhaeve et al. 20180 extends the probabilistic logic programming language ProbLog to handle neural predicates. They are able to learn probabilities for atomic expressions using neural networks. The input data consists of a combination of feature vectors for the neural predicates, together with other probabilistic facts and clauses in the logic program. Targets are only given at the output side of the probabilistic reasoner, allowing them to learn each example with respect to a single query. On the other hand, Deep Probabilistic Logic (DPL) (Wang and Poon 2018) combines neural networks with probabilistic logic for indirect supervision. They learn classifiers using neural networks and use probabilistic logic to introduce distant supervision and labeling functions. Each rule is regarded as a latent variable, and the logic defines a joint probability distribution over all labeling decisions. Then, the rule weights and the network parameters are learned jointly using variational EM. In contrast, DRaiL focuses on learning multiple interdependent decisions from data, handling and requiring supervision for all unknown atoms in a given example. Lastly, Deep Logic Models (DLMs) (Marra et al., 2019) learn a set of parameters to encode atoms in a probabilistic logic program. Similarly to Donadello et al. (2017)

System	Symbolic Features					Neural Features					Open Source
	Symbolic Inputs	Raw Inputs	Declarative	Prob/Logic Inference	Rule Induction	Embed. Symbols	End-to-end Neural	Backprop. to Encoders	Architecture Agnostic	Multi-Task Learning	
MLN	✓		✓	✓							✓
FACTORIE	✓			✓							✓
CCM	✓	✓	✓	✓							✓
PSL	✓		✓	✓							✓
LRNNs	✓					✓	✓				
RelNNs	✓					✓	✓				✓
LTNs	✓		✓	✓		✓	✓				✓
TensorLog	✓		✓	✓		✓	✓				✓
NTPs	✓		✓	✓	✓	✓	✓				✓
Neural LP	✓				✓	✓	✓				✓
DRUM	✓				✓	✓	✓				✓
NLMs	✓				✓	✓	✓				✓
DeepProbLog	✓	✓	✓	✓	✓	✓		✓	✓		✓
DPL	✓	✓	✓	✓		✓		✓	✓		✓
DLMs	✓	✓	✓	✓		✓	✓	✓	✓		
DRAIL	✓	✓	✓	✓		✓		✓	✓	✓	✓

Table 1: Comparing systems.

and Cohen et al. (2020), they use differentiable inference, allowing the model to be trained end-to-end. Like DRAIL, DLMs can work with diverse neural architectures and backpropagate back to the base classifiers. The main difference between DLMs and DRAIL is that DRAIL ensures representation consistency of entities and relations across all learning tasks by employing RELNETS.

Deep structured models. More generally, deep structured prediction approaches have been successfully applied to various NLP tasks such as named entity recognition and dependency parsing (Chen and Manning, 2014; Weiss et al., 2015; Ma and Hovy, 2016; Lample et al., 2016; Kiperwasser and Goldberg, 2016; Malaviya et al., 2018). When the need arises to go beyond sentence-level, some works combine the output scores of independently trained classifiers using inference (Beltagy et al., 2014; **?**; Liu et al., 2016; Subramanian et al., 2017; Ning et al., 2018), whereas others implement joint learning for their specific domains (Niculae et al., 2017; Han et al., 2019). Our main differentiating factor is that we provide a general interface that leverages first order logic clauses to specify factor graphs and express constraints.

To summarize these differences, we outline a feature matrix in Table 1. Given our focus in NLP tasks, we require a neural-symbolic system that (1) allows us to integrate state-of-the-art text encoders and NLP tools, (2) supports structured prediction across long texts, (3) lets us combine several modalities and their representations (e.g., social and textual information), and (4) results in an explainable model where domain constraints can be easily introduced.

3 The DRAIL Framework

DRAIL was designed for supporting complex NLP tasks. Problems can be broken down into domain-specific atomic components (which could be words, sentences, paragraphs or full documents, depending on the task), and dependencies between them, their properties and contextualizing information about them can be explicitly modeled.

In DRAIL, dependencies can be modeled over the predicted output variables (similar to other probabilistic graphical models), as well as over the neural representation of the atoms and their relationships in a shared embedding space. This section explains the framework in detail. We begin with a high-level overview of DRAIL and the process of moving from a declarative definition to a predictive model.

A DRAIL task is defined by specifying a finite set of *entities* and *relations*. Entities are either discrete symbols (e.g., POS tags, ideologies, specific issue stances), or attributed elements with complex internal information (e.g., documents, users). Decisions are defined using rule *templates*, formatted as Horn clauses: $t_{LH} \Rightarrow t_{RH}$, where t_{LH} (*body*) is a conjunction of observed and predicted relations, and t_{RH} (*head*) is the output relation to be learned. Consider the debate prediction task in Figure 1, it consists of several sub-tasks, involving textual inference ($\texttt{Agree}(\texttt{t}_1, \texttt{t}_2)$), social relations ($\texttt{VoteFor}(\texttt{u}, \texttt{v})$) and their combination ($\texttt{Agree}(\texttt{u}, \texttt{t})$). We illustrate how

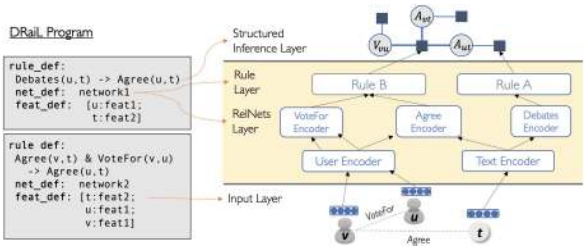


Figure 2: General overview of DRAIL.

to specify the task as a DRAIL program in Figure 2 (left), by defining a subset of rule templates to predict these relations.

Each rule template is associated with a neural architecture and a feature function, mapping the initial observations to an input vector for each neural net. We use a shared *relational* embedding space, denoted RELNETS, to represent entities and relations over them. As described in Figure 2 ("RelNets Layer"), each entity and relation type is associated with an encoder, trained jointly across all prediction rules. This is a form of relational multi-task learning, as the same entities and relations are reused in multiple rules and their representation is updated accordingly. Each rule defines a neural net, learned over the relations defined on the body. They they take a composition of the vectors generated by the relations encoders as an input (Figure 2, "Rule Layer"). DRAIL is architecture-agnostic, and neural modules for entities, relations and rules can be specified using PyTorch (code snippets can be observed in Appendix C). Our experiments show that we can use different architectures for representing text, users, as well as for embedding discrete entities.

The relations 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. The collection of all rule groundings results in a factor graph representing our global decision, taking into account the consistency and dependencies between the rules. This way, the final assignments can be obtained by running an inference procedure. For example, the dependency between the views of users on the debate topic ($\texttt{Agree}(\texttt{u},\texttt{t})$) and the agreement between them ($\texttt{VoteFor}(\texttt{u},\texttt{v})$), is modeled as a factor graph in Figure 2 ("Structured Inference Layer")).

We formalize the DRAIL language in Section 3.1. Then, in Sections 3.2, 3.3, and 4, we describe the neural components and learning procedures.

3.1 Modeling Language

We begin our description of DRAIL by defining the templating language, consisting of entities, relations, and rules, and explaining how these elements are instantiated given relevant data.

Entities are named *symbolic* or *attributed* elements. An example of a symbolic entity is a political ideology (e.g., Liberal or Conservative). An example of an attributed entity is a user with age, gender, and other profile information, or a document associated with textual content. In DRAIL entities can appear either as *constants*, written as strings in double or single quote (e.g., `"user1"`) or as *variables*, which are identifiers, substituted with constants when grounded. Variables are written using unquoted upper case strings (e.g., `X`, `X1`). Both constants and variables are typed.

Relations are defined between entities and their properties, or other entities. Relations are defined using a unique identifier, a named *predicate*, and a list of typed arguments. **Atoms** consist of a predicate name and a sequence of entities, consistent with the type and arity of the relation's argument list. If the atom's arguments are all constants, it is referred to as a **ground atom**. For example, $\texttt{Agree}(\texttt{"user1"},\texttt{"user2"})$ is a ground atom representing whether `"user1"` and `"user2"` are in agreement. When atoms are not grounded (e.g., $\texttt{Agree}(\texttt{X},\texttt{Y})$) they serve as placeholders for all the possible groundings that can be obtained by replacing the variables with constants. Relations can either be *closed* (i.e., all of their atoms are observed) or *open*, when some of the atoms can be unobserved. In DRAIL, we use a question mark `?` to denote unobserved relations. These relations are the units that we reason over.

To help make these concepts concrete, consider the following example analyzing stances in a debate, as introduced in Figure 1. First, we define the entities. $\texttt{User} = \{\texttt{"u1"},\texttt{"u2"}\}, \texttt{Claim} = \{\texttt{"t1"}\}$ $\texttt{Post} = \{\texttt{"p1"},\texttt{"p2"}\}$. Users are entities associated with demographic attributes and preferences. Claims are assertions over which users debate. Posts are textual arguments that users write to explain their position with respect to the claim. We create these associations by

defining a set of relations, capturing authorship $\texttt{Author}(\texttt{User}, \texttt{Post})$, votes between users $\texttt{VoteFor}(\texttt{User}, \texttt{User})?$, and the position users, and their posts, take with respect to to the debate claim. $\texttt{Agree}(\texttt{Claim}, \texttt{User})?$, $\texttt{Agree}(\texttt{Claim}, \texttt{Post})?$. The authorship relation is the only closed one, for example, the atom: $\mathcal{O} = \{\texttt{Author}(\texttt{"u1"}, \texttt{"p1"})\}$.

Rules are functions that map literals (atoms or their negation) to other literals. Rules in DRAIL are defined using templates formatted as Horn clauses: $t_{LH} \Rightarrow t_{RH}$, where t_{LH} (*body*) is a conjunction of literals, and t_{RH} (*head*) is the output literal to be predicted, and can only be an instance of open relations. Horn clauses allow us to describe structural dependencies as a collection of "if-then" rules, which can be easily interpreted. For example, $\texttt{Agree}(\texttt{X}, \texttt{C}) \wedge \texttt{VoteFor}(\texttt{Y}, \texttt{X}) \Rightarrow \texttt{Agree}(\texttt{Y}, \texttt{C})$ expresses the dependency between votes and users holding similar stances on a specific claim. We note that rules can be rewritten in *disjunctive form* by converting the logical implication into a disjunction between the negation of the body and the head. For example, the rule above can be rewritten as $\neg\texttt{Agree}(\texttt{X}, \texttt{C}) \vee \neg\texttt{VoteFor}(\texttt{Y}, \texttt{X}) \vee \texttt{Agree}(\texttt{Y}, \texttt{C})$.

The DRAIL program consists of a set of rules, which can be weighted (i.e., soft constraints), or unweighted (i.e., hard constraints). Each *weighted rule* template defines a learning problem, used to score assignments to the head of the rule. Because the body may contain open atoms, each rule represents a factor function expressing dependencies between open atoms in the body and head. Unweighted rules, or *constraints*, shape the space of feasible assignments to open atoms, and represent background knowledge about the domain.

Given the set of grounded atoms $\mathcal{O}$, rules can be grounded by substituting their variables with constants, such that the grounded atoms correspond to elements in $\mathcal{O}$. This process results in a set of grounded rules, each corresponding to a potential function or to a constraint. Together they define a factor graph. Then, DRAIL finds the optimally scored assignments for open atoms by performing MAP inference. To formalize this process, we first make the observation that rule groundings can be written as linear inequalities, directly corresponding to their disjunctive form, as follows:

$$\sum_{i \in I_r^+} y_i + \sum_{i \in I_r^-} (1 - y_i) \geq 1 \tag{1}$$

Where I_r^+ (I_r^-) correspond to the set of open atoms appearing in the rule that are not negated (respectively, negated). Now, MAP inference can be defined as a linear program. Each rule grounding r, generated from template $t(r)$, with input features $\boldsymbol{x_r}$ and open atoms $\boldsymbol{y_r}$ defines the potential

$$\psi_r(\boldsymbol{x_r}, \boldsymbol{y_r}) = \min \left\{ \sum_{i \in I_r^+} y_i + \sum_{i \in I_r^-} (1 - y_i), 1 \right\} \tag{2}$$

added to the linear program with a weight w_r. Unweighted rule groundings are defined as

$$c(\mathbf{x_c}, \mathbf{y_c}) = 1 - \sum_{i \in I_c^+} y_i - \sum_{i \in I_c^-} (1 - y_i) \tag{3}$$

with $c(\boldsymbol{x_c}, \boldsymbol{y_c}) \leq 0$ added as a constraints to the linear program. This way, the MAP problem can be defined over the set of all potentials Ψ and the set of all constraints C as

$$\arg\max_{\boldsymbol{y} \in \{0,1\}^n} P(\boldsymbol{y}|\boldsymbol{x}) \equiv \arg\max_{\boldsymbol{y} \in \{0,1\}^n} \sum_{\psi_{r,t} \in \Psi} w_r \, \psi_r(\boldsymbol{x_r}, \boldsymbol{y_r})$$
$$\textit{such that } c(\boldsymbol{x_c}, \boldsymbol{y_c}) \leq 0; \;\; \forall c \in C$$

In addition to logical constraints, we also support arithmetic constraints than can be written in the form of linear combinations of atoms with an inequality or an equality. For example, we can enforce the mutual exclusivity of liberal and conservative ideologies for any user $\texttt{X}$ by writing:

$$\texttt{Ideology}(\texttt{X}, \texttt{"con"}) + \texttt{Ideology}(\texttt{X}, \texttt{"lib"}) = 1$$

We borrow some additional syntax from PSL to make arithmetic rules easier to use. Bach et al. (2017) define a *summation atom* as an atom that takes terms and/or sum variables as arguments. A summation atom represents the summations of ground atoms that can be obtained by substituting individual variables and summing over all possible constants for sum variables. For example, we could rewrite the above ideology constraint as $\texttt{Ideology}(\texttt{X}, +\texttt{I}) = 1$, where $\texttt{Ideology}(\texttt{X}, +\texttt{I})$ represents the summation of all atoms with predicate $\texttt{Ideology}$ that share variable $\texttt{X}$.

DRAIL uses two solvers, Gurobi (Gurobi Optimization, 2015) and AD3 (Martins et al., 2015) for exact and approximate inference, respectively.

To ground DRAIL programs in data, we create an in-memory database consisting of all relations

expressed in the program. Observations associated with each relation are provided in column separated text files. DRAIL's compiler instantiates the program by automatically querying the database and grounding the formatted rules and constraints.

3.2 Neural Components

Let r be a rule *grounding* generated from *template* t, where t is tied to a neural scoring function Φ_t and a set of parameters θ_t (Rule Layer in Figure 2). In the previous section, we defined the MAP problem for all potentials $\psi_r(\boldsymbol{x}, \boldsymbol{y}) \in \Psi$ in a DRAIL program, where each potential has a weight w_r. Consider the following scoring function:

$$w_r = \Phi_t(\boldsymbol{x_r}, \boldsymbol{y_r}; \theta^t) = \Phi_t(x_{\text{rel}_0}, \ldots, x_{\text{rel}_{n-1}}; \theta^t) \tag{4}$$

Notice that all potentials generated by the same template share parameters. We define each scoring function Φ_t over the set of atoms on the left hand side of the rule template. Let $t = \text{rel}_0 \wedge \text{rel}_1 \wedge \ldots \wedge \text{rel}_{n-1} \Rightarrow \text{rel}_n$ be a rule template. Each atom rel_i is composed of a relation type, its arguments and feature vectors for them, as shown in Figure 2, "Input Layer".

Given that a DRAIL program is composed of many competing rules over the same problem, we want to be able to share information between the different decision functions. For this purpose, we introduce RELNETS.

3.3 RELNETS

A DRAIL program often uses the same entities and relations in multiple different rules. The symbolic aspect of DRAIL allows us to constrain the values of open relations, and force consistency across all their occurrences. The neural aspect, as defined in Eq. 4, associates a neural architecture with each rule template, which can be viewed as a way to embed the output relation.

We want to exploit the fact that there are repeating occurrences of entities and relations across different rules. Given that each rule defines a learning problem, sharing parameters allows us to shape the representations using complementary learning objectives. This form of relational multi-task learning is illustrated it in Figure 2, "RelNets Layer".

We formalize this idea by introducing relation-specific and entity-specific encoders and their parameters $(\phi_{\text{rel}}; \theta^{\text{rel}})$ and $(\phi_{\text{ent}}; \theta^{\text{ent}})$, which are reused in all rules. As an example, let's write the formulation for the rules outlined in Figure 2, where each relation and entity encoder is defined over the set of relevant features.

$$w_{r_0} = \Phi_{t_0}(\phi_{\text{debates}}(\phi_{\text{user}}, \phi_{\text{text}}))$$
$$w_{r_1} = \Phi_{t_1}(\phi_{\text{agree}}(\phi_{\text{user}}, \phi_{\text{text}}), \phi_{\text{votefor}}(\phi_{\text{user}}, \phi_{\text{user}}))$$

Note that entity and relation encoders can be arbitrarily complex, depending on the application. For example, when dealing with text, we could use BiLSTMs or a BERT encoder.

Our goal when using RELNETS is to learn entity representations that capture properties unique to their types (e.g., users, issues), as well as relational patterns that contextualize entities, allowing them to generalize better. We make the distinction between *raw* (or *attributed*) entities and *symbolic* entities. Raw entities are associated with rich, yet unstructured, information and attributes, such as text or user profiles. On the other hand, symbolic entities are well-defined concepts, and are not associated with additional information, such as political ideologies (e.g., *liberal*) and issues (e.g., *gun-control*). With this consideration, we identify two types of representation learning objectives:

Embed Symbol / Explain Data: Aligns the embedding of symbolic entities and raw entities, grounding the symbol in the raw data, and using the symbol embedding to explain properties of previously unseen raw-entity instances. For example, aligning ideologies and text to (1) obtain an ideology embedding that is closest to the statements made by people with that ideology, or (2) interpret text by providing a symbolic label for it.

Translate / Correlate: Aligns the representation of pairs of symbolic or raw entities. For example, aligning user representations with text, to move between social and textual information, as shown in Figure 1, "Social-Linguistic Relations". Or capturing the correlation between symbolic judgements like agreement and matching ideologies.

4 Learning

The scoring function used for comparing output assignments can be learned *locally* for each rule separately, or *globally*, by considering the dependencies between rules.

Global Learning The global approach uses inference to ensure that the parameters for all weighted rule templates are consistent across all decisions. Let Ψ be a factor graph with potentials $\{\psi_r\} \in \Psi$ over the all possible structures Y. Let $\boldsymbol{\theta} = \{\theta^t\}$ be a set of parameter vectors, and $\Phi_t(\boldsymbol{x_r}, \boldsymbol{y_r}; \theta^t)$ be the scoring function defined for potential $\psi_r(\boldsymbol{x_r}, \boldsymbol{y_r})$. Here $\hat{\mathbf{y}} \in Y$ corresponds to the current prediction resulting from the MAP inference procedure and $\mathbf{y} \in Y$ corresponds to the gold structure. We support two ways to learn $\boldsymbol{\theta}$:

(1) The structured hinge loss

$$\max(0, \max_{\hat{\mathbf{y}} \in Y}(\Delta(\hat{\mathbf{y}}, \mathbf{y}) + \sum_{\psi_r \in \Psi} \Phi_t(\boldsymbol{x_r}, \hat{\boldsymbol{y}}_{\boldsymbol{r}}; \theta^t)) - \sum_{\psi_r \in \Psi} \Phi_t(\boldsymbol{x_r}, \boldsymbol{y_r}; \theta^t) \tag{5}$$

(2) The general CRF loss

$$-\log p(\mathbf{y}|\mathbf{x}) = -\log \left(\frac{1}{Z(\mathbf{x})} \prod_{\psi_r \in \Psi} \exp\left\{\Phi_t(\boldsymbol{x_r}, \boldsymbol{y_r}; \theta^t)\right\} \right) = -\sum_{\psi_r \in \Psi} \Phi_t(\boldsymbol{x_r}, \boldsymbol{y_r}; \theta^t) + \log Z(\mathbf{x}) \tag{6}$$

Where $Z(\mathbf{x})$ is a global normalization term computed over the set of all valid structures Y.

$$Z(\mathbf{x}) = \sum_{\mathbf{y'} \in Y} \prod_{\psi_r \in \Psi} \exp\left\{\Phi_t(\boldsymbol{x_r}, \boldsymbol{y'_r}; \theta^t)\right\}$$

When inference is intractable, approximate inference (e.g., AD^3) can be used to obtain $\hat{\boldsymbol{y}}$. To approximate the global normalization term $Z(\boldsymbol{x})$ in the general CRF case, we follow Zhou et al. (2015); Andor et al. (2016) and keep a pool β_k of k of high-quality feasible solutions during inference. This way, we can sum over the solutions in the pool to approximate the partition function $\sum_{\mathbf{y'} \in \beta_k} \prod_{\psi_r \in \Psi} \exp\left\{\Phi_t(\boldsymbol{x_r}, \boldsymbol{y'_r}; \theta^t)\right\}$.

In this paper, we use the structured hinge loss for most experiments, and include a discussion on the approximated CRF loss in Section 5.7.

Joint Inference The parameters for each weighted rule template are optimized independently. Following Andor et al. (2016), we show that joint inference serves as a way to greedily approximate the CRF loss, where we replace the normalization term in Eq. (6) with a greedy approximation over local normalization as:

$$-\log \left(\frac{1}{\prod_{\psi_r \in \Psi} Z_L(\boldsymbol{x_r})} \prod_{\psi_r \in \Psi} \exp\left\{\Phi_t(\boldsymbol{x_r}, \boldsymbol{y_r}; \theta^t)\right\} \right) = -\sum_{\psi_r \in \Psi} \Phi_t(\boldsymbol{x_r}, \boldsymbol{y_r}; \theta^t) + \sum_{\psi_r \in \Psi} \log Z_L(\boldsymbol{x_r}) \tag{7}$$

where $Z_L(\boldsymbol{x_r})$ is computed over all the valid assignments $\boldsymbol{y'_r}$ for each factor ψ_r. We refer to models that use this approach as JOINTINF.

$$Z_L(\mathbf{x}_r) = \sum_{\boldsymbol{y'_r}} \exp\left\{\Phi_t(\boldsymbol{x_r}, \boldsymbol{y'_r}; \theta^t)\right\}$$

5 Experimental Evaluation

We compare DRAIL to representative models from each category covered in Section 2. Our goal is to examine how different types of approaches capture dependencies and what are their limitations when dealing with language interactions. These baselines are described in Section 5.1. We also evaluate different strategies using DRAIL in Section 5.2.

We focus on three tasks: open debate stance prediction (Sec. 5.3), issue-specific stance prediction (Sec. 5.4) and argumentation mining (Sec. 5.5), details regarding the hyper-parameters used for all tasks can be found in Appendix B.

5.1 Baselines

End-to-end Neural Nets: We test all approaches against neural nets trained locally on each task, without explicitly modeling dependencies. In this space, we consider two variants: INDNETS, where each component of the problem is represented using an independent neural network, and E2E, where the features for the different components are concatenated at the input and fed to a single neural network.

Relational Embedding Methods: Introduced in Section 2.2, these methods embed nodes and edge types for relational data. They are typically designed to represent symbolic entities and relations. However, because our entities can be defined by raw textual content and other features, we define the relational objectives over our encoders. This adaptation has proven successful for domains dealing with rich textual information (Lee and Goldwasser, 2019). We test three

relational knowledge objectives: TransE (Bordes et al., 2013), ComplEx (Trouillon et al., 2016), and RotatE (Sun et al., 2019). **Limitations:** (1) These approaches cannot constrain the space using domain knowledge, and (2) they cannot deal with relations involving more than two entities, limiting their applicability to higher order factors.

Probabilistic Logics: We compare to PSL (Bach et al., 2017), a purely symbolic probabilistic logic, and TensorLog (Cohen et al., 2020), a neuro-symbolic one. In both cases, we instantiate the program using the weights learned with our base encoders. **Limitations:** These approaches do not provide a way to update the parameters of the base classifiers.

5.2 Modeling Strategies

Local vs. Global Learning: The trade-off between local and global learning has been explored for graphical models (MEMM vs. CRF), and for deep structured prediction (Chen and Manning, 2014; Andor et al., 2016; Han et al., 2019). Although local learning is faster, the learned scoring functions might not be consistent with the correct global prediction. Following (Han et al., 2019), we initialize the parameters using local models.

RelNets: We will show the advantage of having relational representations that are shared across different decisions, in contrast to having independent parameters for each rule. Note that in all cases, we will use the global learning objective to train RelNets.

Modularity: Decomposing decisions into relevant modules has been shown to simplify the learning process and lead to better generalization (Zhang and Goldwasser, 2019). We will contrast the performance of *modular* and *end-to-end* models to represent text and user information when predicting stances.

Representation Learning and Interpretability: We will do a qualitative analysis to show how we are able to *embed symbols* and *explain data* by moving between symbolic and sub-symbolic representations, as outlined in Section 3.3.

5.3 Open Domain Stance Prediction

Traditionally, stance prediction tasks have focused on predicting stances on a specific topic, such as abortion. Predicting stances for a different topic, such as gun control, would require learning a new model from scratch. In this task, we would like to leverage the fact that stances in different domains are correlated. Instead of using a pre-defined set of debate topics (i.e., *symbolic* entities) we define the prediction task over claims, expressed in text, specific to each debate. Concretely, each debate will have a different claim (i.e., different value for $\texttt{C}$ in the relation $\texttt{Claim}(\texttt{T},\texttt{C})$, where $\texttt{T}$ corresponds to a debate thread). We refer to these settings as *Open-Domain* and write down the task in Figure 3. In addition to the textual stance prediction problem (r0), where $\texttt{P}$ corresponds to a post, we represent users ($\texttt{U}$) and define a user-level stance prediction problem (r1). We assume that additional users read the posts and vote for content that supports their views, resulting in another prediction problem (r2,r3). Then, we define representation learning tasks, which align symbolic (ideology, defined as $\texttt{I}$) and raw (users and text) entities (r4-r7). Finally, we write down all dependencies and constrain the final prediction (c0-c7).

// Predictions
r0: $\texttt{InThread}(\texttt{T},\texttt{P}) \wedge \texttt{Claim}(\texttt{T},\texttt{C}) \Rightarrow \texttt{Agree}(\texttt{P},\texttt{C})?$
r1: $\texttt{Debates}(\texttt{T},\texttt{U}) \wedge \texttt{Claim}(\texttt{T},\texttt{C}) \Rightarrow \texttt{Agree}(\texttt{U},\texttt{C})?$
r2: $\texttt{Debates}(\texttt{T},\texttt{U}) \wedge \texttt{Votes}(\texttt{T},\texttt{V}) \Rightarrow \texttt{VoteFor}(\texttt{V},\texttt{U})?$
r3: $\texttt{Votes}(\texttt{T},\texttt{V}_1) \wedge \texttt{Votes}(\texttt{T},\texttt{V}_2) \Rightarrow \texttt{VoteSame}(\texttt{V}_1,\texttt{V}_2)?$
// Auxiliary objectives
r4: $\texttt{InThread}(\texttt{T},\texttt{P}) \wedge \texttt{Ideology}(\texttt{I}) \Rightarrow \texttt{HasIdeology}(\texttt{P},\texttt{I})?$
r5: $\texttt{Claim}(\texttt{T},\texttt{C}) \wedge \texttt{Ideology}(\texttt{I}) \Rightarrow \texttt{HasIdeology}(\texttt{C},\texttt{I})?$
r6: $\texttt{Debates}(\texttt{T},\texttt{U}) \wedge \texttt{Ideology}(\texttt{I}) \Rightarrow \texttt{HasIdeology}(\texttt{U},\texttt{I})?$
r7: $\texttt{HasIdeology}(\texttt{A},\texttt{I})? \wedge \texttt{HasIdeology}(\texttt{B},\texttt{I})? \Rightarrow \texttt{Agree}(\texttt{A},\texttt{B})?$

// Author constraints
c0: $\texttt{Agree}(\texttt{P},\texttt{C})? \wedge \texttt{Author}(\texttt{P},\texttt{U}) \Rightarrow \texttt{Agree}(\texttt{U},\texttt{C})?$
c1: $\neg\texttt{Agree}(\texttt{P},\texttt{C})? \wedge \texttt{Author}(\texttt{P},\texttt{U}) \Rightarrow \neg\texttt{Agree}(\texttt{U},\texttt{C})?$
// Debate constraints
c2: $\texttt{Agree}(\texttt{P}_1,\texttt{C})? \wedge \texttt{Respond}(\texttt{P}_1,\texttt{P}_2) \Rightarrow \neg\texttt{Agree}(\texttt{P}_2,\texttt{C})?$
c3: $\neg\texttt{Agree}(\texttt{P}_1,\texttt{C})? \wedge \texttt{Respond}(\texttt{P}_1,\texttt{P}_2) \Rightarrow \texttt{Agree}(\texttt{P}_2,\texttt{C})?$
// Social constraints
c4: $\texttt{Agree}(\texttt{U},\texttt{C})? \wedge \texttt{VoteFor}(\texttt{V},\texttt{U})? \Rightarrow \texttt{Agree}(\texttt{V},\texttt{U})?$
c5: $\neg\texttt{Agree}(\texttt{U},\texttt{C})? \wedge \texttt{VoteFor}(\texttt{V},\texttt{U})? \Rightarrow \neg\texttt{Agree}(\texttt{V},\texttt{U})?$
c6: $\texttt{Agree}(\texttt{V}_1,\texttt{C})? \wedge \texttt{VoteSame}(\texttt{V}_1,\texttt{V}_2)? \Rightarrow \texttt{Agree}(\texttt{V}_2,\texttt{C})?$
c7: $\neg\texttt{Agree}(\texttt{V}_1,\texttt{C})? \wedge \texttt{VoteSame}(\texttt{V}_1,\texttt{V}_2)? \Rightarrow \neg\texttt{Agree}(\texttt{V}_2,\texttt{C})?$

Figure 3: DRaiL Program for O.D. Stance Prediction. T: Thread, C: Claim, P: Post, U: User, V: Voter, I: Ideology, A,B: Can be any in {Claim, Post, User}

Dataset: We collected a set of 7,555 debates from `debate.org`, containing a total of 42,245 posts across 10 broader political issues. For a given issue, the debate topics are nuanced and vary according to the debate question expressed in text (e.g., *Should semi-automatic guns be banned*, *Conceal handgun laws reduce violent crime*). Debates have at least two posts, containing up to 25 sentences each. In addition to debates and posts, we collected the user profiles of all users participating in the debates, as well as all users

	Model	Random			Hard		
		P	U	V	P	U	V
Local	IndNets	63.9	61.3	54.4	62.2	53.0	51.3
	E2E	66.3	71.2	54.4	63.4	68.1	51.3
Reln.	TransE	58.5	54.1	52.6	57.2	53.1	51.2
Emb.	ComplEx	61.0	63.3	**58.1**	57.3	55.0	55.4
	RotatE	59.6	58.3	54.2	57.9	54.6	51.0
Prob.	PSL	78.7	77.5	55.4	72.6	71.8	52.6
Logic.	TensorLog	72.7	71.9	56.2	70.0	67.4	**55.8**
DRaiL	E2E +Inf	80.2	79.2	54.4	76.9	75.5	51.3
	JointInf	80.7	79.5	55.6	75.2	74.0	52.5
	Global	81.0	79.5	55.8	75.3	74.0	53.0
	RelNets	**81.9**	**80.4**	57.0	**78.0**	**77.2**	53.7

	Model	Random			Hard		
		P	U	V	P	U	V
Local	IndNets	63.9	61.3	54.4	62.2	53.0	51.3
	E2E	66.3	71.2	54.4	63.4	68.1	51.3
AC	JointInf	73.6	71.8	-	69.0	67.2	-
	Global	73.6	72.0	-	69.0	67.2	-
	RelNets	73.8	72.0	-	71.7	69.5	-
AC	JointInf	80.7	79.5	-	75.6	74.4	-
DC	Global	81.4	79.9	-	75.8	74.6	-
	RelNets	81.8	80.1	-	77.8	76.4	-
AC	JointInf	80.7	79.5	55.6	75.2	74.0	52.5
DC	Global	81.0	79.5	55.8	75.3	74.0	53.0
SC	RelNets	**81.9**	**80.4**	**57.0**	**78.0**	**77.2**	**53.7**

Table 2: General Results for Open Domain Stance Prediction (Left), Variations of the Model (Right). P:Post, U:User, V:Voter

that cast votes for the debate participants. Profiles consist of attributes (e.g., gender, ideology). User data is considerably sparse. We create two evaluation scenarios, *random* and *hard*. In the random split, debates are randomly divided into ten folds of equal size. In the hard split, debates are separated by political issue. This results in a harder prediction problem, as the test data will not share topically related debates with the training data. We perform 10-fold cross validation and report accuracy.

Entity and Relation Encoders: We represent posts and titles using a pre-trained BERT-small[2] encoder (Turc et al., 2019), a compact version of the language model proposed by Devlin et al. 2019. For users, we use feed-forward computations with ReLU activations over the profile features and a pre-trained node embedding (Grover and Leskovec, 2016) over the friendship graph. All relation and rule encoders are represented as feed-forward networks with one hidden layer, ReLU activations and a softmax on top. Note that all of these modules are updated during learning.

Table 2 (Left) shows results for all the models described in Section 5.1. In E2E models, post and user information is collapsed into a single module (rule), whereas in IndNets, JointInf, Global and RelNets they are modeled separately. All other baselines use the same underlying modular encoders. We can appreciate the advantage of relational embeddings in contrast to IndNets for user and voter stances, particularly in the case of ComplEx and RotatE. We can attribute this to the fact that all objectives are trained jointly and entity encoders are shared. However, approaches that explicitly model inference, like PSL, TensorLog, and DRaiL outperform relational embeddings and end-to-end neural networks. This is because they enforce domain constraints.

We explain the difference between the performance of DRaiL and the other probabilistic logics by: (1) The fact that we use exact inference instead of approximate inference, (2) PSL learns to weight the rules without giving priority to a particular task, whereas the JointInf model works directly over the local outputs, and *most importantly*, (3) our Global and RelNets models back-propagate to the base classifiers and fine-tune parameters using a structured objective.

In Table 2 (Right) we show different versions of the DRaiL program, by adding or removing certain constraints. AC models only enforce author consistency, AC-DC models enforce both author consistency and disagreement between respondents, and finally, AC-DC-SC models introduce social information by considering voting behavior. We get better performance when we model more contextualizing information for the RelNets case. This is particularly helpful in the *Hard* case, where contextualizing information, combined with shared representations, help the model generalize to previously unobserved topics. With respect to the modeling strategies listed in Section 5.2, we can observe: (1) The advantage of using a global learning objective, (2) the advantage of using RelNets to share information and (3) the advantage of breaking down the decision into modules, instead of learning an end-to-end model.

Then, we perform a qualitative evaluation to illustrate our ability to move between symbolic

[2]We found negligible difference in performance between BERT and BERT-small for this task, while obtaining a considerable boost in speed.

Issue	Debate Statements	Con	Libt	Mod	Libl	Pro
Guns	No gun laws should be passed restricting the right to bear arms	**.98**	.01	.00	.01	.00
	Gun control is an ineffective comfort tactic used by the government to fool the American people	.08	**.65**	.22	.02	.03
	Gun control is good for society	.14	.06	**.60**	.15	.06
	In the US handguns ought to be banned	.03	.01	.01	**.93**	.02
	The USA should ban most guns and confiscate them	.00	.00	.01	.00	**.99**

Issue	Ideology	Statements close in the embedding space
LGBT	Libl	gay marriage ought be legalized, gay marriage should be legalized, same-sex marriage should be federally legal
	Con	Leviticus 18:22 and 20:13 prove the anti gay marriage position, gay marriage is not bad, homosexuality is not a sin nor taboo

Table 3: Representation Learning Objectives: Explain Data (Top) and Embed Symbol (Bottom).
Note that ideology labels were learned from user profiles, and do not necessarily represent the official stances of political parties.

and raw information. Table 3 (Top) takes a set of statements and *explains* them by looking at the symbols associated with them and their score. For learning to map debate statements to ideological symbols, we rely on the partial supervision provided by the users that self-identify with a political ideology and disclose it on their public profiles. Note that we do not incorporate any explicit expertise in political science to learn to represent ideological information. We chose statements with the highest score for each of the ideologies. We can see that, in the context of guns, statements that have to do with some form of gun control have higher scores for the center-to-left spectrum of ideological symbols (moderate, liberal, progressive), whereas statements that mention gun rights and the ineffectiveness of gun control policies have higher scores for conservative and libertarian symbols.

To complement this evaluation, in Table 3 (Bottom), we *embed* ideologies and find three example statements that are close in the embedding space. In the context of LGBT issues, we find that statements closest to the liberal symbol are those that support the legalization of same-sex marriage, and frame it as a constitutional issue. On the other hand, the statements closest to the conservative symbol, frame homosexuality and same-sex marriage as a moral or religious issue, and we find statements both supporting and opposing same-sex marriage. This experiment shows that our model is easy to interpret, and provides an explanation for the decision made.

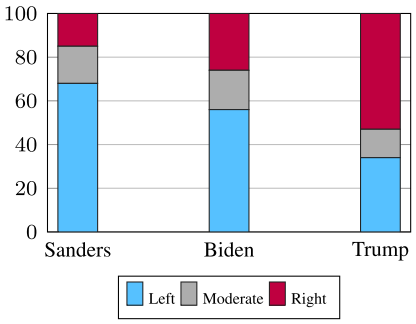
Finally, we evaluate our learned model over entities that have not been observed during training. To do this, we extract statements made by three prominent politicians from *ontheissues.org*. Then, we try to explain the politicians by looking at their predicted ideology. Results for this evaluation can be seen in Table 4. The left part of Figure 4 shows the proportion of statements that were identified for each ideology: left (liberal or progressive), moderate and right (conservative). We find that we are able to recover the relative positions in the political spectrum for the evaluated politicians: Bernie Sanders, Joe Biden, and Donald Trump. We find that Sanders is the most left leaning, followed by Biden. In contrast, Donald Trump stands mostly on the right. We also include some examples of the classified statements. We show that we are able to identify cases in which the statement does not necessarily align with the known ideology for each politician.

	Model	AB		E		GM		GC	
		S	A	S	A	S	A	S	A
Local	IndNets	66.0	61.7	58.2	59.7	62.6	60.6	59.5	61.0
Reln. Embed.	TransE	62.5	62.9	53.5	65.1	58.7	69.3	55.3	65.0
	ComplEx	66.6	73.4	60.7	72.2	66.6	72.8	60.0	70.7
	RotatE	66.6	72.3	59.2	71.3	67.0	74.2	59.4	69.9
Prob. Logic.	PSL	81.6	74.4	69.0	64.9	83.3	74.2	71.9	71.7
	TensorLog	77.3	61.3	68.2	51.3	80.4	65.2	68.3	55.6
DRaiL	JointInf	82.8	74.6	64.8	63.2	84.5	73.4	70.4	66.3
	Global	88.6	**84.7**	72.8	72.2	**90.3**	81.8	76.8	72.2
	RelNets	**89.0**	83.5	**80.5**	**76.4**	89.3	**82.1**	**80.3**	**73.4**

Table 4: General results for issue-specific stance and agreement prediction (Macro F1). AB: Abortion, E: Evolution, GM: Gay Marriage, GC: Gun Control.

5.4 Issue-Specific Stance Prediction

Given a debate thread on a specific issue (e.g., abortion), the task is to predict the stance with respect to the issue for each one of the debate posts (Walker et al., 2012). Each thread forms a tree structure, where users participate and respond to each other's posts. We treat the task as a collective classification problem, and model the agreement between posts and their replies, as well as the consistency between posts written by the



Politician	Issue	Statement	Label
Sanders	Guns	For background checks, and closing loopholes	Left
	Guns	Intervene with mental illness, to prevent mass shootings	Mod
	Abortion	Advocate for family planning and funding for contraceptives	Left
Biden	Guns	Guns need to have trigger locks	Left
	Abortion	Accepts catholic church view that life begins at conception	Right
	Abortion	Ensure access to and funding for contraception	Left
Trump	Guns	No limits on guns; they save lives	Right
	Abortion	I am pro-life; fight ObamaCare abortion funding	Right
	Abortion	Planned Parenthood does great work on women's health	Left

Figure 4: Statements made by politicians classified using our model trained on debate.org.

same author. The DRAIL program for this task can be observed in Appendix A.

Dataset: We use the 4Forums dataset from the Internet Argument Corpus (Walker et al., 2012), consisting of a total of 1,230 debates and 24,658 posts on abortion, evolution, gay marriage, and gun control. We use the same splits as Li et al. (2018) and perform 5-fold cross validation.

Entity and Relation Encoders: We represented posts using pre-trained BERT encoders (Devlin et al., 2019) and do not generate features for authors. As in the previous task, we model all relations and rules using feed-forward networks with one hidden layer and ReLU activations. Note that we fine-tune all parameters during training.

In Table 4 we can observe the general results for this task. We report macro F1 for post stance and agreement between posts for all issues. As in the previous task, we find that ComplEx and RotatE relational embeddings outperform INDNETS, and probabilistic logics outperform methods that do not perform constrained inference. PSL outperforms JOINTINF for evolution and gun control debates, which are the two issues with less training data, whereas JOINTINF outperforms PSL for debates on abortion and gay marriage. This could indicate that re-weighting rules may be advantageous for the cases with less supervision. Finally, we see the advantage of using a global learning objective and augmenting it with shared representations. Table 5 compares our model with previously published results.

Model	A	E	GM	GC	Avg
BERT (Devlin et al., 2019)	67.0	62.4	67.4	64.6	65.4
PSL (Sridhar et al., 2015b)	77.0	80.3	80.5	69.1	76.7
Struct. Rep. (Li et al., 2018)	86.5	82.2	87.6	83.1	84.9
DRAIL RELNETS	**89.2**	**82.4**	**90.1**	**83.1**	**86.2**

Table 5: Previous work on issue-specific stance prediction (stance acc.).

5.5 Argument Mining

The goal of this task is to identify argumentative structures in essays. Each argumentative structure corresponds to a tree in a document. Nodes are predefined spans of text and can be labeled either as *claims, major claims*, or *premises*, and edges correspond to support/attack relations between nodes. Domain knowledge is injected by constraining sources to be premises and targets to be either premises or major claims, as well as enforcing tree structures. We model nodes, links, and second order relations, *grandparent* ($a \rightarrow b \rightarrow c$), and *co-parent* ($a \rightarrow b \leftarrow c$) (Niculae et al., 2017). Additionally, we consider link labels, denoted stances. The DRAIL program for this task can be observed in Appendix A.

Dataset: We used the UKP dataset (Stab and Gurevych, 2017), consisting of 402 documents, with a total of 6,100 propositions and 3,800 links (17% of pairs). We use the splits used by Niculae et al. (2017), and report macro F1 for components and positive F1 for relations.

Entity and Relation Encoders: To represent the component and the essay, we used a BiLSTM over the words, initialized with GloVe embeddings (Pennington et al., 2014), concatenated with a feature vector following Niculae et al. (2017). For representing the relation, we use a feed-forward computation over the components, as well as the relation features used in Niculae et al. (2017).

We can observe the general results for this task in Table 6. Given that this task relies on constructing the tree from scratch, we find that all methods that do not include declarative constraints (INDNETS and relational embeddings) suffer when

	Model	Node	Link	Avg	Stance	Avg
Local	IndNets	70.7	52.8	61.7	63.4	62.3
Reln.	TransE	65.7	23.7	44.7	44.6	44.7
Embed.	ComplEx	69.1	15.7	42.4	53.5	46.1
	RotatE	67.2	20.7	44.0	46.7	44.9
Prob. Logic	PSL	76.5	56.4	66.5	64.7	65.9
	JointInf	78.6	59.5	69.1	62.9	67.0
DRaiL	Global	**83.1**	61.2	72.2	**69.2**	71.2
	RelNets	82.9	**63.7**	**73.3**	68.4	**71.7**

Table 6: General results for argument mining.

Model	Node	Link	Avg
Human upper bound	86.8	75.5	81.2
ILP Joint (Stab and Gurevych, 2017)	82.6	58.5	70.6
Struct RNN strict (Niculae et al., 2017)	79.3	50.1	64.7
Struct SVM full (Niculae et al., 2017)	77.6	60.1	68.9
Joint PointerNet (Potash et al., 2017)	84.9	60.8	72.9
Kuribayashi et al. 2019	**85.7**	**67.8**	**76.8**
DRaiL RelNets	82.9	63.7	73.3

Table 7: Previous work on argument mining.

trying to predict links correctly. For this task, we did not apply TensorLog, given that we couldn't find a way to express tree constraints using their syntax. Once again, we see the advantage of using global learning, as well as sharing information between rules using RelNets.

Table 7 shows the performance of our model against previously published results. While we are able to outperform models that use the same underlying encoders and features, recent work by Kuribayashi et al. (2019) further improved performance by exploiting contextualized word embeddings that look at the whole document, and making a distinction between argumentative markers and argumentative components. We did not find a significant improvement by incorporating their ELMo-LSTM encoders into our framework,[3] nor by replacing our BiLSTM encoders with BERT. We leave the exploration of an effective way to leverage contextualized embeddings for this task for future work.

5.6 Run-time Analysis

In this section, we perform a run-time analysis of all probabilistic logic systems tested. All experiments were run on a 12 core 3.2Ghz Intel i7 CPU machine with 63GB RAM and an NVIDIA GeForce GTX 1080 Ti 11GB GDDR5X GPU.

[3]We did not experiment with their normalization approach, extended BoW features, nor AC/AM distinction.

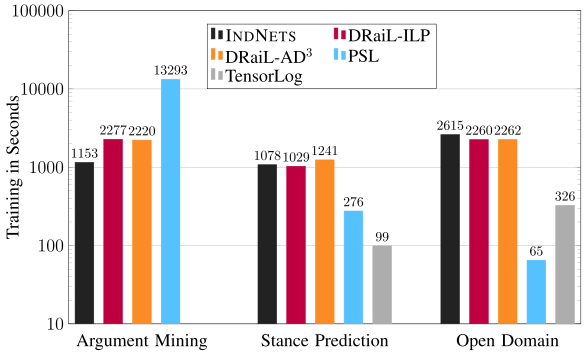


Figure 5: Average overall training time (per fold).

Figure 5 shows the overall training time (per fold) in seconds for each of the evaluated tasks. Note that the figure is presented in logarithmic scale. We find that DRaiL is generally more computationally expensive than both TensorLog and PSL. This is expected given that DRaiL back-propagates to the base classifiers at each epoch, while the other frameworks just take the local predictions as priors. However, when using a large number of arithmetic constraints (e.g., Argument Mining), we find that PSL takes a really long time to train. We found no significant difference when using ILP or AD.[3] We presume that this is due to the fact that our graphs are small and that Gurobi is a highly optimized commercial software.

Finally, we find that when using encoders with a large number of parameters (e.g., BERT) in tasks with small graphs, the difference in training time between training local and global models is minimal. In these cases, back-propagation is considerably more expensive than inference, and global models converge in fewer epochs. For Argument Mining, local models are at least twice as fast. BiLSTMs are considerably faster than BERT, and inference is more expensive for this task.

5.7 Analysis of Loss Functions

In this section we perform an evaluation of the CRF loss for issue-specific stance prediction. Note that one drawback of the CRF loss (Eq. 6) is that we need to accumulate the gradient for the approximated partition function. When using entity encoders with a lot of parameters (e.g., BERT), the amount of memory needed for a single instance increases. We were unable to fit the full models in our GPU. For the purpose of these tests, we froze the BERT parameters after local training

Model	Stance	Agree	Avg	Secs p/epoch
Hinge loss	82.74	78.54	80.64	132
CRF($\beta = 5$)	83.09	81.03	82.06	345
CRF($\beta = 20$)	84.10	82.16	83.13	482
CRF($\beta = 50$)	84.19	81.80	83.00	720

Table 8: Stance prediction (abortion) dev results for different training objectives.

and updated only the relation and rule parameters. To obtain the solution pool, we use Gurobi's pool search mode to find β high-quality solutions. This also increases the cost of search at inference time.

Development set results for the debates on abortion can be observed in Table 8. While increasing the size of the solution pool leads to better performance, it comes at a higher computational cost.

6 Conclusions

In this paper, we motivate the need for a declarative neural-symbolic approach that can be applied to NLP tasks involving long texts and contextualizing information. We introduce a general framework to support this, and demonstrate its flexibility by modeling problems with diverse relations and rich representations, and obtain models that are easy to interpret and expand. The code for DRaiL and the application examples in this paper have been released to the community, to help promote this modeling approach for other applications.

Acknowledgments

We would like to acknowledge current and former members of the PurdueNLP lab, particularly Xiao Zhang, Chang Li, Ibrahim Dalal, I-Ta Lee, Ayush Jain, Rajkumar Pujari, and Shamik Roy for their help and insightful discussions in the early stages of this project. We also thank the reviewers and action editor for their constructive feedback. This project was partially funded by the NSF, grant CNS-1814105.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2442–2452, Berlin, Germany. Association for Computational Linguistics. **DOI:** https://doi.org/10.18653/v1/P16-1231

Stephen H. Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. 2017. Hinge-loss markov random fields and probabilistic soft logic. *Journal of Machine Learning Research (JMLR)*, 181–67.

Islam Beltagy, Katrin Erk, and Raymond Mooney. 2014. Probabilistic soft logic for semantic textual similarity. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1210–1219. Baltimore, Maryland. Association for Computational Linguistics. **DOI:** https://doi.org/10.3115/v1/P14-1114

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013, Translating embeddings for modeling multi-relational data, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2787–2795. Curran Associates, Inc.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642. Lisbon, Portugal, Association for Computational Linguistics. **DOI:** https://doi.org/10.18653/v1/D15-1075

Danqi Chen and Christopher Manning. 2014. 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. Doha, Qatar. Association for Computational Linguistics. **DOI:** https://doi.org/10.3115/v1/D14-1082

William W. Cohen, Fan Yang, and Kathryn Mazaitis. 2020. Tensorlog: A probabilistic

database implemented using deep-learning infrastructure. *Journal of Artificial Intelligence Research*, 67:285–325. **DOI:** https://doi.org/10.1613/jair.1.11944

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Ivan Donadello, Luciano Serafini, and Artur d'Avila Garcez. 2017. Logic tensor networks for semantic image interpretation. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 1596–1602. **DOI:** https://doi.org/10.24963/ijcai.2017/221

Honghua Dong, Jiayuan Mao, Tian Lin, Chong Wang, Lihong Li, and Denny Zhou. 2019. Neural logic machines. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

Alessandro Duranti and Charles Goodwin. 1992. Rethinking context: An introduction, Alessandro Duranti and Charles Goodwin, editors, *Rethinking Context: Language as an Interactive Phenomenon*, chapter 1, pages 1–42, Cambridge University Press, Cambridge.

Noah D. Goodman, Vikash K. Mansinghka, Daniel M. Roy, Keith Bonawitz, and Joshua B. Tenenbaum. 2008. Church: a language for generative models. In *UAI 2008, Proceedings of the 24th Conference in Uncertainty in Artificial Intelligence, Helsinki, Finland, July 9-12, 2008*, pages 220–229.

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 855–864. **DOI:** https://doi.org/10.1145/2939672.2939754, **PMID:** 27853626, **PMCID:** PMC5108654

John J. Gumperz. 1992. Contextualization revisited. In P. Auer and A. Di Luzio, editors, *The Contextualization of Language*, pages 39–54. **DOI:** https://doi.org/10.1075/pbns.22.04gum

Gurobi Optimization Inc. 2015. Gurobi optimizer reference manual.

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, Curran Associates, Inc., pages 1024–1034.

Rujun Han, Qiang Ning, and Nanyun Peng. 2019. Joint event and temporal relation extraction with shared representations and structured prediction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 434–444, Hong Kong, China, Association for Computational Linguistics. **DOI:** https://doi.org/10.18653/v1/D19-1041

Seyed Mehran Kazemi and David Poole. 2018. ReLNN: A deep neural model for relational learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 6367–6375, AAAI Press.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327. **DOI:** https://doi.org/10.1162/tacl_a_00101

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR*

2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings.

Parisa Kordjamshidi, Dan Roth, and Hao Wu. 2015. Saul: Towards declarative learning based programming. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 1844–1851.

Tatsuki Kuribayashi, Hiroki Ouchi, Naoya Inoue, Paul Reisert, Toshinori Miyoshi, Jun Suzuki, and Kentaro Inui. 2019. An empirical study of span representations in argumentation structure parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4691–4698, Florence, Italy. Association for Computational Linguistics. **DOI:** https://doi.org/10.18653/v1/P19-1464

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California, Association for Computational Linguistics. **DOI:** https://doi.org/10.18653/v1/N16-1030

I-Ta Lee and Dan Goldwasser. 2019. Multi-relational script learning for discourse relations. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4214–4226, Florence, Italy. Association for Computational Linguistics. **DOI:** https://doi.org/10.18653/v1/P19-1413

Chang Li, Aldo Porco, and Dan Goldwasser. 2018. Structured representation learning for online debate stance prediction. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3728–3739, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Shulin Liu, Yubo Chen, Shizhu He, Kang Liu, and Jun Zhao. 2016. Leveraging FrameNet to improve automatic event detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2134–2143. Berlin, Germany, Association for Computational Linguistics.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany, Association for Computational Linguistics.

Chaitanya Malaviya, Matthew R. Gormley, and Graham Neubig. 2018. Neural factor graph models for cross-lingual morphological tagging. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2653–2663. Melbourne, Australia. Association for Computational Linguistics. **DOI:** https://doi.org/10.18653/v1/P18-1247

Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. 2018. Deepproblog: Neural probabilistic logic programming. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 3749–3759, Curran Associates, Inc.

Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B. Tenenbaum, and Jiajun Wu. 2019. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision.

Giuseppe Marra, Francesco Giannini, Michelangelo Diligenti, and Marco Gori. 2019. Integrating learning and reasoning with deep logic models. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2019, Würzburg, Germany, September 16-20, 2019, Proceedings, Part II*, pages 517–532. **DOI:** https://doi.org/10.1007/978-3-030-46147-8_31

André F. T. Martins, Mário A. T. Figueiredo, Pedro M. Q. Aguiar, Noah A. Smith, and

Eric P. Xing. 2015. Ad3: Alternating directions dual decomposition for map inference in graphical models. *Journal of Machine Learning Research*, 16(16):495–545.

Andrew McCallum, Karl Schultz, and Sameer Singh. 2009. Factorie: Probabilistic programming via imperatively defined factor graphs. Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1249–1257, Curran Associates, Inc.

Miller McPherson, Lynn Smith-Lovin, and James M. Cook. 2001. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27(1):415–444. **DOI:** https://doi.org/10.1146/annurev.soc.27.1.415

Brian Milch, Bhaskara Marthi, Stuart Russell, David Sontag, Daniel L. Ong, and Andrey Kolobov. 2005. Blog: Probabilistic models with unknown objects. In *(IJCAI '05) Nineteenth International Joint Conference on Artificial Intelligence*.

Vlad Niculae, Joonsuk Park, and Claire Cardie. 2017. Argument mining with structured SVMs and RNNs. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 985–995, Vancouver, Canada. Association for Computational Linguistics. **DOI:** https://doi.org/10.18653/v1/P17-1091

Qiang Ning, Zhili Feng, Hao Wu, and Dan Roth. 2018. Joint reasoning for temporal and causal relations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2278–2288, Melbourne, Australia, Association for Computational Linguistics. **DOI:** https://doi.org/10.18653/v1/P18-1212

Shirui Pan, Jia Wu, Xingquan Zhu, Chengqi Zhang, and Yang Wang. 2016. Tri-party deep network representation. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 1895–1901.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics. **DOI:** https://doi.org/10.3115/v1/D14-1162

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. 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. ACM. **DOI:** https://doi.org/10.1145/2623330.2623732

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, Association for Computational Linguistics. **DOI:** https://doi.org/10.18653/v1/N18-1202

Peter Potash, Alexey Romanov, and Anna Rumshisky. 2017. Here's my point: Joint pointer architecture for argument mining. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1364–1373, Copenhagen, Denmark. Association for Computational Linguistics. **DOI:** https://doi.org/10.18653/v1/D17-1143

Matthew Richardson and Pedro M. Domingos. 2006. Markov logic networks. *Machine Learning*, 62(1–2):107–136. **DOI:** https://doi.org/10.1007/s10994-006-5833-1

Nick Rizzolo and Dan Roth. 2010. Learning based Java for rapid development of NLP systems. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*. Valletta, Malta, European Language Resources Association (ELRA).

Tim Rocktäschel and Sebastian Riedel. 2017. End-to-end differentiable proving. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3788–3800. Curran Associates, Inc.

Ali Sadeghian, Mohammadreza Armandpour, Patrick Ding, and Daisy Zhe Wang. 2019. Drum: End-to-end differentiable rule mining on knowledge graphs. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 15347–15357. Curran Associates, Inc.

Gustav Sourek, Vojtech Aschenbrenner, Filip Zelezný, Steven Schockaert, and Ondrej Kuzelka. 2018. Lifted relational neural networks: Efficient learning of latent relational structures. *Journal of Artificial Intelligence Research*, 62:69–100. **DOI:** https://doi.org/10.1613/jair.1.11203

Dhanya Sridhar, James Foulds, Bert Huang, Lise Getoor, and Marilyn Walker. 2015b. Joint models of disagreement and stance in online debate. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 116–125, Beijing, China, Association for Computational Linguistics. **DOI:** https://doi.org/10.3115/v1/P15-1012

Christian Stab and Iryna Gurevych, 2017. Parsing argumentation structures in persuasive essays. *Computational Linguistics*, 43(3):619–659. **DOI:** https://doi.org/10.1162/COLI_a_00295

Shivashankar Subramanian, Trevor Cohn, Timothy Baldwin, and Julian Brooke. 2017. Joint sentence-document model for manifesto text analysis. In *Proceedings of the Australasian Language Technology Association Workshop 2017*, pages 25–33, Brisbane, Australia.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015*, pages 1067–1077. **DOI:** https://doi.org/10.1145/2736277.2741093

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Eric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2071–2080, New York, New York, USA, PMLR.

Cunchao Tu, Han Liu, Zhiyuan Liu, and Maosong Sun. 2017. CANE: Context-aware network embedding for relation modeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1722–1731, Vancouver, Canada. Association for Computational Linguistics.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: On the importance of pre-training compact models. *arXiv preprint arXiv:1908.08962v2*.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Marilyn Walker, Pranav Anand, Rob Abbott, and Ricky Grant. 2012. Stance classification using dialogic properties of persuasion. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 592–596, Montréal, Canada, Association for Computational Linguistics.

Hai Wang and Hoifung Poon. 2018. Deep probabilistic logic: A unifying framework for indirect supervision. In *Proceedings of the 2018*

Conference on Empirical Methods in Natural Language Processing, pages 1891–1902, Brussels, Belgium. Association for Computational Linguistics. **DOI:** https://doi.org/10.18653/v1/D18-1215

William Yang Wang, Kathryn Mazaitis, and William W. Cohen. 2013. Programming with personalized pagerank: A locally groundable first-order probabilistic logic. In *22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013*, pages 2129–2138. ACM. **DOI:** https://doi.org/10.1145/2505515.2505573

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27-31, 2014, Québec City, Québec, Canada*, pages 1112–1119.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 323–333, Beijing, China. Association for Computational Linguistics. **DOI:** https://doi.org/10.3115/v1/P15-1032, **PMID:** 26003913, **PMCID:** PMC4695984

Han Xiao, Minlie Huang, Lian Meng, and Xiaoyan Zhu. 2017. SSP: semantic space projection for knowledge graph embedding with text descriptions. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 3104–3110.

Fan Yang, Zhilin Yang, and William W. Cohen. 2017. Differentiable learning of logical rules for knowledge base reasoning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 2319–2328. Curran Associates, Inc.

Xiao Zhang and Dan Goldwasser. 2019. Sentiment tagging with partial labels using modular architectures. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 579–590, Florence, Italy. Association for Computational Linguistics.

Hao Zhou, Yue Zhang, Shujian Huang, and Jiajun Chen. 2015. A neural probabilistic structured-prediction model for transition-based dependency parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1213–1222, Beijing, China, Association for Computational Linguistics. **DOI:** https://doi.org/10.3115/v1/P15-1117, **PMCID:** PMC4674637

A DRaiL Programs

// Predictions

r0: $\texttt{InThread}(\texttt{T}, \texttt{P}) \Rightarrow \texttt{IsPro}(\texttt{P}, \texttt{I})?$

r1: $\texttt{Respond}(\texttt{P}_2, \texttt{P}_1) \Rightarrow \texttt{Agree}(\texttt{P}_1, \texttt{P}_2)?$

// Dependencies

c0: $\texttt{Agree}(\texttt{P}_1, \texttt{P}_2)? \wedge \texttt{IsPro}(\texttt{P}_1, \texttt{I})? \Rightarrow \texttt{IsPro}(\texttt{P}_2, \texttt{I})?$

c1: $\texttt{Agree}(\texttt{P}_1, \texttt{P}_2)? \wedge \neg\texttt{IsPro}(\texttt{P}_1, \texttt{I})? \Rightarrow \neg\texttt{IsPro}(\texttt{P}_2, \texttt{I})?$

c2: $\neg\texttt{Agree}(\texttt{P}_1, \texttt{P}_2)? \wedge \texttt{IsPro}(\texttt{P}_1, \texttt{I})? \Rightarrow \neg\texttt{IsPro}(\texttt{P}_2, \texttt{I})?$

c3: $\neg\texttt{Agree}(\texttt{P}_1, \texttt{P}_2)? \wedge \neg\texttt{IsPro}(\texttt{P}_1, \texttt{I})? \Rightarrow \texttt{IsPro}(\texttt{P}_2, \texttt{I})?$

// Author constraints

c4: $\texttt{Author}(\texttt{P}_1, \texttt{A}) \wedge \texttt{Author}(\texttt{P}_2, \texttt{A}) \wedge \texttt{IsPro}(\texttt{P}_1, \texttt{I})? \Rightarrow \texttt{IsPro}(\texttt{P}_2, \texttt{I})?$

c4: $\texttt{Author}(\texttt{P}_1, \texttt{A}) \wedge \texttt{Author}(\texttt{P}_2, \texttt{A}) \wedge \neg\texttt{IsPro}(\texttt{P}_1, \texttt{I})? \Rightarrow \neg\texttt{IsPro}(\texttt{P}_2, \texttt{I})?$

(a) **Issue-Specific Stance Prediction.**

T: Thread, P: Post, I: Issue, A: Author

// Predictions

r0: $\texttt{InPar}(\texttt{C}, \texttt{P}) \Rightarrow \texttt{NodeType}(\texttt{C}, \texttt{T})?$

r1: $\texttt{InPar}(\texttt{C}_1, \texttt{P}) \wedge \texttt{InPar}(\texttt{C}_2, \texttt{P}) \Rightarrow \texttt{Link}(\texttt{C}_1, \texttt{C}_2)?$

r2: $\texttt{Link}(\texttt{C}_1, \texttt{C}_2)? \Rightarrow \texttt{AttackStance}(\texttt{C}_1, \texttt{C}_2)?$

r3: $\texttt{InPar}(\texttt{C}_1, \texttt{P}) \wedge \texttt{InPar}(\texttt{C}_2, \texttt{P}) \wedge \texttt{InPar}(\texttt{C}_3, \texttt{P}) \Rightarrow \texttt{Grandp}(\texttt{C}_1, \texttt{C}_2, \texttt{C}_3)?$

r4: $\texttt{InPar}(\texttt{C}_1, \texttt{P}) \wedge \texttt{InPar}(\texttt{C}_2, \texttt{P}) \wedge \texttt{InPar}(\texttt{C}_3, \texttt{P}) \Rightarrow \texttt{Coparent}(\texttt{C}_1, \texttt{C}_2, \texttt{C}_3)?$

// Higher order dependencies

c0: $\texttt{Grandp}(\texttt{C}_1, \texttt{C}_2, \texttt{C}_3)? \wedge \texttt{Link}(\texttt{C}_1, \texttt{C}_2)? \Rightarrow \texttt{Link}(\texttt{C}_2, \texttt{C}_3)?$

c1: $\texttt{Grandp}(\texttt{C}_1, \texttt{C}_2, \texttt{C}_3)? \wedge \texttt{Link}(\texttt{C}_2, \texttt{C}_3)? \Rightarrow \texttt{Link}(\texttt{C}_1, \texttt{C}_2)?$

c2: $\texttt{Coparent}(\texttt{C}_1, \texttt{C}_2, \texttt{C}_3)? \wedge \texttt{Link}(\texttt{C}_1, \texttt{C}_3)? \Rightarrow \texttt{Link}(\texttt{C}_2, \texttt{C}_3)?$

c3: $\texttt{Coparent}(\texttt{C}_1, \texttt{C}_2, \texttt{C}_3)? \wedge \texttt{Link}(\texttt{C}_2, \texttt{C}_3)? \Rightarrow \texttt{Link}(\texttt{C}_1, \texttt{C}_3)?$

// Source is always a premise

c4: $\texttt{Link}(\texttt{C}_1, \texttt{C}_2)? \Rightarrow \texttt{NodeType}(\texttt{C}_1, \texttt{"Premise"})?$

// Multiclass constraint

c5: $\texttt{HasType}(\texttt{C}, +\texttt{T})? = 1$

// Enforce tree structure

c6: $\texttt{Link}(\texttt{C}_1, +\texttt{C}_2)? \leq 1$

c7: $\texttt{Link}(\texttt{C}_1, \texttt{C}_2)? \Rightarrow \texttt{Path}(\texttt{C}_1, \texttt{C}_2)?$

c8: $\texttt{Path}(\texttt{C}_1, \texttt{C}_2)? \wedge \texttt{Path}(\texttt{C}_2, \texttt{C}_3)? \Rightarrow \texttt{Path}(\texttt{C}_1, \texttt{C}_3)?$

c9: $\texttt{InPar}(\texttt{C}_1, \texttt{P}) \wedge \texttt{InPar}(\texttt{C}_2, \texttt{P}) \wedge (\texttt{C}_1 = \texttt{C}_2) \Rightarrow \neg\texttt{Path}(\texttt{C}_1, \texttt{C}_2)?$

(b) **Argument Mining.** P: Paragraph, C: Component (Node), T: Type

Figure 6: DRaiL programs.

B Hyperparameters

For BERT, we use the default parameters.

Task	Param	Search Space	Selected Value
Open Domain (Local)	Learning Rate	2e-6,5e-6,2e-5,5e-5	2e-5
	Batch size	32 (Max. Mem)	32
	Patience	1,3,5	3
	Optimizer	SGD,Adam,AdamW	AdamW
	Hidden Units	128,512	512
	Non-linearity	–	ReLU
Open Domain (Global)	Learning Rate	2e-6,5e-6,2e-5,5e-5	2e-6
	Batch size	–	Full instance
Stance Pred. (Local)	Learning Rate	2e-6,5e-6,2e-5,5e-5	5e-5
	Patience	1,3,5	3
	Batch size	16 (Max. Mem)	16
	Optimizer	SGD,Adam,AdamW	AdamW
Stance Pred. (Global)	Learning Rate	2e-6,5e-6,2e-5,5e-5	2e-6
	Batch size	–	Full instance
Arg. Mining (Local)	Learning Rate	1e-4,5e-4,5e-3,1e-3,5e-2,1e-2	5e-2
	Patience	5,10,20	20
	Batch size	16,32,64,128	64
	Dropout	0.01,0.05,0.1	0.05
	Optimizer	SGD,Adam,AdamW	SGD
	Hidden Units	128,512	128
	Non-linearity	–	ReLU
Arg. Mining (Global)	Learning Rate	1e-4,5e-4,5e-3,1e-3,5e-2,1e-2	1e-4
	Patience	5,10,20	10
	Batch size	–	Full instance

Table 9: Hyper-parameter tuning.

C Code Snippets

We include code snippets to show how to load data into DRaiL (Figure 7-a), as well as to how to define a neural architecture (Figure 7-b). Neural architectures and feature functions can be programmed by creating Python classes, and the module and classes can be directly specified in the DRaiL program (lines 13, 14, 24, and 29 in Figure 7-a).

```
// Define entities
entity: "Post", arguments: ["postId"::ArgumentType.UniqueID];
entity: "Thread", arguments: ["threadId"::ArgumentType.UniqueID];
entity: "Author", arguments: ["authorId"::ArgumentType.UniqueString];
entity: "Issue", arguments: ["issueId"::ArgumentType.UniqueString];

// Define predicates
predicate: "InThread", arguments: [Thread, Post];
predicate: "Respond", arguments: [Post, Post];
predicate: "IsPro", arguments: [Post, Issue];
predicate: "Agree", arguments: [Post, Post];

// Load data from column-based text files
load: "InThread", file: "in_thread.txt";
load: "RespondTo", file: "respond_to.txt";
...

// Define feature module and class
fe_module: "4forums_ft";
fe_class: "FourForums_ft";

ruleset {
    rule: InThread(T, P) => IsPro(T,"gun_control")^?,
    lambda: 1.0,
    net_module: "nn_bert", net_class: "BertClassifier", net_config: "config_bert.json",
    fe_functions: [input("bert_post")];
    ...

    hardconstr: InThread(T, P) & InThread(T, Q) & Respond(P, Q) & Agree(P, Q)^?
            & IsPro(P, "gun_control")^? => IsPro(Q, "gun_control")^?;
    ...

} groupby: InThread.1;
```

(a) DRaiL Program

```
import torch
from transformers import AutoConfig, AutoModel
from drail.neuro.nn_model import NeuralNetworks

class BertClassifier(NeuralNetworks):

    def __init__(self, config, nn_id, output_dim):
        super(BertClassifier, self).__init__(config, nn_id, output_dim)

    def build_architecture(self, rule_template, fe, shared_params={}):
        self.bert_model_name = self.config["bert_model_type"]
        bert_config = AutoConfig.from_pretrained(self.bert_model_name)
        self.bert_model = AutoModel.from_pretrained(self.bert_model_name)
        self.dropout = torch.nn.Dropout(bert_config.hidden_dropout_prob)
        self.hidden2label = \
                torch.nn.Linear(bert_config.hidden_size, self.output_dim)

    def forward(self, x):
        input_ids, input_mask, segment_ids = x['input']
        outputs = self.bert_model(input_ids, attention_mask=input_mask,
                                  token_type_ids=segment_ids)
        pooled_output = outputs[1]
        pooled_output = self.dropout(pooled_output)
        logits = self.hidden2label(pooled_output)
        probas = torch.nn.functional.softmax(logits, dim=1)
        return logits, probas
```

(b) Neural Network Specification

Figure 7: Code snippets.