Integrating Deep Learning with Logic Fusion for Information Extraction

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

Problem definition

- End to end RE

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 $\mathcal{E}$ and $\mathcal{R}$ the set of possible entity types and relation categories, respectively.[1] 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 $y_i \in \{\text{B-E}_j, \text{I-E}_j, \text{O}\}_{E_j \in \mathcal{E}}$. 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 $r \in \mathcal{R}$ 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.

Motivation

- DNNs can only implicitly capture some correlations without actually enforcing specific relationships.
- 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.

Model

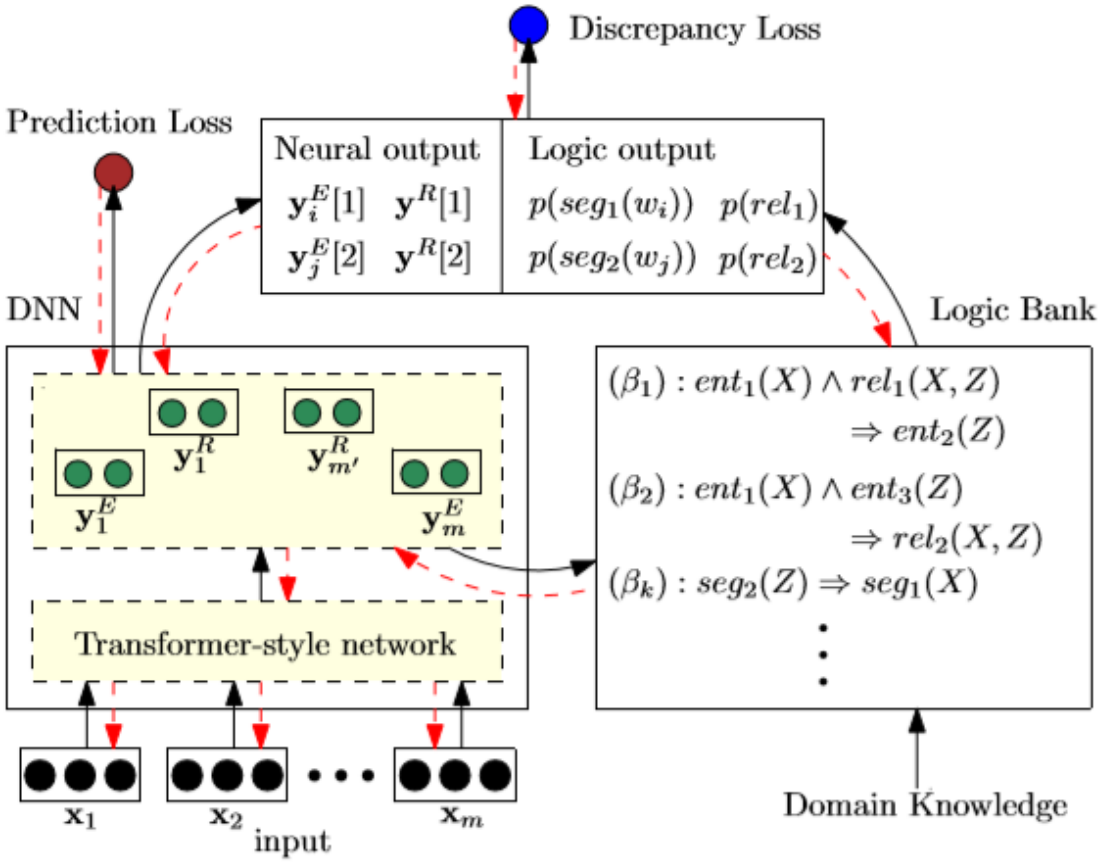

Figure 1: The overall architecture.

Model

- 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.
  - 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.

Deep learning module

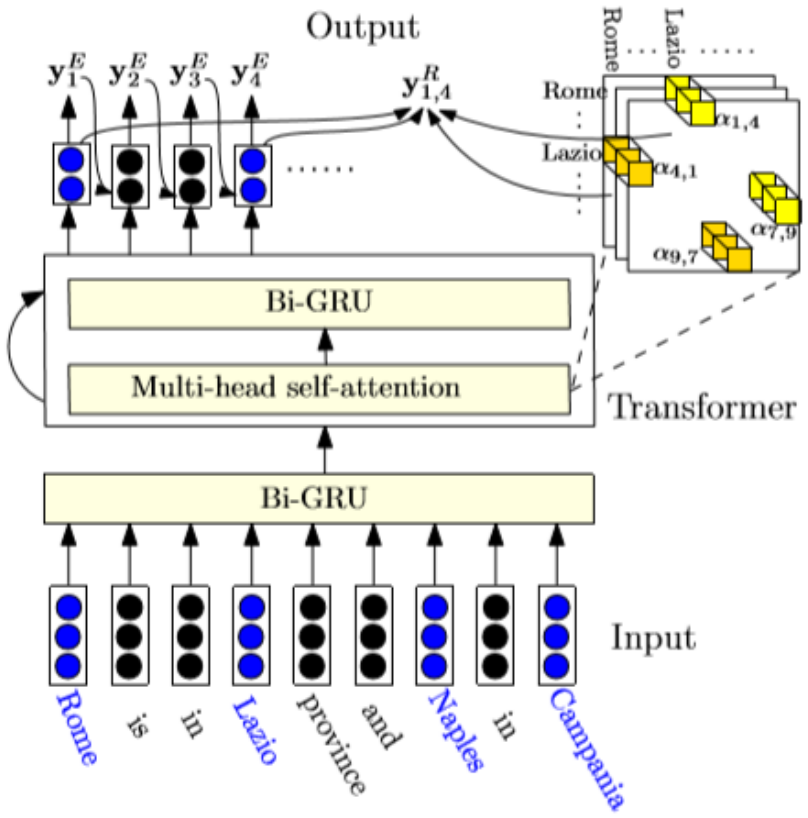

Figure 2: Deep learning module with transformer.

computing a different relation between 2 words. A final hidden vector is produced via $\tilde{\mathbf{h}}_i = \mathbf{W}[\tilde{\mathbf{h}}_i^1 : ... : \tilde{\mathbf{h}}_i^C]$.

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

$$\mathbf{s}_i^E = \tanh(\mathbf{W}_h^E[\mathbf{h}_i^T : \mathbf{x}_{i-1}^l] + \mathbf{b}_h^E), \quad (4)$$

$$p(\mathbf{y}_i^E|\mathbf{x}) = \text{softmax}(\mathbf{W}_y^E \mathbf{s}_i^E + \mathbf{b}_y^E), \quad (5)$$

where $\mathbf{x}_{i-1}^l$ denotes the entity label embedding of the previous token. The injection of label embeddings implicitly informs entity label dependencies. Similarly, the relation prediction $\mathbf{y}^R$ is produced for each pair of extracted entities via

$$\mathbf{s}_{i,j}^R = \tanh(\mathbf{W}_h^R[\mathbf{v}_i : \mathbf{v}_j : \boldsymbol{\alpha}_{i,j} : \boldsymbol{\alpha}_{j,i}] + \mathbf{b}_h^R), \quad (6)$$

$$p(\mathbf{y}_{i,j}^R|\mathbf{x}, \mathcal{E}) = \text{softmax}(\mathbf{W}_y^R \mathbf{s}_{i,j}^R + \mathbf{b}_y^R), \quad (7)$$

where $\mathbf{v}_i = [\mathbf{h}_i^T : \mathbf{x}_i^l]$, $\mathcal{E}$ indicates the set of extracted entities, and $\boldsymbol{\alpha}_{i,j}$ represents the multi-head attention weight vector between w_i and w_j. The attention vectors contain explicit correlation information that could assist relation prediction, as will be verified later in experiments.

Logic fusion

- 2 types
- Segmentation labels
  - segB(wi)⇒segO(wi−1) that enforces the previous word to have label O when the current word is the beginning of an entity.
- correlations between entity types and relations
  - Entity c(X) ∧ rel l(X,Z) ⇒ entity d(Z), which means relation l only appears when the first and second entity has type c and d, respectively.

- $\Gamma(P) = \mathbf{y}(P)$ with P representing a grounded atom and $\mathbf{y}$ as the neural output. For example, $\Gamma(entity_c(w_i)) = \mathbf{y}_i^E[c]$, where c indicates the index of the entity type. This can be interpreted as: the probability for w_i belonging to $entity_c$ equals to its corresponding neural output $\mathbf{y}_i^E[c]$.
- $\Gamma(P_1 \wedge ... \wedge P_n) = \sigma(a_0(\sum_{i=1}^n \Gamma(P_i) - n) + b_0)$.
- $\Gamma(P_1 \vee ... \vee P_n) = \sigma(a_1 \sum_{i=1}^n \Gamma(P_i) + b_1)$.
- $\Gamma(\neg P) = 1 - \Gamma(P)$.

Logic fusion

Here σ indicates the sigmoid activation function. The last 3 mappings are able to approximate the semantics of logic operators according to (Sourek et al. 2018). With these mapping functions, we compute a soft-version of the *immediate consequence operator* using (1) to produce the output $Y_{\mathcal{L}}$ from the logic module. Specifically, the value of the consequent atom $H_{(\phi)}$ in each rule is deduced by applying Γ on the rule body $B_{(\phi)}$, given a grounded clause $B_{(\phi)} \Rightarrow H_{(\phi)}$, where $B_{(\phi)}$ denotes the conjunction $B_{1(\phi)} \wedge ... \wedge B_{n(\phi)}$.

$$
\begin{aligned}
Y_{\mathcal{L}}(H_{(\phi)}) &= \Gamma(B_{1(\phi)} \wedge ... \wedge B_{n(\phi)}) \\
&= \sigma(a_0(\sum_{i=1}^{n} \Gamma(B_{i(\phi)}) - n) + b_0) \\
&= \sigma(a_0(\sum_{i=1}^{n} \mathbf{y}(B_{i(\phi)}) - n) + b_0). \qquad (8)
\end{aligned}
$$

Algorithm 1 Deep Logic

Input: Neural softmax outputs $\{\mathbf{y}_i^E\}_{i=1}^{m}$ (entities) and $\{\mathbf{y}_l^R\}_{l=1}^{m'}$ (relations) for each sentence.
Output: $\{u_{i,k}^E\}_{i=1,k=1}^{m,K}$ (entities) and $\{u_{l,k}^R\}_{l=1,k=1}^{m',K}$ (relations)
Initialize: $u_{i,k}^E = 0$, $u_{l,k}^R = 0$ 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 $r_k : B^{(k)} \Rightarrow H^{(k)}$ **do**
  1: Initialize $\Phi^k = \{\}, \Gamma^k = \{\}$
  2: Find a grounding ϕ such that each $B_{j(\phi)}^{(k)}$ in $B_{(\phi)}^{(k)}$ is true according to neural predictions $\{\mathbf{y}_i^E\}_{i=1}^{m}$, $\{\mathbf{y}_l^R\}_{l=1}^{m'}$
  3: Update $\Phi^k \leftarrow \Phi^k \cup \{H_{(\phi)}^{(k)}\}$, $\Gamma^k \leftarrow \Gamma^k \cup \{Y_{\mathcal{L}}(H_{(\phi)}^{(k)})\}$
end for
Compute logic output for rule heads
for k from 1 to K **do**
  Initialize $c_i^E = 0$ for $i \in \{1, ..., m\}$, $c_l^R = 0$ for $l \in \{1, ..., m'\}$
  for $(\phi, \gamma) \in (\Phi^k, \Gamma^k)$ **do**
    Return the exact word w_i or relation r_l that corresponds to grounding ϕ
    Update $u_{i,k}^E \leftarrow u_{i,k}^E + \gamma$, $c_i^E \leftarrow c_i^E + 1$ or $u_{l,k}^R \leftarrow u_{l,k}^R + \gamma$, $c_l^R \leftarrow c_l^R + 1$
  end for
  $u_{i,k}^E \leftarrow u_{i,k}^E / c_i^E$, $u_{l,k}^R \leftarrow u_{l,k}^R / c_l^R$
end for

Loss

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

$$\ell_Y = -\frac{1}{N}\sum_{n=1}^{N}(\log p(\hat{\mathbf{y}}_n^E|\mathbf{x}_n) + \log p(\hat{\mathbf{y}}_n^R|\mathbf{x}_n, \mathcal{E}_n)), \quad (10)$$

where $p(\hat{\mathbf{y}}_n^E|\mathbf{x}_n) = \prod_{i=1}^{|s_n|} p(\mathbf{y}_i^E = \hat{\mathbf{y}}_i^E|\mathbf{x}_n)$ using (5). $|s_n|$ indicates the length of the nth sentence. Similar procedure

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

$$\begin{aligned}\ell_D(\mathcal{F}, \mathcal{L}) &= \mathbb{E}_{x\sim\mathcal{X}}(d(Y_{\mathcal{F}}(x), Y_{\mathcal{L}}(x))) \\ &= \frac{1}{K}\sum_{\{B^{(k)}\Rightarrow H^{(k)}\}}\frac{1}{|\Phi^k|}\sum_{\phi\in\Phi^k}\beta_k d(\mathbf{y}(\phi), \mathbf{u}_k(\phi)), \quad (9)\end{aligned}$$

where $Y_{\mathcal{F}}(x)$ and $Y_{\mathcal{L}}(x)$ denote the neural output and logic output, respectively. Φ^k collects the consequent atoms whose precondition is satisfied. We further assign a confidence weight $\beta_k \in [0, 1]$ 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.

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; Pontiki et al. 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 data with 7 coarse-grained entity types and 6 coarse-grained relation types between entities. We follow the same setting as (Li and Ji 2014).

Experiment

	Restaurant14	Laptop14	Restaurant16
(Wang et al. 2016)	84.25	77.26	69.74
(Wang et al. 2017)	84.38	76.45	73.87
(Li and Lam 2017)	-	77.58	73.44
(Xu et al. 2018a)	84.24	81.59	74.37
(Yu, Jiang, and Xia 2019)	84.50	78.69	-
TransF	84.64	81.76	73.56
Ours	**85.62**	**82.46**	**74.67**

Table 1: Comparison with baselines on OTE.

Setting	Model	Evaluation	Entity	Relation
w/ Boundary	(Gupta et al. 2016)	relaxed	92.4	69.9
	(Bekoulis et al. 2018b)	relaxed	93.3	67.0
	(Bekoulis et al. 2018a)	relaxed	93.0	68.0
	(Miwa and Sasaki 2014)	strict	92.3	71.0
	(Adel and Schütze 2017)	strict	92.1	65.3
	Pipeline	strict	94.1	69.7
	Pipeline+feat	strict	94.5	70.2
	TransF	strict	94.6	72.8
	Ours (w/o) POS	strict	94.3	71.7
	Ours	strict	**95.1**	**74.1**
w/o Boundary	(Miwa and Sasaki 2014)	relaxed	80.7	61.0
	(Adel and Schütze 2017)	relaxed	82.1	62.5
	(Bekoulis et al. 2018b)	strict	83.0	61.0
	(Bekoulis et al. 2018a)	strict	83.6	62.0
	Pipeline	strict	84.2	57.2
	Pipeline+feat	strict	84.2	58.4
	TransF	strict	85.8	62.7
	Ours (w/o) POS	strict	85.0	63.1
	Ours	strict	**87.1**	**64.6**

Table 2: Results of End-to-End RE on TREC.

The first setting is that assumes the entity boundaries are given and the task is to predict the entity types and relations

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

Experiment

Settings	Models	Res14	Res16	Lap14	TREC		ACE05	
		OTE			Entity	Relation	Entity	Relation
Entity	TransF−$\mathbf{x}^t$	84.1	72.7	77.6	83.3	-	82.9	-
	TransF	84.6	73.6	81.8	84.2	-	83.2	-
	TransF+SR	85.0	73.9	82.1	84.9	-	83.2	-
Joint	TransF−$\boldsymbol{\alpha}$	-	-	-	84.3	60.6	83.3	57.4
	TransF	-	-	-	85.8	62.7	**83.4**	59.1
	TransF+SR+RR	**85.6**	**74.7**	**82.5**	**87.1**	**64.6**	**83.4**	**59.3**

Table 4: Comparisons on different model settings.

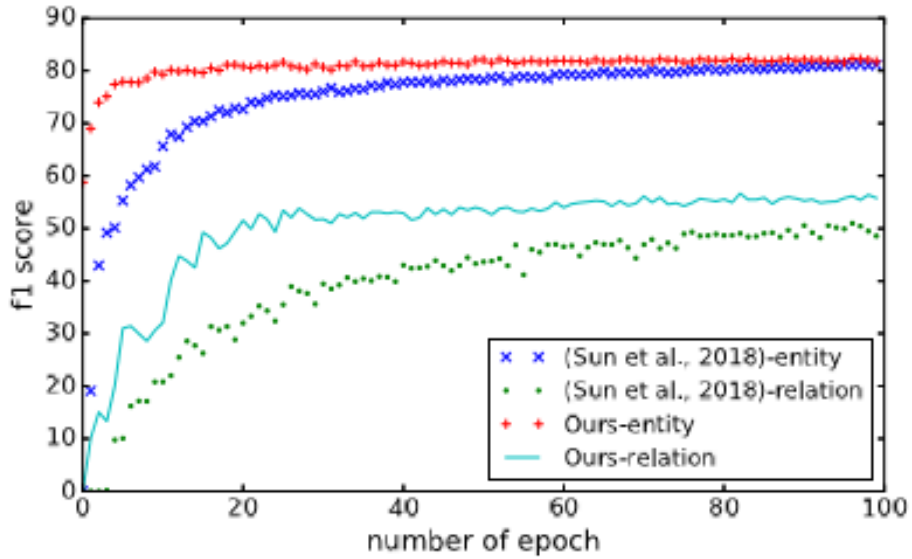

Figure 3: Performance trend comparison on 2 models.