Integrating Deep Learning with Logic Fusion for Information Extraction

Wenya Wang, Sinno Jialin Pan
Nanyang Technological University, Singapore
{wangwy, sinnopan}@ntu.edu.sg

Abstract

Information extraction (IE) aims to produce structured information from an input text, e.g., Named Entity Recognition and Relation Extraction (OTE). Various attempts have been proposed for IE via feature engineering or deep learning. However, most of them fail to associate the complex relationships inherent in the task itself, which has proven to be especially crucial. For example, the relation between 2 entities is highly dependent on their entity types. These dependencies can be regarded as complex constraints that can be efficiently expressed as logical rules. To combine such logic reasoning capabilities with learning capabilities of deep neural networks, we propose to integrate logical knowledge in the form of first-order logic into a deep learning system, which can be trained jointly in an end-to-end manner. The integrated framework is able to enhance neural outputs with knowledge regularization via logic rules, and at the same time update the weights of logic rules to comply with the characteristics of the training data. We demonstrate the effectiveness and generalization of the proposed model on multiple IE tasks.

Introduction

Information extraction (IE) involves the identification of important information from a piece of input text and is a fundamental step towards knowledge inference. Various problems can be categorized as IE tasks, e.g., Named Entity Recognition (NER), Entity Linking, Opinion Target Extraction (OTE), Relation Extraction (RE), etc. In this work, we target at 2 challenging IE tasks including OTE and end-to-end RE. Given an input text, end-to-end RE aims to extract target entities as well as entity relations (Li and Ji 2014). For example, given the sentence “Rome is in Lazio province and Naples in Campania”, the task requires the identification of Rome, Lazio, Naples and Campania as location entities, and the relation between Rome and Lazio as LocatedIn, same for the relation between Naples and Campania. The task of OTE aims to identify opinion targets within an opinionated text (Hu and Liu 2004), e.g., service staff in “The service staff in this restaurant is very kind”.

Deep neural networks (DNNs) have been widely used for various IE tasks. Existing works adopted convolutional neural networks (Xu et al. 2018a; Adel and Schütze 2017) and recurrent/recursive neural networks (Wang et al. 2016; Miwa and Sasaki 2014) to learn context-aware and high-level features to facilitate predictions. Pointer networks have also been proposed for relation extraction (Katiyar and Cardie 2017). Despite their advantage over low-level feature engineering, the complex networks make learning harder when the amount of training data is insufficient, which is the case for many IE tasks. Moreover, the automation in DNNs makes it challenging to inject prior knowledge to guide the training process. On the opposite, symbolic logic systems provide an effective way to express complex domain knowledge in terms of logic rules and have proven to be advantageous when data is scarce. Inspired by the cognitive process that learns from both experiences and background knowledge, recent years have witnessed a growing interest in combining deep learning with logic reasoning (Manhaeve et al. 2018; Dong et al. 2019) mostly for solving logical problems.

To enhance the extraction performance in the NLP domain, we propose to incorporate domain knowledge as logic rules that are integrated into the representation learning system through a unified framework. The proposed model consists of a deep learning module as well as a logic module, where the deep learning module contains a transformer-style neural network to learn a rich feature representation for each word. The transformer model computes complex word-level correlations in multiple dimensions regardless of context distance (Vaswani et al. 2017), and has shown promising results in several NLP tasks, e.g., semantic role labeling (Tan et al. 2018). We believe this mechanism could be more beneficial to propagate information between related entities, compared to other deep models. The multi-head attention weight indicate the interactions between each pair of words which can be further fed into a relation classifier. The logic module is composed of a set of logic rules represented by First-order Logic (FOL). These rules explicitly specify the complex relationships in the output label space, which could not be handled using simple constraints. For example, a FOL rule, Live_In(Z, X) ∧ person(Z) ⇒ location(X), specifies that if the relation between two entities is Live_In and the first entity is of type person, then the second entity should have type location.
To associate distributed features with logic reasoning, we integrate the deep learning module and the logic module through 2 operations: 1) We design some mapping functions such that the information from the neurons could be passed to the logic system. Specifically, the neural outputs are treated as the inputs to the logic module, which combined with probabilistic logic operators, produces the logic outputs. Hence the outputs from the logic module reflects both neural learning and logic interactions among correlated atoms. Furthermore, a learnable weight is assigned to each logic rule to indicate its confidence level. The learnable weight for each rule makes the logic system more flexible and adaptable to specific training dataset, where a higher weight makes the corresponding rule more important within the corpus. 2) A discrepancy loss is proposed to measure the disagreement between the deep learning module and the logic system, which is minimized to allow for regularization of DNNs via logical knowledge. The discrepancy loss prompts the update of neural parameters towards rule-constrained directions, and at the same time adjusts the rule weights to be compatible with specific corpus.

To summarize, the proposed framework has the following contributions: 1) We use transformer mechanism for IE tasks to fully exploit interactions among the input space, which is also indicative for relation predictions. 2) We use logic rules to enforce complicated correlations in the output space and integrate these rules into the distributed representation learning system with a joint learning mechanism to achieve joint inference. To the best of our knowledge, this is the first work for information extraction that unifies DNN with logical knowledge in a rather smooth way to benefit learning of each other. 3) We introduce a general framework for knowledge fusion through discrepancy minimization, which can be adopted in various DNN models. We also demonstrate its effectiveness on different IE tasks.

Related Work

Information Extraction Various approaches have been proposed for entity and relation extraction, either through a pipeline procedure, or a joint inference framework. The pipeline strategy first learns an entity extraction model and then independently predicts relations based on the extracted entities (Chan and Roth 2011; Lin et al. 2016). This strategy suffers from error propagation. To solve this problem, joint inference is proposed to learn shared information between the neurons could be passed to the logic system. Specifically, the neural outputs are treated as the inputs to the logic module, which combined with probabilistic logic operators, produces the logic outputs. Hence the outputs from the logic module reflects both neural learning and logic interactions among correlated atoms. Furthermore, a learnable weight is assigned to each logic rule to indicate its confidence level. The learnable weight for each rule makes the logic system more flexible and adaptable to specific training dataset, where a higher weight makes the corresponding rule more important within the corpus. 2) A discrepancy loss is proposed to measure the disagreement between the deep learning module and the logic system, which is minimized to allow for regularization of DNNs via logical knowledge. The discrepancy loss prompts the update of neural parameters towards rule-constrained directions, and at the same time adjusts the rule weights to be compatible with specific corpus.

To summarize, the proposed framework has the following contributions: 1) We use transformer mechanism for IE tasks to fully exploit interactions among the input space, which is also indicative for relation predictions. 2) We use logic rules to enforce complicated correlations in the output space and integrate these rules into the distributed representation learning system with a joint learning mechanism to achieve joint inference. To the best of our knowledge, this is the first work for information extraction that unifies DNN with logical knowledge in a rather smooth way to benefit learning of each other. 3) We introduce a general framework for knowledge fusion through discrepancy minimization, which can be adopted in various DNN models. We also demonstrate its effectiveness on different IE tasks.

Deep Learning with Logic Rules The combination of neural learning systems with symbolic rules has long since been proposed, known as neural-symbolic systems (Garcez, Broda, and Gabbay 2012; Manhaeve et al. 2018; Dong et al. 2019; Sourek et al. 2018) that construct a network or connect the distributed systems with given rules for reasoning and inference in logic domains. Xu et al. treated logic knowledge as semantic regularizaztion in the loss function. The injection of logic rules in NLP tasks was recently proposed in (Rocktäschel, Singh, and Riedel 2015; Guo et al. 2016) for relation and knowledge graph learning that embed logic into the same space as distributed features in a single system. Hu et al. (2016) fused logical knowledge into deep models through posterior regularization. Logic rules were also used as evidences to construct adversarial sets (Minervini et al. 2017; Minervini and Riedel 2018), or as a form of indirect supervision (Wang and Poon 2018) to improve model training. Li and Srikkumar (2019) augmented deep learning models with logic neurons. In this work, we propose to combine DNN with logic in a smooth way, which adopts probabilistic logic instead of 0/1 hard assignments to facilitate backpropagation through the whole framework. Instead of inserting logic within the DNN architecture, we use a discrepancy loss to progressively bridge the gap between deep learning and logic rules.

Problem Definition & Motivation

We use end-to-end RE, which aims to jointly extract entities and their relations, as a motivating task to describe our proposed method. Denote by $E$ and $R$ the set of possible entity types and relation categories, respectively. Given an input sentence $\{w_1, w_2, ..., w_m\}$, entity extraction involves both entity segmentation as well as entity typing. We use BIO encoding scheme combined with entity types to form the sequence of output labels $y = \{y_1, y_2, ..., y_m\}$, where $y_i \in \{B-E_1, I-E_1, O\}_{E \in E}$. For example, B-PER (I-PER) indicates the beginning (inside) position of an entity of type person. Relation extraction aims to output a set of triplets $(e_1, e_2, r)$, where $e_1$ and $e_2$ represents the first and second entity, respectively, and $r \in R$ indicates the relation type between them. In this work, we treat entity extraction as a sequence labeling problem and relation extraction as a classification problem based on the identified entities.

A key motivation behind our proposed method is the complex correlation inherent both in the input and output spaces. In the input space, there exist intensive interactions among entities within a sentence to facilitate information propagation. For example, if Rome is extracted as a location entity and has close relationship with Lazio, it may help to identify Lazio as another entity. To exploit these interactions, we use the transformer mechanism with multi-head self-attentions to generate a correlation factor for each pair of words, which is injected for both entity and relation predictions. However, DNNs can only implicitly capture some correlations with-

\footnote{Note that the task of OTE can be regarded as a special case of RE, where there is only one entity type.}
out actually enforcing specific relationships. For example, if we know that one of the entities is of type person and the relations between the entities is Live_In, the other entity should be of type location. Even within entity predictions, there also exists dependencies enforcing possible positions within an entity, e.g., label “O” cannot be followed by “I”.

These relationships are especially crucial for the task, and yet few existing works have taken them into account. Although sequence labeling models, like Conditional Random Fields (CRFs) (Lafferty, McCallum, and Pereira 2001), are able to encode certain segmentation rules implicitly, they may not learn the optimal strategy when there is insufficient training data. Furthermore, they fail to model more complex relational rules. A few works treat the complex relationships among objects as constraints within the objective function. However, it is non-trivial to express those complex constraints, and the resultant optimization is challenging. Moreover, the constraints cannot be updated to align with the training corpus. In the literature, symbolic logic rules are well-known to be effective at modeling complex semantic relationships. For example, a dependency between entity types and relations can be expressed using FOL as

\[
\text{person}(X) \land \text{Live_In}(X, Z) \Rightarrow \text{location}(Z).
\]

When the training data is insufficient, as is the case for IE, logic rules provide crucial clues to assist learning.

Compared with the pipeline approach which is prone to error propagation, joint inference on related tasks, e.g., entity recognition and relation classification, has shown to be effective for (end-to-end) information extraction. Though joint learning has been proposed in deep architectures, most existing models fail to explicitly enforce the consistency among separate tasks. To address this problem, it is desirable to integrate logic rules specifying task relationships for joint inference. At this point, we propose to unify deep learning with logic rules in an end-to-end learning framework. To align discrete symbolic system with distributed representation learning, we propose to compute logic rules in a probabilistic way and define mapping functions to map the continuous output from a DNN to the logic units. Furthermore, a discrepancy loss is proposed that measure the discrepancy between the DNN outputs and the logic outputs to make these two modules consistent with each other. The discrepancy loss is able to regularize the DNN through domain knowledge, and at the same time update the logic module to comply with the training data.

### Methodology

The overall architecture of the proposed method is shown in Figure 1. It consists of 3 components, namely a deep neural network, a logic bank and a discrepancy unit. The DNN component takes a sequence of words as the input and finally produces a prediction vector for each word (and possibly candidate relations). The logic bank is fed with general domain knowledge that is easy to obtain and formalizes the knowledge as a set of first-order logic rules. Note that we assign a non-negative weight to each logic rule to indicate its confidence level which is updated according to the training corpus. The output from DNN is fed into the logic module to produce logic output. The discrepancy unit is responsible for aligning neural outputs with outputs from the logic bank. Specifically, we compute a distance between the output distributions from DNN and logic module with the aim to minimize the distance throughout the learning process.

### First-Order Logic

FOL is formed from constants, variables and predicates with propositional connectives \(\land, \lor, \neg\) and quantifiers. To avoid confusion, we use upper-case letters \((A)\) to represent variables and lower-case letters \((a)\) to represent constants. An atom is an n-ary predicate with n arguments \((R(A, B))\). A ground atom assigns constants to all of its arguments \((R(a, b))\). A clause can be written in the form of a rule: \(B_1 \land ... \land B_k \Rightarrow H\), where \(H\) is called the consequent of the rule and \(B_1 \land ... \land B_k\) is the precondition. The grounding of a clause is a substitution that maps each occurring variable in the clause to a constant: \(B_1(\phi) \land ... \land B_n(\phi) \Rightarrow H(\phi)\), where \(\phi\) denotes a substitution. A Herbrand interpretation is a mapping that assigns a truth value to each ground atom. To make it a Herbrand model, all the logic formulas should be satisfied. To find a Herbrand model, a feasible method is through the immediate consequence operator, which is a mapping \(T_p\) from Herbrand interpretation to itself:

\[
T_p(I) = \{H(\phi)| (B_1 \land ... \land B_n \Rightarrow H) \in \mathcal{P}, \\
\{B_1(\phi), ..., B_n(\phi)\} \in I\},
\]

(1)

where \(I\) is a Herbrand interpretation, \(\mathcal{P}\) is a set of clauses. Given known grounded atoms, we can find other grounded atoms as immediate consequences of the logic formulas. In our formulation, we use neural networks to simulate the immediate consequence operator and applies probabilistic logic where each formula is assigned a confidence score and each grounded atom has a continuous truth value within \([0, 1]\) to indicate its probability of being true.

### Deep Learning Module

The deep learning component is modeled as a transformer-style network consisting of multiple layers of self-attentions and Bi-GRU in order to model both sequential and distant...
We define an input vector $\hat{\mathbf{x}} = [\mathbf{x}_i^c : \mathbf{x}_j^p]$ for each word by concatenating its pre-trained word embedding $\mathbf{x}_i^c$ and its associated POS-tag embedding $\mathbf{x}_j^p$. To incorporate sequential context interactions, a Bi-GRU model with parameters $\Theta$ is firstly applied on top of $\mathbf{x}_i$ to generate hidden representations $\mathbf{h}_i$ by considering both forward and backward information. Mathematically, we denote this process by

$$\mathbf{h}_i = \left[ \hat{\mathbf{h}}_i : \tilde{\mathbf{h}}_i \right] = \left[ f(\mathbf{x}_i, \hat{\mathbf{h}}_{i-1}; \Theta) : f(\mathbf{x}_i, \tilde{\mathbf{h}}_{i+1}; \Theta) \right],$$

where $\hat{\mathbf{h}}_i$ and $\tilde{\mathbf{h}}_i$ indicate the forward and backward GRU output, respectively.

Subsequently, $\mathbf{h}_i$ is further fed into a transformer model consisting of multiple layers, where each layer stacks a Bi-GRU network on top of a multi-head self-attention module. Specifically, the self-attention network transforms its input $\mathbf{h}_i^t$ at $t$-th layer to $\mathbf{h}_i^{t+1}$ via a series of attention mechanism. For ease of illustration, we drop the superscript $t$ in the sequel. Mathematically, given a query vector corresponding to an input vector $\mathbf{h}_i$, each attention head computes one type of interactions between itself and other positions within the sequence and produces a transformed hidden representation:

$$\mathbf{h}_i^t = \sum_{j=1}^m \alpha_{ij}^c(W_q c \mathbf{h}_j), \text{ and}$$

$$\alpha_{ij}^c = \text{softmax}\left( \frac{(W_q c \mathbf{h}_i)(W_k c \mathbf{h}_j)}{\sqrt{d}} \right),$$

where $\mathbf{H}$ is a matrix consisting of $\mathbf{h}_i$ as column vectors, and $d$ is the dimension of $\mathbf{h}_i$. $\{W_q c, W_k c, W_v c\}$ are transformation matrices corresponding to the $c$-th attention head. We define $C$ different transformations for multi-head mechanism, where each transformation accounts for one representative interaction space. For IE, each head is regarded as computing a different relation between 2 words. A final hidden vector is produced via $\mathbf{h}_i = [W[h_i^1 : \ldots : h_i^C]]$.

Denote the final feature representation after applying Bi-GRU in the last transformer layer $T$ as $\mathbf{h}_i^T$. The neural outputs for entity prediction $y^E$ are generated through a fully-connected layer followed by a softmax layer:

$$s_{i,i-1}^E = \tanh(W_{i,i-1}^E [\mathbf{h}_i^T : \mathbf{x}_{i-1}], \mathbf{h}_{i-1}^T),$$

$$p(y_i^E | x_i) = \text{softmax}(W_y^E s_i^E + b_y^E),$$
consequence operator using (1) to produce the output $Y_C$ from the logic module. Specifically, the value of the consequent atom $H(φ)$ in each rule is deduced by applying $Γ$ on the rule body $B(φ)$, given a grounded clause $B(φ) ⇒ H(φ)$, where $B(φ)$ denotes the conjunction $B_{1(φ)} ∧ ... ∧ B_{n(φ)}$.

$$Y_C(H(φ)) = Γ(B_{1(φ)} ∧ ... ∧ B_{n(φ)}) = σ(\sum_{i=1}^{n} Γ(B_{i(φ)}) - n) + b_0 = σ(\sum_{i=1}^{n} y(B_{i(φ)}) - n) + b_0.$$ (8)

A detailed procedure to produce logic output is shown in Algorithm 1. Given neural outputs $\{y^E_{l,i\in 1\to m}\}$ in (5) and $\{y^R_{l,i\in 1\to m}\}$ in (7) for each sentence, and a set of rules $\{B^{(k)} ⇒ H^{(k)}\}_{k=1}^K$, the logic system produces a logic output $u_{l,k}^E$ for each word and $u_{l,k}^R$ for each relation corresponding to each rule $r_k$ as follows: For each rule $B^{(k)} ⇒ H^{(k)}$, we find its satisfyinggrundings $φ_0$ for $B^{(k)}$ from neural predictions and generate the logic output $Y_C(H^{(k)}_{φ_0})$ for its consequent atom $H^{(k)}_{φ_0}$ using (8). For example, given the rule $entiy_{l,i}(X) ∧ rel_{l,i}(X, Z) ⇒ entity_{l,i}(Z)$, if the neural model predicts $w_i$ as entity, and the relation between $w_i$ and $w_j$ as rel $l$, $(X = w_i, Z = w_j)$ is regarded as a satisfying grounding. Then the logic output for the consequent atom $entity_{l,i}(w_j)$ will be produced as $σ(σ(σ(σ(w^E_i[c] + y^R_{l,i}[l] - 2) + b_0)).$ We use $Φ_k$ and $Γ_k$ to collect the deduced consequent groundings and their logic values, respectively. The final logic output for each word $u_{l,k}^E$ (relation $u_{l,k}^R$) for each rule $r_k$ is obtained by aggregating the logic values across all the situations when acting as a consequent atom.

By making the DNN and the logic module compatible with each other, we can measure their discrepancy $ℓ_D(F, L)$ by comparing their distributions of their outputs:

$$ℓ_D(F, L) = \mathbb{E}_{z \sim \mathcal{X}} d(Y(x), Y_C(x)) = \frac{1}{K} \sum_{B^{(k)} ⇒ H^{(k)}} \frac{1}{|φ_k|} \sum_{φ_0 \in φ_k} β_k d(y(φ_0), Φ_k(φ_0)),$$ (9)

where $Y(x)$ and $Y_C(x)$ denote the neural output and logic output, respectively. $Φ_k$ collects the consequent atoms whose precondition is satisfied. We further assign a confidence weight $β_k ∈ [0, 1]$ for each rule to indicate its confidence and adjust its contribution to the discrepancy loss. The higher the weight, the more penalty to be given when neural outputs disagree with logic outputs. We use Mean-Squared-Error as the distance metric $d(\cdot, \cdot)$, because it provides a better gradient flow for a more stable training process.

**Training**

The integrated model can be trained end-to-end with gradient descent by minimizing $ℓ = ℓ_Y + ℓ_D$, where $ℓ_Y$ is the prediction loss for the deep learning model. Here we use cross-entropy loss for both entity and relation predictions:

$$ℓ_Y = -\frac{1}{N} \sum_{n=1}^{N} \left( log p(y^E_n|x_n) + log p(y^R_n|x_n, E_n) \right).$$ (10)

### Algorithm 1 Deep Logic

**Input:** Neural softmax outputs $\{y^E_{l,i\in 1\to m}\}$ (entities) and $\{y^R_{l,i\in 1\to m}\}$ (relations) for each sentence.

**Output:** $\{u_{l,k}^E\}_{i=1...m, k=1...K}$ (entities) and $\{u_{l,k}^R\}_{i=1...m', k=1...K}$ (relations)

**Initialize:** $u_{l,k}^E = 0, u_{l,k}^R = 0$ for $i \in \{1,...,m\}, l \in \{1,...,m'\}, k \in \{1,...,K\}$.

Collect feasible groundings of rule head and rule body.

for each rule $r_k : B^{(k)} ⇒ H^{(k)}$ do

1: Initialize $Φ_k = \emptyset$, $Γ_k = \emptyset$

2: Find a grounding $φ$ such that each $B^{(k)}_{φ_0}$ in $B^{(k)}_j$ is true according to neural predictions $\{y^E_{l,i\in 1\to m}, y^R_{l,i\in 1\to m'}\}$

3: Update $Φ_k ← Φ_k ∪ \{H^{(k)}_{φ_0}, Γ_k ← Γ_k ∪ \{Y_C(B^{(k)}_{φ_0})\}$

end for

Compute logic output for rule heads

for $k$ from 1 to $K$ do

1: Initialize $c_{i,k}^E = 0$ for $i \in \{1,...,m\}$, $c_{i,l}^R = 0$ for $l \in \{1,...,m'\}$

2: for $(φ, γ) ∈ (Φ_k, Γ_k)$ do

   Return the exact word $w_i$ or relation $r_l$ that corresponds to grounding $φ$

   Update $u_{l,k}^E ← u_{l,k}^E + γ$, $c_{i,k}^E ← c_{i,k}^E + 1$ or $u_{l,k}^R ← u_{l,k}^R + γ$, $c_{i,l}^R ← c_{i,l}^R + 1$

end for

end for

where $p(y^E_n|x_n) = \prod_{n=1}^{N} p(y^E_n = y^E_n|x_n)$ using (5). $|s_n|$ indicates the length of the $n$-th sentence. Similar procedure applies to $p(y^R_n|x_n, E_n)$ using (7). $E_n$ denotes the set of extracted entities for the $n$-th sequence. The backpropagation procedure is revealed in Figure 1 via (dashed) downward arrows. Specifically, the discrepancy loss updates both neural and logic outputs, together with rule weights according to (9) through gradient descent. To restrict the logic weights within $[0, 1]$, we apply a sigmoid function such that $β_k = σ(β_k^*).$ Then the gradient of the logic weight becomes

$$\frac{∂ℓ_D}{∂β_k^*} = \frac{1}{K|Φ_k|} \sum_{φ_0 \in φ_k} d(β_k^*)(1 - σ(β_k^*)),$$ (11)

where $d = d(y(φ_0), Φ_k(φ_0)).$ The gradients for logic output $u_{l,k}(φ)$ is further passed back to neural logits $y(B_{φ_0})$ according to (8), which combined with the classification loss, updates all the parameters within the neural model. Ideally, the discrepancy loss will punish the situation when the neural output highly differs from the logic output. In this case, the deep model will modify its network to be more aligned with the logic rules. On the other hand, the logic module will adapt its weights as well as the mappings that are passed back to the neurons. For example, if the deep module predicts $Rome$ as location, $Lazio$ as location and their relation as $OrgBased_In$ (which is wrong). When feeding them into the rule $loc(Rome) ∧ OrgBased_In(Lazio, Rome) ⇒ org(Lazio)$, the logic output for $entity_{org}(Lazio)$ will be high, different from the neural output. In this case, the discrepancy revises its rule body to decrease the incorrect neural output for relation $OrgBased_In$.
Table 1: Comparison with baselines on OTE.

Experiment
To demonstrate the effectiveness of our proposed method, we conduct experiments on 5 datasets from 2 tasks:

OTE: We use Restaurant and Laptop reviews from SemEval 2014 and 2016 (Pontiki et al. 2014; 2016).

End-to-End RE: 1) TREC: entity and relation dataset introduced in (Roth and Yih 2004). It has 4 entity types: others, person, location and organization, and 5 relations: Located_In, Live_In, OrgBased_In, Work_For and Kill. We follow the preprocessing from (Gupta, Schütze, and Andrassy 2016) 2) ACE05: annotated dataset with 7 coarse-grained entity types and 6 coarse-grained relation types between entities. We follow the same setting as (Li and Ji 2014).

For the OTE task, we follow the setting in (Wang et al. 2016) by first pre-training the word embedding using word2vec (Mikolov et al. 2013) on Yelp Challenge dataset and electronic dataset in Amazon reviews for restaurant domain and laptop domain, respectively. For RE task, the word embedding is pre-trained on wikipedia corpus using Glove (Pennington, Socher, and Manning 2014). For all experiments, the dimension for word embedding and POS embedding is set to 300 and 50, respectively. The hidden layers has dimension 200. We set label embedding with dimension 25. Following (Vaswani et al. 2017), we also use positional encoding that is added to the input vectors. The multi-head self-attention adopts 10 heads that leads to 10-dim attention weight vectors. For RE task, we use scheduled sampling, similar to (Miwa and Bansal 2016). To train the model, adam is adopted with initial rate as 1.0 and with dropout rate 0.1. For evaluation, we use micro-F1 scores on non-negative classes. An entity is counted as correct based on exact match. A relation is correct if both of its entities are correct and the relation type matches the ground-truth label.

Results & Analysis
Comparison on OTE task: Table 1 shows the comparison results for opinion target extraction with popular baselines. The last two rows indicate our proposed models, where TransF is the deep learning module without logic integration. Since the OTE task can be viewed as single-class entity extraction, the proposed model can be adapted to this task by ignoring relation predictions. From the results, we can observe that even without logic rules, the transformer model is able to achieve 2-out-of-3 best results compared to existing works. This proves the effectiveness of transformer for implicit interaction modeling.

Comparison on RE task: The comparison results on TREC is shown in Table 2. Existing works for this domain consists of 2 different settings and evaluations. The first setting is introduced in (Roth and Yih 2004) that assumes the entity boundaries are given and the task is to predict the entity types and relations. The second setting requires both segmentation and entity typing. Evaluations include relaxed version where an entity is regarded as correct if at least one of its consisting words have the correct type prediction. The strict version only treats a predicted entity as correct given a complete match. When boundaries are given, our model could be easily modified to treat each entity as a single unit for type predictions. To show the effect of joint inference with logic rules, we construct a pipeline model by first predicting entities followed by relation predictions given fixed entity parameters. Another model (Pipeline+feat) further append rule-based features as 1-hot vectors for relation prediction. Obviously, the pipeline model achieves inferior performance and simple features are far less expressive than logic rules. For both settings, our model achieves the best results with a large margin. Furthermore, we also test our model without the POS embedding, shown as “Ours (w/o) POS”, which still demonstrates some performance gain.

Table 3 shows the comparison results on ACE05 dataset. Note that Sun et al. (2018) used a non-decomposable global logic knowledge, we use segmentation rules that enforces possible segmentation labels for 2 adjacent words. Furthermore, we also incorporate implicit relational rules that state entity(X) ∧ entity(Z) ⇒ related(X, Z). This is achieved by keeping the relation prediction layer in the deep learning module. We find this strategy slightly improves our results which will be shown later. The results also show that the integration of logic rules is more effective than CRF by comparing with (Wang et al. 2016). Although Yu, Jiang, and Xia (2019) incorporated explicit constraints through integer linear programming, the separation from DNN during learning makes it suboptimal. This demonstrates the advantage of our unified framework that associates logic reasoning with representation learning. Clearly, our model achieves the state-of-the-art results on all 3 datasets.
Table 3: Comparison with baselines on relation extraction using ACE2005 dataset.

<table>
<thead>
<tr>
<th>Entity</th>
<th>Joint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res14</td>
<td>92.3</td>
</tr>
<tr>
<td>Res16</td>
<td>93.4</td>
</tr>
<tr>
<td>Lap14</td>
<td>93.0</td>
</tr>
<tr>
<td>TREC</td>
<td>84.2</td>
</tr>
<tr>
<td>ACE05</td>
<td>84.1</td>
</tr>
</tbody>
</table>

Table 4: Comparisons on different model settings.

<table>
<thead>
<tr>
<th>Setsups</th>
<th>Models</th>
<th>Res14</th>
<th>Res16</th>
<th>Lap14</th>
<th>TREC</th>
<th>ACE05</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity</td>
<td>TransF−x′</td>
<td>84.1</td>
<td>73.6</td>
<td>83.9</td>
<td>83.3</td>
<td>83.4</td>
</tr>
<tr>
<td></td>
<td>TransF−SR</td>
<td>85.0</td>
<td>73.9</td>
<td>82.1</td>
<td>84.9</td>
<td>83.2</td>
</tr>
<tr>
<td>Joint</td>
<td>TransF−α</td>
<td>84.3</td>
<td>85.6</td>
<td>83.3</td>
<td>84.3</td>
<td>57.4</td>
</tr>
<tr>
<td></td>
<td>TransF−SR+RR</td>
<td>85.4</td>
<td>74.7</td>
<td>82.5</td>
<td>87.1</td>
<td>83.4</td>
</tr>
</tbody>
</table>

Figure 3: Performance trend comparison on 2 models.

Conclusion

In this work, we propose a novel unified model to associate distributed learning with symbolic rules. The integrated framework is able to pass information from the neural model to the logic module and compute a discrepancy loss between these two components, which is minimized to update the whole network. The marriage between these two systems could regularize deep learning in the form of knowledge distillation. On the other hand, the logic system is also updated in terms of rule weights to adapt to specific data domain. Experimental results demonstrate the advantage of combining DNNs and logic for joint inference.

Acknowledgements

This work is supported by NTU Nanyang Assistant Professorship (NAP) grant M4081532.020, Singapore MOE AcRF Tier-2 Grant MOE2016-T2-2-060, and the Lee Kuan Yew Postdoctoral Fellowship program, Singapore.
References


Chan, Y. S., and Roth, D. 2011. Exploiting syntactico-semantic structures for relation extraction. In ACL-HLT.


Gupta, P.; Schütze, H.; and Andrassy, B. 2016. Table filling multitask recurrent neural network for joint entity and relation extraction. In COLING, 2537–2547.


Li, Q., and Ji, H. 2014. Incremental joint extraction of entity mentions and relations. In ACL, 402–412.


Lin, Y.; Shen, S.; Liu, Z.; Luan, H.; and Sun, M. 2016. Neural relation extraction with selective attention over instances. In ACL.


Miwa, M., and Bansal, M. 2016. End-to-end relation extraction using LSTMs on sequences and tree structures. In ACL.


Yin, Y.; Wei, F.; Dong, L.; Xu, K.; Zhang, M.; and Zhou, M. 2016. Unsupervised word and dependency path embeddings for aspect term extraction. In IJCAI.

