Computational Linguistics Just Accepted MS.
https://doi.org/10.1162/coli_a_00415

Variational Deep Logic Network for Joint Inference of Entities and Relations

Wenya Wang*
Nanyang Technological University, Singapore

Sinno Jialin Pan**
Nanyang Technological University, Singapore

Nowadays, deep learning models have been widely adopted and achieved promising results on various application domains. Despite of their intriguing performance, most deep learning models function as black-boxes, lacking explicit reasoning capabilities and explanations, which are usually essential for complex problems. Take joint inference in information extraction as an example. This task requires the identification of multiple structured knowledge from texts, which is inter-correlated, including entities, events and the relationships between them. Various deep neural networks have been proposed to jointly perform entity extraction and relation prediction, which only propagate information implicitly via representation learning. However, they fail to encode the intensive correlations between entity types and relations to enforce their co-existence. On the other hand, some approaches adopt rules to explicitly constrain certain relational facts. However, the separation of rules with representation learning usually restrains the approaches with error propagation. Moreover, the pre-defined rules are inflexible and might bring negative effects when data is noisy. To address these limitations, we propose a variational deep logic network that incorporates both representation learning and relational reasoning via the variational EM algorithm. The model consists of a deep neural network to learn high-level features with implicit interactions via the self-attention mechanism and a relational logic network to explicitly exploit target interactions. These two components are trained interactively to bring the best of both worlds. We conduct extensive experiments ranging from fine-grained sentiment terms extraction, end-to-end relation prediction to end-to-end event extraction to demonstrate the effectiveness of our proposed method.

1. Introduction

Joint inference is commonly adopted in the field of information extraction (IE), e.g., end-to-end relation extraction, end-to-end event extraction. Compared to a pipelined procedure, joint inference performs multiple correlated subtasks in a single model simultaneously, which avoids error propagation and exploits inter-task correlations. For example, end-to-end relation extraction involves both the entity extraction and relation classification between entities. As shown in Figure 1(a), given a text input "*W. Dale*

* School of Computer Science and Engineering, Nanyang Technological University, 50 Nanyang Avenue, Singapore, 639798. E-mail: wangwy@ntu.edu.sg
** Corresponding author. School of Computer Science and Engineering, Nanyang Technological University, 50 Nanyang Avenue, Singapore, 639798. E-mail: sinnopan@ntu.edu.sg

Submission received: 26 October 2020; Revised version received: 9 July 2021; Accepted for publication: 19 July 2021

Computational Linguistics Just Accepted MS.
https://doi.org/10.1162/coli_a_00415

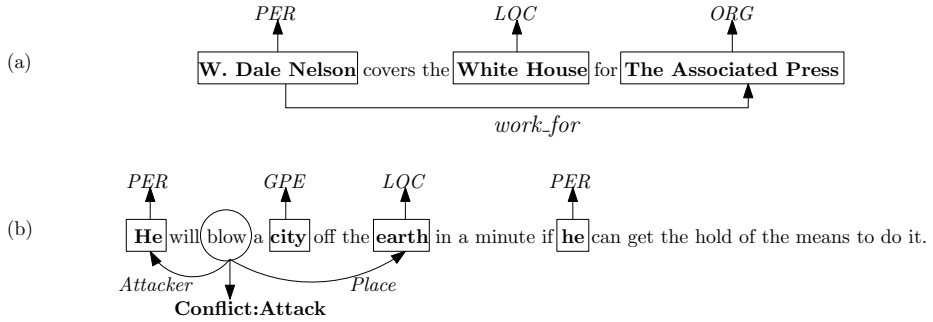

Figure 1
Examples of IE tasks: (a) End-to-end relation extraction. (b) End-to-end event extraction.

Nelson covers the White House for The Associated Press", end-to-end relation extraction requires the identification of *W. Dale Nelson* as an entity of type **person** (PER), *White House* as an entity of type **location** (LOC) and *The Associated Press* as an entity of type **organization** (ORG). At the same time, the relation between *W. Dale Nelson* and *The Associated Press* needs to be classified as **work_for**. For end-to-end event extraction, an event consists of an event trigger and an arbitrary number of arguments. The task involves the identification and classification of the following three items:

- **Entity mention**: An entity mention is a reference to an entity in the form of a noun phrase or a pronoun.
- **Event trigger**: An event trigger usually refers to the main word that clearly expresses an event occurrence. Event triggers can be verbs, nouns, and occasionally adjectives.
- **Event argument**: Event arguments refer to entities that fill specific roles in the event. They mainly include participants i.e., the entities that are involved in the event, and general event attributes such as place and time.

For example, in Figure 1(b), there are four entity mentions with their corresponding types labeled above. *blow* is a trigger for the event **Conflict:Attack** with two different arguments: *He* (**Attacker**) and *city* (**Place**).

Various deep learning models have been proposed to jointly extract entities, or events and their relations through either parameter/feature sharing (Miwa and Bansal 2016; Katiyar and Cardie 2017) to exploit task commonalities, or designing loss functions that consider task correlations, e.g., adopting a novel tagging scheme (Li and Ji 2014; Miwa and Sasaki 2014; Gupta, Schütze, and Andrassy 2016; Zhang, Zhang, and Fu 2017; Zheng et al. 2017). However, these joint deep models only exploit task interactions implicitly via parameter sharing or high-level feature learning without effective relational knowledge integration. We observe that intensive correlations or relational patterns exist among targets being extracted. Take Figure 1(a) as an example, if we know entity *W. Dale Nelson* is a person and it has relation 'work_for' with another entity *The Associated Press*, we can probably infer that *The Associated Press* is an organization. Note that the widely used BIO segmentation scheme in entity segmentation can be considered as a special case of correlation constraints among targets, e.g., "I" should not follow "O".

Computational Linguistics Just Accepted MS.
https://doi.org/10.1162/coli_a_00415

To fuse such explicit dependencies among different targets, some early works enforce the model predictions with constraints (Yang and Cardie 2013; Roth, Yih, and Yih 2007) or rely on global graphical models (Yu and Lam 2010) to produce structured predictions. These approaches, however, fail to connect final predictions with feature updates, resulting in error propagation. Logic rules have been integrated into deep learning architectures for natural language processing as a form of prior knowledge integration recently (Hu et al. 2016; Li and Srikumar 2019; Wang and Pan 2020). However, in existing methods, rules are explicitly given and kept fixed with learnable weights during model learning, which limits the expressiveness and adaptation of knowledge from training data.

To address the above limitations, we propose a novel marriage between deep feature learning and relational logic reasoning, which is named as **V**ariational **D**eep **L**ogic **N**etwork (VDLN), for joint inference in the IE domain. The complex relationships among target variables could be effectively captured both implicitly and explicitly via the mutual enhancements of deep neural networks and automatic logic inference in a joint learning framework. Specifically, VDLN consists of two modules: a deep learning module $\mathcal{Q}$ and a logic reasoning module $\mathcal{P}$. The deep learning module adopts the self-attention mechanism to explore the dependencies among each token in a sentence in order to generate word-level and relation-level features. It is also flexible to incorporate structured models, e.g., Conditional Random Fields (CRFs) (Lafferty, McCallum, and Pereira 2001) to produce structured outputs for entity segmentations. For the logic reasoning module, we construct a novel logic network which parameterizes logic inference process via a hierarchy of layers consisting of an atom layer and a rule layer. The final output of the logic network simulates rule entailments which reflects the probability of the target atom being true given the input atoms. The target atom could be regarded as a binary classifier for each target label. The logic network aims to learn relational correlations among the related variables, which is crucial for the task at hand. For example, the aforementioned dependency between entity and relation labels could be reflected via the first-order-logic (FOL) rule: $\text{person}(X_1) \wedge \text{work\_for}(X_1, X_2) \Rightarrow \text{organization}(X_2)$. It is worth noting that the logic reasoning module is flexible enough to achieve both rule learning given some simple rule templates and integration of pre-defined logic rules.

To smoothly integrate these two modules and to model dependencies of correlated variables for joint inference, we propose a variational EM learning paradigm. The E-step involves learning of module $\mathcal{Q}$ to produce probabilistic predictions for each variable. For the M-step, the logic reasoning module $\mathcal{P}$ conducts knowledge inference and updates its parameters according to the outputs of $\mathcal{Q}$. The alternation between E-step and M-step facilitates the integration and mutual enhancement of both knowledge reasoning and abstractive feature learning to achieve the best of both worlds.

To demonstrate our model's generality, we apply VDLN on a range of challenging IE tasks, focusing on different kinds of correslations and with increasing levels of difficulties. Specifically, we take *Aspect and Opinion Extraction* as the first IE task that focuses on entity extraction by treating aspect and opinion terms as two different entity types and exploring their interactions to boost the extraction accuracy. The second IE task is *End-to-End Relation Extraction* that considers correlations among entities and their relations. We use *End-to-End Event Extraction* as our third IE task that contains rich correlations between entities and events. The proposed model achieves better performances across all these tasks without the need to construct any prior knowledge. To summarize, our contributions include:

Computational Linguistics Just Accepted MS.
https://doi.org/10.1162/coli_a_00415

- We propose a novel logic-inspired network incorporating logic semantics for probabilistic reasoning, which is more expressive and beneficial for exploiting target interactions for joint inference. The logic network is able to learn effective reasoning patterns given the training corpus, and at the same time allows the integration of pre-defined logic rules.
- We design a variational EM algorithm within our deep logic networks for IE tasks, which bridge the gap between deep feature learning and knowledge reasoning to enhance the final performance.
- We conduct extensive experiments on 6 benchmark datasets across 3 IE tasks with increasing levels of difficulties to demonstrate the effectiveness and generality of our proposed model.

2. Related Work

2.1 Information Extraction

Information extraction aims to extract structured knowledge from texts, e.g., entities, relational triplets. In this paper, we mainly review three IE tasks that are related to our proposals. The first task is aspect and opinion extraction which focus on the identification of product aspects/attributes and their corrsponding opinion expressions. Existing works either relie on pre-defined rules and patterns among aspect terms and opinion terms utilising syntactic information of a sentence (Hu and Liu 2004; Qiu et al. 2011; Li et al. 2010), or design deep learning models considering different types of dependencies, e.g., contextual dependencies (Liu, Joty, and Meng 2015; Wang et al. 2017; Li and Lam 2017; Xu et al. 2018a), syntactic dependencies (Yin et al. 2016; Wang et al. 2016) and task dependencies (Chen and Qian 2020). Another recent work (Yu, Jiang, and Xia 2019) exploits the combination of explicit rules with deep feature learning via linear integer programming. However, such integration only treats rules as fixed constraints to revise deep learning predictions, without the ability to update rules and propagate information back to feature learning.

For end-to-end relation extraction. The early works adopt a pipeline procedure that first learns an entity extraction model and then trains a relation classifier based on the extracted entities (Chan and Roth 2011; Lin et al. 2016). This strategy is prone to error propagation resulting from the extracted entities. To resolve this limitation, subsequent works propose joint extraction models by sharing parameters (Miwa and Bansal 2016; Katiyar and Cardie 2017; Bekoulis et al. 2018; Bekoulis, Deleu, and Demeester 2018; Takanobu et al. 2019; Dixit and Al-Onaizan 2019; Dai et al. 2019a) or by designing loss functions to encode the task interactions, e.g., structured perceptron (Li and Ji 2014), novel labeling strategies (Miwa and Sasaki 2014; Gupta, Schütze, and Andrassy 2016; Zhang, Zhang, and Fu 2017; Zheng et al. 2017; Wang et al. 2018), global loss (Sun et al. 2018; Adel and Schütze 2017) and triplet/answer generation (Zeng et al. 2018; Li et al. 2019). Wang and Lu (2020) proposed to combine both sequence encoder and table encoder together with rich input embeddings for joint extraction. However, these approaches only exploits correlations among the subtasks implicitly. Another strategy is to enforce relational facts via explicit rule constraints (Roth, Yih, and Yih 2007; Yang and Cardie 2013; Kate and Mooney 2010) or graphical models (Yu and Lam 2010), which are separated from feature learning.

The third task which is more challenging is event extraction. Pipelined models are firstly proposed which require extensive feature engineering (Ji and Grishman 2008;

Computational Linguistics Just Accepted MS.
https://doi.org/10.1162/coli_a_00415

Liao and Grishman 2010; Patwardhan and Riloff 2009; Hong et al. 2011; McClosky, Surdeanu, and Manning 2011; Miwa et al. 2014). To capture interactions among different subtasks, graphical and structured prediction models have been proposed for joint inference of event triggers and event arguments (Poon and Vanderwende 2010; Venugopal et al. 2014; Riedel et al. 2009; Li et al. 2014; Judea and Strube 2016; Yang and Mitchell 2016). Recently, deep neural networks were also introduced for joint prediction in the domain of event extraction (Nguyen, Cho, and Grishman 2016; Sha et al. 2018; Liu, Luo, and Huang 2018; Nguyen and Nguyen 2019; Zhang, Ji, and Sil 2019; Wadden et al. 2019). However, most of the existing works depend on external linguistic resources to generate semantic and syntactic features in order to enhance the final prediction. Lin et al. (2020) adopted manually-designed global features to capture cross-task and cross-instance interactions.

2.2 Deep Learning with Logic Reasoning

Considering the limitation of pure deep learning models which lack the reasoning capabilities, and the inflexibility of pure symbolic models, a marriage between them has been proposed, namely *Neural-Symbolic Learning* which aims to equip distributed representation learning with some form of real intelligence, or on the other hand, assists symbolic models to handle uncertainties (Garcez, Broda, and Gabbay 2002; França, Zaverucha, and D'avila Garcez 2014; Serafini and d'Avila Garcez 2016; Evans and Grefenstette 2018; Manhaeve et al. 2018; Dong et al. 2019; Xu et al. 2018b; Tran and d'Avila Garcez 2018; Wang et al. 2019; Dai et al. 2019b; d'Avila Garcez et al. 2019; Ciravegna et al. 2020; Lamb et al. 2020; Yang and Song 2020). Deep neural networks have been used to simulate logic reasoning by parameterizing logic operators and logic atoms with neural weights (França, Zaverucha, and D'avila Garcez 2014; Tran and d'Avila Garcez 2018). Another group of researches focus on smooth integration of logic rules within the deep learning frameworks (Manhaeve et al. 2018; Xu et al. 2018b). A more challenging direction is to induce logic rules automatically through representation learning and differentiable back-propagation (Evans and Grefenstette 2018; Dong et al. 2019; Wang et al. 2019; Yang and Song 2020).

In the NLP domain, Rocktäschel, Singh, and Riedel (2015) and Guo et al. (2016) embedded logic rules into the distributed feature space for knowledge graph learning. Hu et al. (2016) fused discrete logic rules into DNNs through posterior regularization and Qu and Tang (2019) used variational EM algorithm to distill knowledge from a graph neural network into a Markov logic network. Another group of work used logic rules to construct adversarial sets (Minervini et al. 2017; Minervini and Riedel 2018), or as indirect supervision to improve model training (Wang and Poon 2018). Logic knowledge has also been inserted into deep architectures as named neurons (Li and Srikumar 2019). Recently, differentiable theorem proving has been proposed that parameterizes symbolic unification in the backward chaining process of prolog (Gallaire and Minker 1978) with neural weight learning (Rocktäschel and Riedel 2017; Campero et al. 2018; Minervini et al. 2020). Inspired by (Qu and Tang 2019), we also adopt the variational EM algorithm for knowledge distillation. But different from the above works, we design a semantically-meaningful deep architecture for automatic logic reasoning. The logic-inspired network is able to learn expressive and useful reasoning patterns that are adapted given the training corpus, and at the same time flexible to incorporate pre-defined logic rules. In the domain of information extraction, Wang and Pan (2020) used pre-defined logic rules as a form of regularizer to be imposed to the learning of DNNs. The regularizer is realized via a discrepancy loss between the deep learning predictions

Computational Linguistics Just Accepted MS.
https://doi.org/10.1162/coli_a_00415

Symbols	Description
$\mathcal{E}$, $\mathcal{R}$, $\mathcal{V}$	the set of all entity types, relation categories, event trigger categories
E	the set of segmentation labels $\mathrm{E} = \{\mathrm{B}_j, \mathrm{I}_j, \mathrm{O}\}_{j \in \mathcal{E}}$ with $j \in \mathcal{E}$ an entity type
$\mathcal{N}_\epsilon$, $\mathcal{N}_r$	the set of all words, the set of all relations within a sentence
D	a set of atoms $D = \{d_1, ..., d_N\}$
ϵ_i, $r_{(\epsilon_i, \epsilon_j)}$, v_i	a constant representing an entity, a relation, an event trigger
w_i, y_i, $\mathbf{y}_i$	an input word, an output label, an output prediction vector
θ, ϕ	all the parameters corresponding to module $\mathcal{Q}$, module $\mathcal{P}$
x_i, X_i	a logic constant, a logic variable
d_i	a logic atom consisting of a predicate and arguments $d_i = pred(X_1, ..., X_m)$
h	the head atom of a clause $d_1 \wedge ... \wedge d_n \Rightarrow h$
$v(\cdot)$	the probabilistic value of an atom or a clause $v(\cdot) \in [0, 1]$
$\mathbf{x}$, $\mathbf{h}$, $\mathbf{u}$, $\boldsymbol{\alpha}$	a vector representation of an input, a hidden neuron, an entity type, attention scores
$\mathbf{W}$, $\mathbf{b}$	a trainable transformation matrix, a trainable bias vector
$\mathbf{v}_{i,n}$, $\mathbf{V}_n$	a trainable transformation vector, bi-linear transformation matrix for atom evaluations
$\mathbf{q}$, $\mathbf{p}$	an output probabilistic vector from module $\mathcal{Q}$, module $\mathcal{P}$
R, γ	a logic rule identifier, the confidence score of a rule
$m \in \mathcal{N}_\epsilon \cup \mathcal{N}_r$	a logic constant referring to either a word or a relation
$\mathrm{ctx}(m)$	the set of logic constants that form the context of m
$\tilde{\mathbf{y}}_{\mathrm{ctx}(m)}$	a vector of probabilistic inputs for module $\mathcal{P}$: $\tilde{\mathbf{y}}_{\mathrm{ctx}(m)} = (\mathbf{q}_{m_1}, \mathbf{q}_{m_2}, ..., \mathbf{q}_{m_{\lvert\mathrm{ctx}(m)\rvert}})$
σ	the sigmoid function
β^t, $\mathbf{d}$	a weight vector that weighs each logic rule, a vector of atom values
Y, Z	the set of target random variables, the set of hidden random variables
$p(\cdot)$, $q(\cdot)$	probabilistic distributions

Table 1
A table with all of the symbols used in this work and their descriptions.

and the satisfiability of their corresponding logic rules. However, this mechanism only locally influences the learning of DNNs. Compared with (Wang and Pan 2020), our proposed model is able to learn different combinations of logic atoms to form the rules and it is also flexible to incorporate pre-defined knowledge. Moreover, our EM training algorithm alternates between an inference step and a learning step to achieve mutual enhancement which globally enforces the learning of both modules, instead of sample-wise regularization.

3. Problem Definition & Preliminary

For ease of illustration, we first list all the symbols used in this work together with their descriptions in Table 1.

3.1 Problem Definition

For all the three IE tasks, the target variables can be categorized as: 1) Entities, with the set of all entity types denoted by $\mathcal{E}$. 2) Events, with $\mathcal{V}$ denoting the set of all event types. 3) Relational triplets (s, r, o) governed by a set of relation categories $r \in \mathcal{R}$, with s and o being the subject and object of relation r, respectively. For convenience, we use $r_{(s,o)}$ to denote the relational triplet. Given an input sentence $\{w_1, w_2, ..., w_n\}$, entity extraction is formalized as a sequence labeling problem to generate entity segmentation. Denote the set of segmentation labels by $\mathrm{E} = \{\mathrm{B}_j, \mathrm{I}_j, \mathrm{O}\}_{j \in \mathcal{E}}$, with B_j, I_j, O indicating the beginning, inside and outside of an entity of type j, respectively. The output is a label sequence

Computational Linguistics Just Accepted MS.
https://doi.org/10.1162/coli_a_00415

$\{y_1, y_2, ..., y_n\}$, where $y_i \in$ E. End-to-end relation extraction aims to generate both entity segmentation as well as a set of relational triplets $r_{(\epsilon_1,\epsilon_2)}$, where ϵ_1 and ϵ_2 correspond to entities. End-to-end event extraction consists of 3 sub-tasks: entity extraction, event trigger extraction and event argument prediction. Event trigger extraction is formalized as a token-based classification problem with $|\mathcal{V}|$ classes. Event argument prediction aims to produce relational triplet $r_{(\epsilon,v)}$ where ϵ is an entity, v is an event trigger and r denotes the argument relation between ϵ and v. For relational triplet prediction, we pair all candidate entities (or entities with event triggers) that are extracted in the first place to predict the relation label.

3.2 Variational EM

Given a model p_ϕ parameterized by ϕ, the objective is to maximize $\mathcal{L} = \log p(Y;\phi)$ w.r.t ϕ, where Y is the target variable. We can re-formalize the objective by introducing another model q parameterized by θ:

$$\mathcal{L} = \log \int q(Z;\theta)\frac{p(Y,Z;\phi)}{q(Z;\theta)}dZ \geq \mathbb{E}_{q(Z;\theta)}[\log p(Y,Z;\phi) - \log q(Z;\theta)]. \tag{1}$$

Here $p(Y;\phi) = \int p(Y,Z;\phi)dZ$ with Z being the hidden variables that are highly correlated with Y. The expectation in (1) is the Evidence Lower Bound (ELBO) of the original objective. The equality of (1) holds when $q(Z;\theta) = p(Z|Y;\phi)$. Hence, the original problem can be optimized via the variational EM algorithm (Neal and Hinton 1999) that alternates between an E-step and an M-step. In the E-step, p is fixed and q is updated to approximate the equality

$$q(Z;\theta) = p(Z|Y;\phi). \tag{2}$$

In the M-step, q is fixed, and p is updated by maximizing:

$$\mathbb{E}_{q(Z;\theta)}[\log p(Y,Z;\phi)], \tag{3}$$

given that the last term $\mathbb{E}_{q(Z;\theta)}[\log q(Z;\theta)]$ of the ELBO is a constant w.r.t. p. Such formulation has 2 advantages: 1) It promotes mutual learning from 2 different perspectives when only optimizing the single model p is hard and insufficient. With such consideration, we treat the logic module $\mathcal{P}$ as p and the deep learning module $\mathcal{Q}$ as q in (1). 2) EM exploits the dependencies between input and hidden variables, which is beneficial for modeling inter-dependencies for joint inference, e.g., the correlations between entity types and relation categories. But different from (Qu and Tang 2019), we design a semantically-meaningful deep architecture for automatic logic reasoning.

Note that Qu and Tang (2019) adopted this formulation to distill information from graph neural networks for Markov logic networks with given logic rules. Compared to other existing works that either used manually constructed logic rules to enhance the learning of DNNs, or learn logic rules but are limited in terms of computational efficiency, we build on top of (Qu and Tang 2019) to achieve mutual learning of both DNNs and logic reasoning.

Computational Linguistics Just Accepted MS.
https://doi.org/10.1162/coli_a_00415

3.3 First-Order Logic

A first-order logic (FOL) program associates *constants*, *variables* and *predicates* with logic connectives, namely $\vee$, $\wedge$ and $\neg$, and quantifiers. A *constant* x is an object, e.g., a word or a relation between two words. A *variable* X refers to a group of constants. A *predicate* $pred$ can be regarded as a function that maps constants or variables to *True* or *False*. An FOL formula consists of atoms connected with $\vee$, $\wedge$ or $\neg$, representing logic "OR", logic"AND" and logic "NOT", respectively. Here an atom is an n-ary predicate taking constants or variables as arguments. For example, $person(X)$ is a 1-ary atom, $same(X_1, X_2)$ is a 2-ary atom. An atom is said to be grounded if all of its variables are instantiated with constants. Given these definitions, an FOL formula in the form of entailment could be represented as

$$d_1 \wedge d_2 \wedge ... \wedge d_n \Rightarrow h, \tag{4}$$

where $d_i = pred(X_1, ..., X_m)$ is a body atom and h is the head atom of the formula. $\Rightarrow$ in (4) can be replaced with $\vee$ and $\neg$, converting (4) to an equivalent form: $\neg d_1 \vee \neg d_2 \vee ... \vee \neg d_n \vee h$ consisting of valid connectives.

In our problem setting, we treat each different classifier as FOL entailments and define each target label as the head atom of a set of FOL formula. For example, $d_1 \wedge d_2 \wedge ... \wedge d_n \Rightarrow person(X)$ explains how $person(X)$ can be deduced from its body atoms. In this case, if $d_1 \wedge d_2 \wedge ... \wedge d_n$ evaluates to *True*, $person(X)$ will also be *True*.

In order to smoothly integrate first-order logic with deep learning, probabilistic logic has been proposed that translates the hard assignment of *True* and *False* to a soft version within $[0, 1]$ that indicates the probability of an atom being true. Hence, we can define $\upsilon(person(X)) = \upsilon(d_1 \wedge d_2 \wedge ... \wedge d_n) \in [0, 1]$, where $\upsilon(\cdot)$ denotes the probabilistic evaluation of the input. Furthermore, we adopt T-norm (Klement, Mesiar, and Pap 2013) for probabilistic evaluations of logic connectives:

$$\upsilon(d_1 \wedge ... \wedge d_n) = \min(\upsilon(d_1), ..., \upsilon(d_n)), \tag{5}$$

$$\upsilon(d_1 \vee ... \vee d_n) = \max(\upsilon(d_1), ..., \upsilon(d_n)), \tag{6}$$

$$\upsilon(\neg d_i) = 1 - \upsilon(d_i). \tag{7}$$

To encode uncertainties within probabilistic logic, we assign each FOL formula $d_1 \wedge d_2 \wedge ... \wedge d_n \Rightarrow h$ with a learnable confidence score $\gamma \in [0, 1]$. The higher the score, the more confidence the formula plays in the computation process.

4. Motivation

Conventional deep learning usually lacks knowledge integration and fails to explicitly model the crucial interactions among the targets. Recently, logic reasoning has been adopted and integrated with DNNs to enhance performance by introducing knowledge as FOL rules. Among them, probabilistic logic converts the hard $0/1$ assignment to soft probabilities (Nilsson 1986), which facilitates optimization through gradient descent. However, the pre-designed FOLs may not be expressive enough to represent the inherent patterns and prevents adaptation to a given training dataset. To address this limitation, we propose variational deep logic networks (VDLNs) which inherit the representational power of deep learning, and at the same time simulate the logic rule learning process via a novel logic network consisting of a hierarchy of an atom layer, a

Computational Linguistics Just Accepted MS.
https://doi.org/10.1162/coli_a_00415

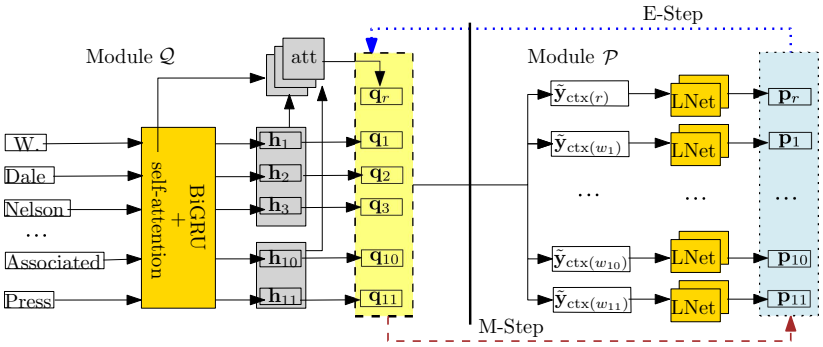

Figure 2
An overview of the proposed model. The left part corresponds to module $\mathcal{Q}$ and the right part corresponds to module $\mathcal{P}$. Module $\mathcal{Q}$ transforms the text input to a set of prediction vectors $\mathbf{q}_i$'s which can be fed as input to module $\mathcal{P}$ to produce a set of prediction vectors $\mathbf{p}_i$'s. Then an EM algorithm is used to train all the parameters that alternates between an E-step and a M-step.

rule layer and an output layer. Given some predefined rule templates, the atom layer implements a neural transformation process to convert the inputs to a set of abstract atoms. Then our logic network learns to discriminatively select the most relevant atoms in the atom layer to compose a logic rule in the rule layer. Our network design avoids a manual construction of atoms for each rule that is task-dependent. It is also flexible to inject any prior knowledge into the logic network if the rules are easy to obtain. The combination of automatically-learned and pre-defined logic rules is realized via a form of residual connection.

To integrate a logic system with deep learning, most existing works only use knowledge to regularize feature learning or feed deep learning outputs as the inputs to the logic system, but ignore the mutual interactions. In this work, we introduce a novel integration of DNNs and knowledge reasoning via variational EM. Note that Qu, Bengio, and Tang (2019) proposed to adopt variational EM for semi-supervised classification by associating 2 graph neural networks. Qu and Tang (2019) further extended the algorithm for efficient inference in Markov Logic (Richardson and Domingos 2006). However, their work only updates the weights for pre-defined rules without learning the predicates of rules. Different from previous works, our proposed model automatically learns useful predicates and the weights of different instantiations of those predicates which explore the associations among highly dependent classifiers for joint inference.

5. Methodology

The overview of the proposed model **VDLN** is shown in Figure 2. It consists of 2 modules: 1) Module $\mathcal{Q}$ consists of a deep neural network that transforms the input sequence of text into abstractive features $\mathbf{h}_i$'s and produces the probabilistic outputs $\mathbf{q}_i$'s. 2) A logic module $\mathcal{P}$ consists of a set of logic networks (LNets), with one LNet corresponding to each specific word w_i and relation r. Each LNet takes $\tilde{y}_{\text{ctx}(w_i)}$ ($\tilde{y}_{\text{ctx}(r)}$) as input which consists of information from all of its associated variables to conduct knowledge reasoning among these variables, and generates the final probabilistic evaluations $\{\mathbf{p}_i\}$'s. Note that in VDLN, besides modeling complex correlations between targets, the logic module $\mathcal{P}$ also implements the BIO labeling scheme. The entire model is trained via the variational EM algorithm that alternates between an E-step

Computational Linguistics Just Accepted MS.
https://doi.org/10.1162/coli_a_00415

(inference) and an M-step (learning). In the E-step, the deep module $\mathcal{Q}$ generates soft predictions of each word and candidate relation by distilling knowledge from $\mathcal{P}$. In the M-step, the logic module $\mathcal{P}$ takes the predictions of $\mathcal{Q}$ as input and generates a probabilistic output for each target class of each word and relation. With a more concrete example, the overall procedure is the following: Given an input sentence of 11 tokens "*W. Dale Nelson covers the White House for The Associated Press*", module $\mathbf{Q}$ first produces the hidden representations $\{\mathbf{h}_1, ..., \mathbf{h}_{11}\}$ and the output vectors $\{\mathbf{q}_1, ..., \mathbf{q}_{11}\}$, as shown in Figure 2. Likewise, a relation output vector $\mathbf{q}_r$ is generated for each pair of candidate entities predicted via $\mathbf{h}_i$, e.g., (*W. Dale Nelson*, *Associated Press*), based on their hidden representations $\{\mathbf{h}_i\}_{i\in\{1,2,3,10,11\}}$ and their attention scores. Then these vectors $\{\{\mathbf{q}_r\}'\mathrm{s}, \mathbf{q}_1, ..., \mathbf{q}_{11}\}$, where $\{\mathbf{q}_r\}'$s collects the set of all entity pairs for relation predictions, are used to form the input $\tilde{\mathbf{y}}_{\mathrm{ctx}(w_i)}$ (or $\tilde{\mathbf{y}}_{\mathrm{ctx}(r)}$) for each word (or relation) in module $\mathcal{P}$ to produce the final probabilistic output vectors $\{\{\mathbf{p}_r\}'\mathrm{s}, \mathbf{p}_1, ..., \mathbf{p}_{11}\}$ for all the words and relations from Module $\mathcal{P}$. With the output vectors from both modules, we conduct EM training algorithm that firstly update the parameters in $\mathcal{Q}$ by treating the predictions from $\mathbf{p}_i$'s as the supervision labels. Then in the next iteration, we update the parameters in $\mathcal{P}$ by treating the predictions from $\mathbf{q}_i$'s as the supervision labels.

In the following, we will describe the architecture of VDLN in Section 5.1 and Section 5.2 in detail.

5.1 Deep Learning with Self-Attention

As shown in Figure 2, $\mathcal{Q}$ is a deep neural network based on the self-attention mechanism and a bidirectional Gated Recurrent Unit (BiGRU) in order to model both non-local and contextual token-level interactions, respectively. Specifically, we use a transformer-style framework with multi-head self-attentions to generate a hidden representation for each word incorporating its correlation with other tokens. Given input embeddings $\{\mathbf{x}_i\}$'s corresponding to a text sequence $\{w_1, w_2, ..., w_n\}$, the multi-head self-attention model generates a hidden representation $\bar{\mathbf{h}}_i$ for each word through a linear transformation of all attention heads

$$\bar{\mathbf{h}}_i = \mathbf{W}[\bar{\mathbf{h}}_i^1 : ... : \bar{\mathbf{h}}_i^C], \tag{8}$$

where each attention head c produces

$$\bar{\mathbf{h}}_i^c = \sum_{j=1}^{m} \alpha_{ij}^c (\mathbf{W}_v^c \mathbf{x}_j), \text{ with } \boldsymbol{\alpha}_i^c = \mathrm{softmax}\left(\frac{(\mathbf{W}_q^c \mathbf{x}_i)(\mathbf{W}_k^c \mathbf{x})}{\sqrt{d}}\right). \tag{9}$$

It is flexible to stack multiple layers of self-attentions. Then a BiGRU network, f_G, is applied on top of $\bar{\mathbf{h}}_i$ to generate the final feature $\mathbf{h}_i$ incorporating sequential interactions:

$$\mathbf{h}_i = [\overrightarrow{\mathbf{h}}_i : \overleftarrow{\mathbf{h}}_i] = [f_G(\bar{\mathbf{h}}_i, \overrightarrow{\mathbf{h}}_{i-1}) : f_G(\bar{\mathbf{h}}_i, \overleftarrow{\mathbf{h}}_{i+1})]. \tag{10}$$

A softmax classifier is used to produce the final prediction for each token corresponding to the entity labels as

$$\mathbf{q}_i = \mathrm{softmax}(\mathbf{W}_t \mathbf{h}_i + \mathbf{b}_t). \tag{11}$$

Computational Linguistics Just Accepted MS.
https://doi.org/10.1162/coli_a_00415

For end-to-end event extraction, we use two separate classifiers for entity and event trigger prediction, respectively, which corresponds to two different sets of parameters: $\{\mathbf{W}_t^{ent}, \mathbf{b}_t^{ent}\}$ and $\{\mathbf{W}_t^{event}, \mathbf{b}_t^{event}\}$ in (11).

To generate entity-relation triplets, we first construct candidate entity pairs for each sentence by enumerating all the predicted entities. For each entity pair (ϵ_1, ϵ_2), the relation feature is a concatenation of its associated entity features, entity types and attention scores obtained through the transformer network:

$$\mathbf{h}_r = [\tilde{\mathbf{h}}_1; \tilde{\mathbf{h}}_2; \mathbf{u}_{\epsilon_1}; \mathbf{u}_{\epsilon_2}; \boldsymbol{\alpha}_{12}; \boldsymbol{\alpha}_{21}], \tag{12}$$

where $\tilde{\mathbf{h}}_1 = \frac{1}{|\epsilon_1|} \sum_{w_i \in \epsilon_1} \mathbf{h}_i$ with $|\epsilon_1|$ representing the number of words w_i within a candidate entity ϵ_1. Similarly, $\tilde{\mathbf{h}}_2$ is obtained from another candidate entity ϵ_2. $\mathbf{u}_{\epsilon_1}$ and $\mathbf{u}_{\epsilon_2}$ denote the entity type embedding for ϵ_1 and ϵ_2, respectively, by looking up an entity type embedding matrix $\mathbf{U}$ with $|\mathcal{E}|$ (the total number of entity types) columns which is randomly initialized and trained throught the learning process. The attention vector $\boldsymbol{\alpha}_{12}$ corresponds to the averaged multi-head attention score between $w_i \in \epsilon_1$ and $w_j \in \epsilon_2$, while $\boldsymbol{\alpha}_{21}$ records the reverse order of the 2 entities. The final prediction for relation r of the entity pair, i.e., the triplet $(\epsilon_1, r, \epsilon_2)$, is produced via

$$\mathbf{q}_r = \text{softmax}(\mathbf{W}_r \mathbf{h}_r + \mathbf{b}_r). \tag{13}$$

For end-to-end event extraction, the event argument relation triplet (ϵ, r, v) is generated in a similar manner by replacing ϵ_2 with event trigger v. We additionally use a binary classifier to decide whether there is a relation between the entity and event trigger due to the sparsity of relation labels.

5.2 Logic Network

As described in Section 5.1, the deep learning model only implicitly learns word correlations via high-level features and attentions, but ignores the explicit correlations among target variables, specially for those of different types. In fact, the entity/event labels are highly dependent on the relation types, e.g., "person$(w_i) \wedge$ work_for$(r_{(w_i,w_j)}) \Rightarrow$ organization(w_j)". Moreover, the segmentation labels are highly correlated within a context window. Although such segmentation interactions could be captured in $\mathcal{Q}$ via structured loss, it is more efficient and capable of modeling more complex correlations together with relation information. Here, we treat such segmentation dependencies as a form of knowledge reasoning.

Recently, some approaches have been proposed to combine deep learning with logic reasoning to regularize the learning process or induce new rules. However, most of them are not expressive enough by limiting themselves to the tasks within the logic domain, or are computationally expensive to work on real application domains. There is also a lack of focus on directly modeling rules for classifiers. For expressiveness, Shanahan et al. (2019) proposed a relational neural network, which only translates to a single logic rule that is propositional in nature. We propose a novel logic network within the logic module $\mathcal{P}$ that simulates FOL and enhances reasoning capabilities through multi-level rule constructions within a deep architecture.

As shown in Figure 2, $\mathcal{P}$ consists of a separate logic network (LNets) applied on each word and relation. Follow the introduction of FOL in Section 3.3, we first adapt the problem into the logic domain, where a logic variable corresponds to a word w

Computational Linguistics Just Accepted MS.
https://doi.org/10.1162/coli_a_00415

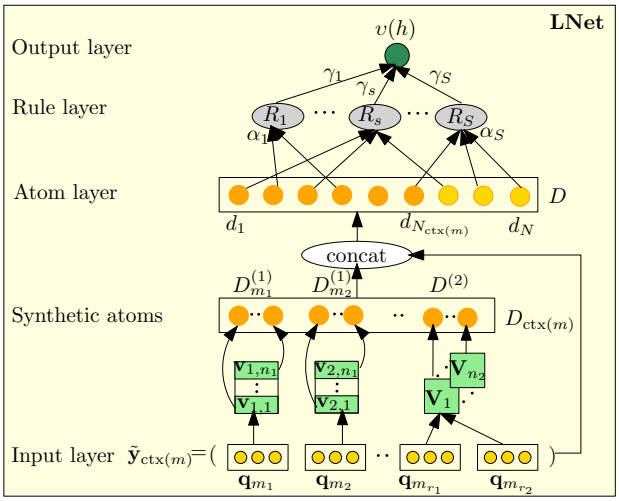

Figure 3
An example of a logic network. The input consists of prediction vectors $\mathbf{q}_i$'s from module $\mathcal{Q}$. The synthetic atoms consist of $\{D^{(1)}_{m_i}\}$'s, consisting of 1-ary synthetic atoms, the value of which is obtained by transforming from $\mathbf{q}_{m_i}$ using a vector $\mathbf{v}_{i,n}$, and $D^{(2)}$, a set of 2-ary synthetic atoms, the value of which is obtained by transforming from 2 input vectors $\mathbf{q}_{m_{r_1}}$, $\mathbf{q}_{m_{r_2}}$ using a bilinear matrix $\mathbf{V}_n$. The set of synthetic atoms can be combined with pre-defined atoms via a concatenation operation to form the atom layer. The rule layer then selects relevant atoms in the atom layer to form logic rules to generate the final value $\upsilon(h)$ for the head atom.

or a relational triplet $r_{(\epsilon_1,\epsilon_2)}$. All possible words and relations form the set of logical constants. Each target class $y \in E \cup \mathcal{V} \cup \mathcal{R}$ could be regarded as a predicate, and when taking constants as arguments, becomes a grounded atom. When the target class $y \in E \cup \mathcal{V}$ is an entity type or event type, it takes a single word (or phrase) as the argument, e.g., $person$(W. Dale Nelson) with $y =$ person. When the target class $y \in \mathcal{R}$ is a relation, it takes a relational triplet as the argument. For example, work_for$(r_{(\epsilon_1,\epsilon_2)})$ with $y =$ work_for specifies entity ϵ_1 and entity ϵ_2 has relation *work_for*. We use $d(x_1, ..., x_n)$ to denote an n-ary atom and $\upsilon(d) \in [0, 1]$ to denote the probability of the atom being true. For example, υ(work_for$(r_{(\epsilon_1,\epsilon_2)}) = 0.8$ indicates that ϵ_1, and ϵ_2 has relation "work_for" with probability 0.8.

As discussed in Section 3.3, we treat each target class as a form of logic entailment where the target class is the head atom h of a set of logic rules/formula $R \in \{R_1, ..., R_S\}$ with $R : d_1 \wedge d_2 \wedge ... \wedge d_T \Rightarrow h$. Here R is a rule identifier. As a concrete example, if we aim to predict whether a text segment ϵ_j belongs to the target class "organization (entity)", we may define a logic rule to entail the target entity type "organization": person$(\epsilon_i) \wedge$ work_for$(r_{(\epsilon_i,\epsilon_j)}) \Rightarrow$ organization(ϵ_j), where the head atom $h =$ organization(ϵ_j) corresponds to the target entity type. The result depends on its precondition which consists of two atoms $d_1 =$ person(ϵ_i) and $d_2 =$ work_for$(r_{(\epsilon_i,\epsilon_j)})$. Then an FOL program aims to produce the truth probability of h given the set of all possible rules $\{R_1, ..., R_S\}$. In most cases, such rules may not be readily available. Hence, it is desirable to learn the FOL rules automatically. To achieve that, we use a separate logic network (LNet) to generate relevant rules corresponding to the same head atom and evaluates its truth probability through its preconditions.

Computational Linguistics Just Accepted MS.
https://doi.org/10.1162/coli_a_00415

The detailed computation process for each LNet is shown in Figure 3. For a logic constant $m \in \mathcal{N}_\epsilon \cup \mathcal{N}_r$ referring to either a word or a relation, we build a set consisting of its relevant contexts $\mathrm{ctx}(m) = \{m_1, ..., m_{|\mathrm{ctx}(m)|}\}$. Then the input to a LNet becomes $\tilde{\mathbf{y}}_{\mathrm{ctx}(m)} = (\mathbf{q}_{m_1}, \mathbf{q}_{m_2}, ..., \mathbf{q}_{m_{|\mathrm{ctx}(m)|}})$ which combines deep learning predictions of each element in $\mathrm{ctx}(m)$.[1] The LNet aims to produce probabilistic evaluations of a set of N atoms $D = \{d_1, ..., d_N\}$ in the atom layer, which are in turn used to form a logic program consisting of a set of logic rules $\{R_1, ..., R_S\}$ of the form $d_{j_1} \wedge ... \wedge d_{j_T} \Rightarrow h$. All these rules share an identical head atom $h = y_m$ that indicates whether m belongs to a target class $y_m \in E \cup \mathcal{V} \cup \mathcal{R}$. The final output is a probabilistic evaluation $\upsilon(h)$ by accumulating all the logic rules considering their confidence scores $\gamma_1, ..., \gamma_S$. In this way, the LNet is able to model the correlations of related constants formed by each word's or relation's relevant contexts.

As shown in Figure 3, to produce the set of N atoms, each LNet firstly creates a set of 1-ary atoms $D^{(1)}_{m_i} = \{d^{(1)}_{i,1}, ..., d^{(1)}_{i,n_1}\}$ corresponding to a context $m_i \in \mathrm{ctx}(m)$, where $d^{(1)}_{i,n} \in D^{(1)}_{m_i}$ records a unique property of its corresponding input m_i. It also produces a set of 2-ary atoms $D^{(2)} = \{d^{(2)}_1, ..., d^{(2)}_{n_2}\}$, where $d^{(2)}_n \in D^{(2)}$ indicates a relation between 2 interacting contexts $m_{r_1}, m_{r_2} \in \mathrm{ctx}(m)$. The value for each atom is computed automatically via parameterized transformations [2]:

$$\upsilon(d^{(1)}_{i,n}) = \sigma(\mathbf{v}_{i,n}{}^\top \mathbf{q}_{m_i}), \tag{14}$$

$$\upsilon(d^{(2)}_{n}) = \sigma(\mathbf{q}_{m_{r_1}}{}^\top \mathbf{V}_n \mathbf{q}_{m_{r_2}}), \tag{15}$$

where σ is the sigmoid function for probabilistic evaluations. $\mathbf{v}_{i,n} \in \mathbb{R}^{|\mathbf{q}_{m_i}|}$ and $\mathbf{V}_n \in \mathbb{R}^{|\mathbf{q}_{m_{r_1}}| \times |\mathbf{q}_{m_{r_2}}|}$ are transformation parameters that convert the input vector to a scalar. We can view (14) and (15) as computing the probability of a specific property of the input (e.g., $\mathbf{q}_{m_i}$) being true. Then $D_{\mathrm{ctx}(m)} = \{D^{(1)}_{m_i}\}_{m_i \in \mathrm{ctx}(m)} \cup D^{(2)}$ could be regarded as recording different properties or relationships of the input context $\mathrm{ctx}(m)$. We call these automatically-generated atoms $D_{\mathrm{ctx}(m)}$ as synthetic atoms.

Take the sentence "*W. Dale Nelson covers the White House for The Associated Press*" as an example. To make predictions on the word *Dale* in Module $\mathcal{P}$, we first identify its context $\mathrm{ctx}(\text{Dale}) = \{\text{W., Dale, Nelson, The Associated Press}, r_{(\text{Dale, The Associated Press})}\}$ if *The Associated Press* is extracted as an entity. Then the input $\tilde{\mathbf{y}}_{\mathrm{ctx}(\text{Dale})}$ is the concatenation of all the prediction vectors $\mathbf{q}$'s corresponding to each element in $\mathrm{ctx}(\text{Dale})$ obtained from module $\mathcal{Q}$: $\tilde{\mathbf{y}}_{\mathrm{ctx}(\text{Dale})} = (\mathbf{q}_{\text{W.}}, \mathbf{q}_{\text{Dale}}, \mathbf{q}_{\text{Nelson}}, \mathbf{q}_{\text{The Associated Press}}, \mathbf{q}_{r_{(\text{Dale, The Associated Press})}})$. Given $\tilde{\mathbf{y}}_{\mathrm{ctx}(\text{Dale})}$, the values of the synthetic atoms are obtained in the following process. Specifically, for the first input vector $\mathbf{q}_{\text{W.}}$ corresponding to the previous word of *Dale*, we produce $D^{(1)}_{\text{W.}} = \{d^{(1)}_{1,1}, ..., d^{(1)}_{1,n_1}\}$ with values $\upsilon(d^{(1)}_{1,1}) = \sigma(\mathbf{v}_{1,1}{}^\top \mathbf{q}_{\text{W.}}), ..., \upsilon(d^{(1)}_{1,n_1}) = \sigma(\mathbf{v}_{1,n_1}{}^\top \mathbf{q}_{\text{W.}})$ corresponding to n_1 different properties the previous word of *Dale*, according to (14). We treat these atoms as unary synthetic atoms. In a similar manner, we obtain $D^{(1)}_{\text{Dale}}$ and $D^{(1)}_{\text{Nelson}}$ as another 2 sets of n_1 1-ary atoms. Each of the n_2 produced 2-ary atom $d^{(2)}_n \in D^{(2)}$ corresponds to an interaction property among *Dale*, *The Associated*

1 How to construct relevant context for each $m \in \mathcal{N}_\epsilon \cup \mathcal{N}_r$ is explained in detail in Section 6.
2 Here n_1 and n_2 are hyper-parameters corresponding to the number of 1-ary atoms and 2-ary atoms, respectively, for each input.

Computational Linguistics Just Accepted MS.
https://doi.org/10.1162/coli_a_00415

Press and $r_{(\text{Dale,The Associated Press})}$ via $\upsilon(d_n^{(2)}) = \sigma(\mathbf{q}_{\text{The Associated Press}}{}^{\top}\mathbf{V}_n\mathbf{q}_{r_{(\text{Dale,The Associated Press})}})$, according to (15).

To make the logic network flexible and comprehensive, we further enhance LNet with residual connections to incorporate pre-defined atoms and logic rules when provided. As shown in Figure 3, a *concat* operation concatenates the synthetic atoms and original inputs to form the atom layer $D = \{d_1, ..., d_{N_{\text{ctx}(m)}}, d_{N_{\text{ctx}(m)+1}}, ..., d_N\}$, where the first $N_{\text{ctx}(m)} = |D_{\text{ctx}(m)}|$ atoms are the synthetic atoms, while the last $N - N_{\text{ctx}(m)}$ atoms are the pre-defined atoms. Different from synthetic atoms which do not have exact semantic meanings, the pre-defined atoms are formed by the original inputs $\mathbf{q}_i$ specifying the probability of each target class corresponding to the input word/relation, e.g., $d_j = \text{person}(\epsilon), N_{\text{ctx}(m)} + 1 \leq j \leq N$ will inherit the value as $\upsilon(d_j) = (\mathbf{q}_\epsilon)_{[person]}$ which indicates the probability of label *person* for ϵ. The pre-defined atoms facilitates the incorporation of prior knowledge, e.g., $\text{person}(\epsilon_i) \wedge \text{work\_for}(r_{(\epsilon_i,\epsilon_j)}) \Rightarrow \text{organization}(\epsilon_j)$, into the rule layer.

The rule layer aims to learn a set of logic rules $\{R_1, ..., R_S\}$ corresponding to the same head atom h by choosing proper body atoms from D. It consists of two kinds of logic rules: learned rules and fixed rules. The learned rules are constructed based on an iterative selection process via attention mechanism. Given the set of all possible atoms D, a logic rule $R : d_{j_1} \wedge, ..., \wedge d_{j_T} \Rightarrow h$ is formed by learning to select $d_{j_t} \in D$ at each iteration $t \in \{1, ..., T\}$. The selection process is parameterized and approximated using attention mechanism, where a trainable weight vector $\boldsymbol{\beta}^t \in \mathbb{R}^N$ is used to record the relevance of all N atoms in D at each iteration t. To achieve sparse selection, we adopt sparsemax (Martins and Astudillo 2016):

$$\text{sparsemax}(\boldsymbol{\beta}^t) = \text{argmin}_{x \in \Delta^{N-1}} \|\boldsymbol{x} - \boldsymbol{\beta}^t\|^2, \tag{16}$$

with $\Delta^{N-1} = \{\boldsymbol{x} \in \mathbb{R}^N | \mathbf{1}^\top \boldsymbol{x} = 1, \boldsymbol{x} \geq \mathbf{0}\}$. Intuitively, $\text{sparsemax}(\boldsymbol{\beta}^t)$ transforms $\boldsymbol{\beta}^t$ to a sparse probabilistic vector indicating the most relevant atoms to be selected. Note that $\boldsymbol{\beta}^t$ is randomly initialized and trained throughout the learning process. To produce the value for the head atom h, we follow (5) to obtain

$$\begin{aligned} \upsilon_R(h) &= \min\{\upsilon(d_{j_1}), ..., \upsilon(d_{j_T})\} \\ &= \min_{1 \leq t \leq T}\{\mathbf{d}^\top \text{sparsemax}(\boldsymbol{\beta}^t)\}, \end{aligned} \tag{17}$$

where $\mathbf{d} = [\upsilon(d_1), ..., \upsilon(d_N)]$ denotes the vector of atom evaluations. Specifically, for each timestamp t, $\mathbf{d}^\top \text{sparsemax}(\boldsymbol{\beta}^t) \approx \upsilon(d_t)$ when $\boldsymbol{\beta}^t$ assigns the most probabilistic mass to d_t, which should be expected using sparsemax, compared to softmax. Then after T iterations of selection, the resulted logic rules should have the form $d_{j_1} \wedge ... \wedge d_{j_T} \Rightarrow h$. Fixed rules correspond to prior knowledge which is manually constructed. These rules can be used to enhance the final prediction when the learned rules are not expressive enough. To incorporate such prior knowledge, we transform the body atoms of each given rule into 1-hot attention weights. For example, for $\text{person}(\epsilon_i) \wedge \text{work\_for}(r_{(\epsilon_i,\epsilon_j)}) \Rightarrow \text{organization}(\epsilon_j)$, we construct its corresponding attention weight $\bar{\boldsymbol{\beta}}^1 = \mathbb{1}(d_i = \text{person})$, $\bar{\boldsymbol{\beta}}^2 = \mathbb{1}(d_j = \text{work\_for})$, where $\mathbb{1}(\cdot)$ is a indicator function. These weight vectors are kept fixed during training.

The final value for h considering all the relevant rules $\{R_1, ..., R_S\}$ is obtained via

$$\upsilon(h) = \boldsymbol{\gamma}^\top [\upsilon_{R_1}(h), ...\upsilon_{R_S}(h)], \tag{18}$$

Computational Linguistics Just Accepted MS.
https://doi.org/10.1162/coli_a_00415

where $\boldsymbol{\gamma} \in \mathbb{R}^S$ indicates the confidence level for each rule and is trainable. $\upsilon(h)$ can be regarded as a binary classifier for its corresponding target class, e.g., "organization (entity)". We use the same atom set D with different attention vectors to parameterize different target classes. Denote by $\upsilon(h_l)$ the output from a LNet for each target class l, we can produce the final multi-class predictions $\mathbf{p}_m$ in module $\mathcal{P}$ as

$$\mathbf{p}_m = \text{softmax}([\upsilon(h_1), ..., \upsilon(h_{\text{L}(m)})]) \tag{19}$$

for $m \in \mathcal{N}_r \cup \mathcal{N}_\epsilon$. The number of nodes in the output layer is $\text{L}(m) = |E|$ for entity ($m \in \mathcal{N}_\epsilon$), $\text{L}(m) = |\mathcal{V}|$ for event trigger ($m \in \mathcal{N}_\epsilon$) and $\text{L}(m) = |\mathcal{R}|$ for relation ($m \in \mathcal{N}_r$).

As both shown in Figure 2 and (19), the output of LNet $\upsilon(h)$ is used to produce the probabilistic vector $\mathbf{p}_m$ as the output of module $\mathcal{P}$ for each constant m. These probabilistic vectors $\mathbf{p}_m$'s, together with the outputs $\mathbf{q}_m$'s from module $\mathcal{Q}$ will further be used to train our joint model via the EM algorithm, as discussed in the sequel.

6. Learning with Expectation-Maximization

6.1 Inference

The E-step involves inference and update of module $\mathcal{Q}$ by taking the output $\mathbf{p}_m$ from the logic module $\mathcal{P}$. Recall from Section 3.2, the objective is to solve $q(Z;\theta) = p(Z|Y;\phi)$ with p fixed. Here we have $Z = \{\mathbf{y}_m\}_{m\in\mathcal{N}_\epsilon\cup\mathcal{N}_r}$ corresponds to the predictions for all the words $\mathcal{N}_\epsilon$ and relations $\mathcal{N}_r$, and $Y = \{\mathbf{y}_{\text{ctx}(m)}\}_{m\in\mathcal{N}_\epsilon\cup\mathcal{N}_r}$, with $\text{ctx}(m)$ denoting the context of node m which will be introduced later. Using the mean-field variational approximation, the above probabilities factorizes as $q(Z) = \prod_{m\in\mathcal{N}_\epsilon\cup\mathcal{N}_r} q(\mathbf{y}_m)$ and $p(Z|Y) = \prod_{m\in\mathcal{N}_\epsilon\cup\mathcal{N}_r} p(\mathbf{y}_m|\mathbf{y}_{\text{ctx}(m)})$ (θ and ϕ are omitted for ease of illustration). To train our model with EM algorithm, we associate $p(\mathbf{y}_m|\mathbf{y}_{\text{ctx}(m)})$ with the logic module $\mathcal{P}$ such that $p(\mathbf{y}_m = y|\mathbf{y}_{\text{ctx}(m)}) = (\mathbf{p}_m)_{[y]}$ representing the probability when m has label y, where $\mathbf{p}_m$ is the output of the logic module obtained from (19). The subscription $[y]$ denotes the y-th entry of the corresponding vector. Similarly, we associate $q(\mathbf{y}_m)$ with module $\mathcal{Q}$, where $q(\mathbf{y}_m = y) = (\mathbf{q}_m)_{[y]}$. For relation prediction, $q(\mathbf{y}_r = y) = (\mathbf{q}_r)_{[y]}$ with $r \in \mathcal{N}_r$. Note that the bold symbols (e.g., $\mathbf{y}_m, \mathbf{y}_r, \mathbf{y}$) within distributions $p(\cdot)$ and $q(\cdot)$ denote random variables for label predictions and the non-bold symbols (e.g., y_i, y) indicate the actual label assignment. According to (Qu, Bengio, and Tang 2019), the optimal solution satisfies the approximated condition:

$$\log q\left(\mathbf{y}_m\right) \approx \mathbb{E}_{q\left(\mathbf{y}_{\text{ctx}(m)}\right)}\left[\log p\left(\mathbf{y}_m|\mathbf{y}_{\text{ctx}(m)}\right)\right] \tag{20}$$

for $m \in \mathcal{N}_r \cup \mathcal{N}_\epsilon$ with $\mathcal{N}_r$ and $\mathcal{N}_\epsilon$ denoting the set of relations and words, respectively. Here θ and ϕ is omitted for ease of illustration. The above condition can be further converted to

$$\log q\left(\mathbf{y}_m\right) \approx \log p\left(\mathbf{y}_m|\tilde{\mathbf{y}}_{\text{ctx}(m)}\right) \tag{21}$$

by approximating the expectation via sampling $\tilde{\mathbf{y}}_{\text{ctx}(m)}$ from $q(\mathbf{y}_{\text{ctx}(m)})$ in module $\mathcal{Q}$. Intuitively, (21) aims to align the distributions from two modules. To update q in terms

Computational Linguistics Just Accepted MS.
https://doi.org/10.1162/coli_a_00415

of θ, we minimize the following objective fixing p:

$$O_E = -\sum_{m \in \mathcal{N}_\epsilon \cup \mathcal{N}_r} \mathbb{E}_{p(\mathbf{y}_m|\tilde{\mathbf{y}}_{\mathrm{ctx}(m)})} \log q(\mathbf{y}_m), \tag{22}$$

To achieve that, we first generate the prediction $y_m = \mathrm{argmax}_y p(\mathbf{y}_m = y|\tilde{\mathbf{y}}_{\mathrm{ctx}(m)})$ through the logic module $\mathcal{P}$ using (19) given $\tilde{\mathbf{y}}_{\mathrm{ctx}(m)}$ (will be discussed later). We use the predicted label y_m as the target label to train module $\mathcal{Q}$, replacing $\mathbb{E}_{p(\mathbf{y}_m|\tilde{\mathbf{y}}_{\mathrm{ctx}(m)})} \log q(\mathbf{y}_m)$ in (22) with $\log q(\mathbf{y}_m = y_m)$.

During training, as the ground-truths are available, we also utilize label information to update q. Specifically for each m, we update q using the aforementioned strategy with probability 0.5, otherwise we replace y_m (or $\{y_1, ..., y_n\}$) predicted from $\mathcal{P}$ with the ground-truth label to update q.

6.2 Learning

The M-step involves learning and update of module $\mathcal{P}$. Recall the objective in (3), we fix $\mathcal{Q}$ and use it to update $\mathcal{P}$. Assuming conditional independence given the contexts, we have

$$\log p(Y, Z) = \sum_{m \in \mathcal{N}_r \cup \mathcal{N}_\epsilon} \log p(\mathbf{y}_m|\mathbf{y}_{\mathrm{ctx}(m)}). \tag{23}$$

Then the objective of maximizing (3) becomes

$$O_M = \mathbb{E}_{q(Z)}[\log p(Y, Z)] \approx \sum_{m \in \mathcal{N}_r \cup \mathcal{N}_\epsilon} \log p(\mathbf{y}_m = y_m|\tilde{\mathbf{y}}_{\mathrm{ctx}(m)}). \tag{24}$$

Here $y_m = \mathrm{argmax}_y q(\mathbf{y}_m = y)$ is the predicted label which corresponds to the maximum probability in $q(\mathbf{y}_m)$ from module $\mathcal{Q}$. To avoid randomness brought by sampling $\tilde{\mathbf{y}}_{\mathrm{ctx}(m)}$, we directly use the real-valued outputs given by module $\mathcal{Q}$ such that $\tilde{\mathbf{y}}_{\mathrm{ctx}(m)} = (\mathbf{q}_{m_1}, ..., \mathbf{q}_{m_{|\mathrm{ctx}(m)|}})$ with $\mathrm{ctx}(m) = \{m_1, ..., m_{|\mathrm{ctx}(m)|}\}$. This aligns with the input of the logic network shown in Figure 2. (24) can be viewed as maximizing the log-likelihood of the predictions given by $\mathcal{Q}$ using module $\mathcal{P}$. To incorporate label supervision, we use y_m with probability 0.5, otherwise we replace y_m in (24) with m's true label to update p.

To compute $p(\mathbf{y}_m|\tilde{\mathbf{y}}_{\mathrm{ctx}(m)})$ within the logic module $\mathcal{P}$, we define the context $\mathrm{ctx}(m)$ of each variable m to be those variables that have intensive correlations with m for the task at hand by constructing some rule templates given the output $\{\mathbf{q}_i\}$'s from module $\mathcal{Q}$. When $m = w_i \in \mathcal{N}_\epsilon$, we use 3 types of dependencies for the rule templates:

- The prediction of a word w_i from $\mathcal{Q}$ is a direct precondition: $\mathbf{q}_i \rightarrow \mathbf{p}_i$.
- The prediction of another word w_j from $\mathcal{Q}$ that has relation with w_i could inform the target prediction: $\mathbf{q}_j, \mathbf{q}_{r_{ij}} \rightarrow \mathbf{p}_i$.
- The prediction of w_i's preceding and following words from $\mathcal{Q}$ could inform the target prediction: $\mathbf{q}_{i-1}, \mathbf{q}_{i+1} \rightarrow \mathbf{p}_i$. Note that this type of dependencies is applicable when the structured prediction is implemented in the logic module ($\mathcal{P}$) not the deep learning module ($\mathcal{Q}$).

For relation prediction when $m = r_{ij} \in \mathcal{N}_r$, we use 2 templates:

Computational Linguistics Just Accepted MS.
https://doi.org/10.1162/coli_a_00415

- The prediction of r_{ij} from $\mathcal{Q}$ is a direct precondition: $\mathbf{q}_{r_{ij}} \rightarrow \mathbf{p}_{r_{ij}}$.
- The predictions of w_i and w_j from $\mathcal{Q}$ could inform the target: $\mathbf{q}_i, \mathbf{q}_j \rightarrow \mathbf{p}_{r_{ij}}$.

Given these dependency templates, we construct the input of the logic network $\tilde{\mathbf{y}}_{\text{ctx}(m)}$ for $m = w_i \in \mathcal{N}_\epsilon$ as $\tilde{\mathbf{y}}_{\text{ctx}(m)} = (\mathbf{q}_{i-1}, \mathbf{q}_i, \mathbf{q}_{i+1}, \mathbf{q}_j, \mathbf{q}_{r_{ij}})$ for entity (or event) prediction of each word w_i, where $\mathbf{q}_{i-1}$, $\mathbf{q}_i$ and $\mathbf{q}_{i+1}$ are separately used to construct 1-ary atoms, and both $\mathbf{q}_j$ and $\mathbf{q}_{r_{ij}}$ are used to construct 2-ary atoms in module $\mathcal{P}$. Intuitively, the corresponding words and relations form the context of w_i, denoted by $\text{ctx}(i) = \{w_{i-1}, w_i, w_{i+1}, w_j, r_{ij}\}$. Similarly, the input $\tilde{\mathbf{y}}_{\text{ctx}(m)}$ when $m = r_{ij} \in \mathcal{N}_r$ for relation prediction of r_{ij} is $\tilde{\mathbf{y}}_{\text{ctx}(m)} = (\mathbf{q}_{r_{ij}}, \mathbf{q}_i, \mathbf{q}_j)$. We use $\mathbf{q}_{r_{ij}}$, $\mathbf{q}_i$, $\mathbf{q}_j$, respectively, to produce 1-ary atoms. Again, both $\mathbf{q}_i$ and $\mathbf{q}_j$ are used to produce 2-ary atoms. Given the construction of $\tilde{\mathbf{y}}_{\text{ctx}(m)}$, the output $\mathbf{p}_m$ of the logic network will then be computed following (19).

6.3 Optimization

Overall, the training process involves alternating between variational E-step and M-step to update module $\mathcal{P}$ using (24) and module $\mathcal{Q}$ using (22). For both steps, the output of $\mathcal{P}$ is obtained by sampling the context predictions of the target using $\mathcal{Q}$, which reflects the intensive interactions between these two modules. This interaction is also enhanced by learning to approximate these two distributions throughout the training process. To facilitate training, we firstly pretrain $\mathcal{Q}$ using the ground-truth labels for several iterations before the variational EM procedure. In the testing phase, both $\mathcal{P}$ and $\mathcal{Q}$ can be adopted to generate the predictions. In our experiments, we use a similar strategy as ensemble learning that assigns each module a weight which is tuned according to the validation set to compute a weighted average of the two modules as our final predictions. The complete training procedure for end-to-end relation extraction is shown in Algorithm 1.

7. Experiment

7.1 Tasks & Data

We conduct experiments on 6 benchmark datasets from 3 IE tasks:

- **Aspect and Opinion Terms Extraction**: Aspect terms refer to the product features or attributes that the users commented on in the customer reviews. Opinion terms are those carrying subjective opinions towards the products or services. For example, given a review sentence "*The service staff is terrible.*", *service staff* is an aspect term and *terrible* is the opinion term. We use a **restaurant** review corpus and a **laptop** review corpus from SemEval 2014 (Pontiki et al. 2014). The statistics of the two datasets are shown in Table 2.
- **End-to-end relation extraction**: This task involves the identification and classification of both entities and relations between entities. For this task, three benchmark datasets are used, including CoNLL04 (Roth and Yih 2004), ACE04 (Doddington et al. 2004) and ACE05 (Li and Ji 2014). As shown in Table 3, CoNLL04 consists of 4 entity types and 5 relation categories. ACE04 defines 7 entity types with 7 relation categories and

Computational Linguistics Just Accepted MS.
https://doi.org/10.1162/coli_a_00415

Algorithm 1 Variational Deep Logic Network

Input: A sequence of input words $\{w_1, ..., w_n\}$ and their corresponding word embeddings $\mathbf{x} = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$. Ground-truth entity label for each word $\{y_1, ..., y_n\}$. Relation label for each candidate entity pairs $\{y_{r_{(\epsilon_1,\epsilon_2)}}\}$'s.
Output: Trained parameters θ for Module $\mathcal{Q}$, and ϕ for Module $\mathcal{P}$.

1: Pretrain Module $\mathcal{Q}$
for $k = 1, 2, ..., K$ **do**
Produce hidden representations $\{\mathbf{h}_1, ..., \mathbf{h}_n\}$ from self-attentions and Bi-GRU.
Produce softmax probabilities $\{\mathbf{q}_1, ..., \mathbf{q}_n\}$ using (11).
Compute loss via cross-entropy $\mathcal{L}^E_{pretrain} = -\sum_{i=1}^{n} \mathbf{y}_i \log \mathbf{q}_i$.
Produce softmax predictions for relations from candidate entity pairs as $\{\mathbf{q}_r\}$'s using (13).
Compute relation loss via cross-entropy $\mathcal{L}^R_{pretrain} = -\sum_r \mathbf{y}_r \log \mathbf{q}_r$.
Update $\theta := \theta - \delta \frac{\partial \mathcal{L}^E_{pretrain} + \mathcal{L}^R_{pretrain}}{\partial \theta}$ where δ is the learning rate.
end for

2: EM training between $\mathcal{P}$ **and** $\mathcal{Q}$
Generate $\tilde{\mathbf{y}}_{\mathrm{ctx}(m)} = (\mathbf{q}_{m_1}, ..., \mathbf{q}_{m_{\mathrm{ctx}(m)}})$ from outputs of module $\mathcal{Q}$.

for each iteration **do**

3: Inference
for $k = 1, 2, ..., K'$ **do**
Produce probabilistic output $\mathbf{p}_m$ in module $\mathcal{P}$ from LNets using (19).
Obtain predictions $y_m = \mathrm{argmax}_y p(\mathbf{y}_m = y | \tilde{\mathbf{y}}_{\mathrm{ctx}(m)}) = \mathrm{argmax}_y (\mathbf{p}_m)_{[y]}$.
Update module $\mathcal{Q}$ via $\theta := \theta - \delta_q \frac{\partial O_E}{\partial \theta}$, where δ_q is the learning rate for the E-step. O_E is obtained through (22).
end for

4: Learning
for $k = 1, 2, ..., K'$ **do**
Obtain predicted label $y_m = \mathrm{argmax}_y q(\mathbf{y}_m = y)$ from module $\mathcal{Q}$.
Compute distribution $p(\mathbf{y}_m | \tilde{\mathbf{y}}_{\mathrm{ctx}(m)}) = \mathbf{p}_m$ from module $\mathcal{P}$ using (19).
Update module $\mathcal{P}$ via $\phi := \phi - \delta_p \frac{\partial O_M}{\partial \phi}$, where δ_p is the learning rate for the M-step. O_M is obtained through (24).
end for
end for

ACE05 adopts the same entity types as ACE04 but defines 6 relation types. CoNLL04 and ACE04 do not provide official train/test split, hence we conduct 3-fold and 5-fold cross-validation for CoNLL04 and ACE04, respectively to report our final results. We follow the same preprocessing and data split as (Li and Ji 2014) on ACE05 dataset.

- **End-to-end event extraction**: This task involves three subtasks, namely extraction and classification of entity mentions, extraction and classification of event triggers and discovering of relationships between entity mentions and event triggers for event argument extraction and classification. For this task, the same ACE05 dataset is used. For entity

Computational Linguistics Just Accepted MS.
https://doi.org/10.1162/coli_a_00415

Data		# Sentences	Entity Type
Restaurant 14	train test	3041 800	aspect, opinion
Laptop 14	train test	3045 800	aspect, opinion

Table 2
Dataset statistics for aspect and opinion terms extraction.

Data		# Sentences	# Entities	# Relation	Entity Type	Relation Type
CoNLL04		1,437	5,336	2,040	person, location, organization, other	located_in, org_based_In, work_for, live_in, kill
ACE04		6,789	22,740	4,368	person, vehicle, organization, location, facility, weapon, geographical entity	physical, PER/ORG-affiliation, employment-organization, person-social, GPE-affiliation, Agent-Artifact, discourse
ACE05	train dev test	7,273 1,765 1,535	26,470 6,421 5,476	4,779 1,179 1,147	person, vehicle, organization, location, facility, weapon, geographical entity	physical, ORG-affiliation, employment-organization, agent-artifact, part-whole, person-social, GPE-affiliation

Table 3
Dataset statistics for end-to-end relation extraction..

Data		# Sentences	# Entities	# Triggers	# Argument
ACE05	train	14,837	48,797	4,337	7,768
	dev	863	3,917	497	933
	test	672	4,184	438	911

Table 4
Dataset statistics for end-to-end event extraction.

mentions, we consider ACE entity types PER, ORG, GPE, LOC, FAC, VEH, WEA and ACE VALUE and TIME expressions, following the common setting of the existing works. There are in total 33 event subtypes that are involved in the event trigger classification task. The total number of different argument roles for entities participating in various events is 35 and we collapse 8 of them that are time-related, following (Yang and Mitchell 2016). The detailed statistics of ACE05 dataset for event extraction is shown in Table 4. For evaluation, we treat an entity as correct if both its entity type and offset matches one of the ground-truth entities. An event trigger is correctly identified if its offset matches one of the reference event triggers, and it is regarded as correctly classified if its type is also correct. An argument role is correctly identified if the corresponding entity type, entity offset and event type matches one of the reference argument roles, and it is correctly classified if the argument role is also correct.

Computational Linguistics Just Accepted MS.
https://doi.org/10.1162/coli_a_00415

Model		GInf	Rule-distill	DLogic	DLogic*	VDLN	SOTA (Chen and Qian 2020)
Res14	Aspect	84.50	*87.27**	*85.41**	*86.57**	***87.71***	86.71
	Opinion	85.20	*86.40**	*84.21**	*86.11**	***87.32***	87.18
Lap14	Aspect	78.69	*81.05**	*81.01**	*81.25**	***82.44***	82.34
	Opinion	79.89	*80.56**	*79.57**	*79.89**	***81.40***	81.00

Table 5
Results on 2 benchmark datasets for aspect and opinion extraction. The *italic* numbers show the average results over 3 different runs. * indicates the results are significant with $p < 0.05$.

7.2 Experimental Setting

To integrate self-attention mechanism, we use a pretrained BERT model (base-uncased) (Devlin et al. 2019) to initialize all the word embeddings and to produce the attention scores for each pair of words. The batch size is 20 and the dimension for BiGRU is 100 with dropout rate 0.1. For the logic network, we set the number of 1-ary atoms (n_1) and 2-ary atoms (n_2) for each input variable to 20 and the number of body atoms in each rule as $T = 8$. The total number of rules is set to $S = 30$. During training, we use adadelta with initial learning rate 0.01 for module $\mathcal{Q}$ and adam with initial learning rate 0.01 for module $\mathcal{P}$. The sampling rate for both E-step and M-step is set to 0.5, i.e., for 50% of the time, the ground-truth label is used for learning desired modules. For each experiment, we first pretrain $\mathcal{Q}$ for 50 epochs and then alternate between $\mathcal{P}$ and $\mathcal{Q}$ with every 2 epochs for each module. The final prediction is made by ensemble strategy with weight 0.6 and 0.4 for $\mathcal{Q}$ and $\mathcal{P}$ respectively. All the hyper-parameters are selected via the validation set. For evaluation, we use micro-F1 scores on non-negative classes. An entity is correct if both segmentation and entity type are correct. A relation is correct if both of its entities (events) and the relation type matches the ground-true label. We use the same evaluation metric as (Yang and Mitchell 2016) for event extraction.

For time complexity, we report the duration using 1 GPU of Tesla V100 250W. Pure neural model (e.g., BERT) takes 38s and 28s for training 1 epoch on Res14 and Lap14 dataset, respectively. It takes 556s and 522s for training 1 iteration of VDLN that consists of 2 epochs of both modules on Res14 and Lap14 dataset, respectively. On CoNLL04, it takes 56s for training 1 epoch of BERT and 271s for training 1 iteration of VDLN. On ACE04, it takes 131s for training 1 epoch of BERT and 967s for training 1 iteration of VDLN. On ACE05, it takes 242s for training 1 epoch of BERT and 1713s for training 1 iteration of VDLN. For memory usage, experiments on Res14 and Lap14 data occupy around 8.9G. Experiment on CoNLL04 takes 15.9G. Experiments on ACE04 and ACE05 occupy around 6.7G. All these memory usages are almost the same when compared with pure deep learning models.

7.3 Result

Aspect and Opinion Terms Extraction: To demonstrate the effectiveness of our proposed model, we compare with the following most recent baselines:

- **GInf**: A pipelined model combining deep neural networks with integer linear programming (Yu, Jiang, and Xia 2019). The predictions produced from the deep neural networks are taken as input to the integer linear

Computational Linguistics Just Accepted MS.
https://doi.org/10.1162/coli_a_00415

programming system where explicit relational constraints among aspect terms and opinion terms are enforced considering syntactic information.

- **Rule-distill**: A posterior-regularization-based framework to regularize deep learning predictions via prior knowledge. The training is conducted via a teach-student knowledge distillation (Hu et al. 2016). To adapt this model to our problem setting, we construct a few logic rules, as shown in Table 14 to form the teacher network. For fairness, we use the same neural model (module $\mathcal{Q}$) as the student network.
- **DLogic**: A joint model incorporating explicit logic rules into the deep learning model (Wang and Pan 2020). The deep learning predictions are made as probabilistic evaluations of input atoms to produce the output for the head atom of each rule. Then a discrepancy loss is computed to align the deep learning predictions with a set of pre-defined logic rules.
- **DLogic***: Replace the deep neural networks of DLogic with the the one used in our proposed model for fair evaluations.
- **VDLN**: The proposed model consisting of a logic module $\mathcal{P}$ and a deep learning module $\mathcal{Q}$.
- **SOTA** (Chen and Qian 2020): The current state-of-the-art model on aspect and opinion terms extraction which implements BERT-large with collaborative learning considering the interactions among aspect terms, opinion terms and sentiment polarities.

Table 5 shows the result for aspect and opinion terms extraction. Since some of the baseline models do not have published code, we only conduct 3 different runs over Rule-distill, DLogic, DLogic* and our proposed model VDLN. For the other baseline models, we use fixed results as reported. The numbers in *italic* form indicate the average results over 3 different runs. This task can be casted as a special case of entity extraction by treating aspect terms and opinion terms as 2 different entity types. Yu, Jiang, and Xia (2019) incorporated explicit relational knowledge among aspect and opinion words through integer linear programming. However, the separation of knowledge reasoning from DNN during learning makes the result suboptimal. As a comparison, VDLN makes these 2 components interactive via variational learning. Compared with Rule-distill (Hu et al. 2016), VDLN outperforms the teacher-student network, demonstrating the advantage of EM algorithm for mutual learning and the ability to learn correlation patterns as logic rules. To verify the expressiveness of our proposed logic network for knowledge reasoning, we compare with explicit rule integration (Wang and Pan 2020), which bridges the DNN outputs with explicit logic rules by minimizing their discrepancies. For fair comparison, we replace their DNN module with ours, denoted by DLogic*. Clearly, VDLN gives better performances at all times, which proves the advantage of automatically learning a logic network over fixed rules. The SOTA model (Chen and Qian 2020) adopted BERT-large as the feature learning backbone and implemented multitask learning framework with collaborative learning mechanism to explore interactions among target terms and sentiment polarities for joint extraction. It is obvious that VDLN with logic reasoning outperforms the SOTA model even with BERT-base neural component. In general, VDLN significantly outperforms all baselines with $p<0.05$ using paired t-test, except the SOTA model on opinion extraction of Res14.

Computational Linguistics Just Accepted MS.
https://doi.org/10.1162/coli_a_00415

Dataset	Model	Entity	Relation
CoNLL04	Gopt (Zhang, Zhang, and Fu 2017)	85.6	67.8
	MtQA (Li et al. 2019)	87.8	68.9
	Rule-distill (Hu et al. 2016)	*88.2**	*71.6**
	DLogic (Wang and Pan 2020)	*87.1**	*64.6**
	DLogic* (Wang and Pan 2020)	*88.3**	*69.9**
	VDLN (ours)	*89.1*	*72.4*
	SOTA (Wang and Lu 2020)	**90.1**	**73.6**
ACE04	MtQA (Li et al. 2019)	83.6	49.4
	Rule-distill (Hu et al. 2016)	*87.7*	*58.1*
	DLogic (Wang and Pan 2020)	*81.6**	*50.2**
	DLogic* (Wang and Pan 2020)	*85.6**	*55.9**
	VDLN (ours)	*87.9*	*57.8*
	SOTA (Wang and Lu 2020)	**88.6**	**59.6**
ACE05	MtQA (Li et al. 2019)	84.8	60.2
	Rule-distill (Hu et al. 2016)	*87.8**	*62.8**
	SpanRel (Dixit and Al-Onaizan 2019)	86.0	62.8
	DLogic (Wang and Pan 2020)	*83.8**	*59.3**
	DLogic* (Wang and Pan 2020)	*87.2**	*62.4**
	VDLN (ours)	*88.5*	*63.7*
	SOTA (Wang and Lu 2020)	**89.5**	**64.3**

Table 6
Results on 3 benchmark datasets for end-to-end relation extraction. *Italic* numbers on CoNLL04 and ACE04 indicate average results over 3 random splits and 5 random splits, respectively, and those on ACE05 are averaged over 3 different runs. * indicates the results are significant with $p < 0.05$.

End-to-End Relation Extraction: Besides the aforementioned baseline DLogic and DLogic*, we further adopt the following baselines:

- **Gopt**: A globally optimized neural model for end-to-end relation extraction (Zhang, Zhang, and Fu 2017). The work convert the entity and relation extraction problem into a single table filling task, which produces a score for each label in the next step given the state of a partially-filled table. Moreover, global optimization is used which treat the entire sentence as a unit.
- **MtQA**: The extraction of entities and relations is casted as the task of identifying answer spans from the context given some question templates (Li et al. 2019). The question encodes relevant information corresponding to the target entity or relation to be identified.
- **SpanRel**: An end-to-end deep learning model based on span-level predictions (Dixit and Al-Onaizan 2019). Instead of token-level modeling, span-based models take the features corresponding to all possible spans within a sentence for both entity and relation predictions.
- **SOTA** (Wang and Lu 2020): A joint model using two different encoders, namely a table encoder and a sequence encoder to intensively exploit the target interactions, together with rich encodings combining word vectors, character vectors and strong pretrained contextualized vectors.

Table 6 lists the performances of the proposed models and baseline models for each end-to-end relation extraction dataset. Note that although (Luan et al. 2019) also showed

Computational Linguistics Just Accepted MS.
https://doi.org/10.1162/coli_a_00415

Model	Entity Extraction	Event trigger Identification	Event trigger Classification	Event argument Identification	Event argument Classification
JEventEntity	81.8	71.0	68.8	50.6	48.4
dbRNN	-	-	69.6	57.2	50.1
GAIL	87.1	73.9	72.0	55.1	52.4
Joint3EE	-	72.5	69.8	**59.9**	52.1
DYGIE++	89.7	-	69.7	53.0	48.8
VDLN	87.7	75.6	73.2	56.1	52.7
SOTA (ONEIE)	**90.2**	**78.2**	**74.7**	59.2	**56.8**

Table 7
Comparisons with SOTA models on ACE05 for end-to-end event extraction.

promising results on ACE04 and ACE05 datasets, it depends on auxiliary coreference supervisions, which is not fair to be compared with. Nevertheless, we still achieve comparable performances. MtQA (Li et al. 2019) treats the task as a question answering problem with pre-defined question templates and uses BERT as a backbone. Compared with Gopt (Zhang, Zhang, and Fu 2017) without self-attention, the improvement shows the advantage of modeling token-level dependencies for information extraction. The results also verify our consistent improvement over Rule-distill (Hu et al. 2016). Compared with the SOTA model (Wang and Lu 2020) which adopted rich encodings combining word vectors (Glove), character embeddings and contextualized embeddings (ALBERT-large which is an extensively pretrained large model), our model produces slightly lower performances. We conjecture that the high result of the SOTA model depend on its rich encodings, as when replacing ALBERT-large with BERT in their model, the F1 score for entity extraction on ACE05 drops to 87.8, according to (Wang and Lu 2020). In general, VDLN significantly outperforms the other baselines except SOTA, and Rule-distill on entity extraction of ACE04 with $p < 0.05$.

End-to-End Event Extraction: For this task, the state-of-the-art models to be compared are listed in the following

- **JEventEntity**: A probabilistic model taking into consideration of intensive dependencies between event triggers and entity mentions, as well as relationships among events (Yang and Mitchell 2016). A joint inference procedure is then proposed to globally optimize all the predictions within a text input.
- **dbRNN**: A novel dependency-bridged recurrent neural network for event extraction (Sha et al. 2018), which fully utilises both the sequential and syntactic structure of a sentence to enhance the extraction performance.
- **GAIL**: A deep learning model based on generative adversarial imitation learning (Zhang, Ji, and Sil 2019). The authors use reinforcement learning to model sequential predictions and aims to produce proper reward values estimated from discriminators in GAN.
- **Joint3EE**: A joint deep learning model to simultaneously achieve end-to-end event extraction (Nguyen and Nguyen 2019) by decomposing the joint probability into a product of the probability of each target variable conditioned on the processed units.

Computational Linguistics Just Accepted MS.
https://doi.org/10.1162/coli_a_00415

Model	Res14		Lap14		CoNLL04		ACE04		ACE05	
	Aspect	Opinion	Aspect	Opinion	Entity	Relation	Entity	Relation	Entity	Relation
BERT	86.2	86.1	80.2	79.5	87.6	69.3	85.8	55.5	87.4	61.3
VDLN (BERT)	87.5	**87.1**	**82.7**	81.3	89.1	72.4	87.9	57.8	**88.3**	63.8
BERT-large	87.9	86.5	80.7	81.1	88.1	71.0	88.1	**59.8**	87.8	63.2
VDLN (BERT-large)	**88.4**	**87.1**	81.6	**81.8**	88.6	72.6	**88.2**	59.4	87.6	**64.6**
SpanBERT	87.1	86.3	79.4	77.6	87.1	67.6	85.9	54.9	85.7	59.6
VDLN (SpanBERT)	88.2	86.4	81.2	78.8	87.2	70.3	86.2	55.1	86.2	61.6
Roberta	86.6	86.2	79.3	78.4	89.3	72.1	85.1	55.8	86.2	61.0
VDLN (Roberta)	86.9	85.7	80.3	79.1	**90.1**	**73.4**	85.3	56.5	86.4	61.5
transformer	84.3	84.2	76.2	77.3	85.8	62.7	82.1	51.4	83.4	59.1
VDLN (transformer)	85.2	85.7	76.8	78.3	86.5	63.3	82.7	53.1	83.9	58.8

Table 8
Comparison using different pretrained models for the neural component.

Model	Res14		Lap14		CoNLL04		ACE04		ACE05	
	Aspect	Opinion	Aspect	Opinion	Entity	Relation	Entity	Relation	Entity	Relation
$\mathcal{Q}$	86.2	86.1	80.2	79.5	87.6	69.3	85.8	55.5	87.4	61.3
$\mathcal{P}$	86.0	86.5	81.1	79.9	88.0	70.4	86.0	55.7	87.5	60.9
$\mathcal{Q}^*$	87.3	**87.1**	82.5	81.2	88.9	**72.5**	**87.9**	**57.8**	87.8	**63.8**
$\mathcal{P}^*$	**87.6**	87.0	**82.7**	81.2	**89.1**	71.9	87.3	57.6	88.1	63.5
$\mathcal{P} + \mathcal{Q}$	87.5	**87.1**	**82.7**	**81.3**	**89.1**	72.4	**87.9**	**57.8**	**88.3**	**63.8**

Table 9
Performances for each separate module and its variations on the task of aspect/opinion extraction and end-to-end relation extraction.

- **DYGIE++**: A joint model for end-to-end event extraction based on contextualized span representations (Wadden et al. 2019). The span representations encode local and global interactions with a dynamic graph update to propagate long-range information.
- **SOTA (ONEIE)**: A joint neural model consisting of contextualized text representations and manually-designed global features to capture the cross-task and cross-instance interactions (Lin et al. 2020).

The results on end-to-end event extraction is listed in Table 7. JEventEntity (Yang and Mitchell 2016) adopts joint inference with extensive manually-designed liguistic features. Its performance is inferior compared to deep learning models. On the other hand, DNNs alone lack explicit knowledge that is crucial for the task at hand. Hence, dbRNN (Sha et al. 2018) and Joint3EE (Nguyen and Nguyen 2019) incorporate linguistically-informed features, e.g., dependency relations, to enhance the performance of DNNs. From the comparison results, VDLN achieves the best performances among the baselines except the last row without requiring any external linguistic resources and only needs to generate simple rule templates that associate extracted entities and events with the argument relation predictions in an automatic manner. SOTA (ONEIE) (Lin et al. 2020) produces the best result mainly originated from the manually-designed global features to enforce cross-task and cross-instance relationships, e.g., *A TRANSPORT event has only one DESTINATION argument*.

Computational Linguistics Just Accepted MS.
https://doi.org/10.1162/coli_a_00415

Model	Entity Extraction	Event trigger Identification	Event trigger Classification	Event argument Identification	Event argument Classification
$\mathcal{Q}$	86.5	75.0	72.7	54.7	51.2
$\mathcal{P}$	86.9	74.7	72.7	55.3	51.5
$\mathcal{Q}$*	87.5	75.2	**73.2**	55.7	52.6
$\mathcal{P}$*	**87.8**	75.3	72.8	56.0	52.3
$\mathcal{P}+\mathcal{Q}$	87.7	**75.6**	**73.2**	**56.1**	**52.7**

Table 10
Performances for each separate module and its variations on end-to-end event extraction.

7.4 Analysis

Our default experimental setting uses BERT as the neural component. To demonstrate the generality of our proposed VDLN architecture, we conduct extra experiments by replacing BERT with three other strong contextualized neural models, namely SpanBERT, Roberta and BERT-large, respectively with fine-tuning, as well as a non-pretrained model using pure transformers. The results are listed in Table 8. We denote by "VDLN (*)" as the proposed joint model with the neural component $\mathcal{Q}$ replaced by $* =$ BERT-large, SpanBERT, Roberta, transformer. All the experiments with pretrained models involve fine-tuning of the pretrained parameters. The transformer model follows the one in (Wang and Pan 2020). From Table 8, we observe that large models (BERT-large) usually produce the best results, whereas VDLN (Roberta) performs best on CoNLL04. SpanBERT has inferior performances on average. Clearly, the joint model VDLN outperforms its neural component alone across almost all experiments. With such observation, we show that the proposed methodology is able to benefit a wide variety of its neural counterpart.

To analyze the effect of each module within the proposed framework and the effect of the EM training procedure, we conduct experiments on each separate module as well as some variations within a module. The results are shown in Table 9 and Table 10 for all the three different tasks. Specifically, $\mathcal{Q}$ and $\mathcal{P}$ record the performance of each individual module alone, respectively, without the EM training alternation. Since $\mathcal{P}$ requires the output from the deep learning predictions as its input for logic reasoning, we initialize $\mathcal{P}$ with the output from a pretrained module $\mathcal{Q}$. $\mathcal{Q}$* and $\mathcal{P}$* record the performance using $\mathcal{Q}$ and $\mathcal{P}$, respectively, for final predictions after jointly training both modules alternatively via variational EM. It could be observed that for separate models, $\mathcal{P}$ is slightly better than $\mathcal{Q}$ because it inherits the result from $\mathcal{Q}$ and further conducts logic reasoning based on the intensive interactions among the output variables. However, both of them are inferior than those with variational learning paradigm, which proves that the EM algorithm encourages a mutual enhancement between two different modules. For final predictions, the results from $\mathcal{P}$* and $\mathcal{Q}$* are comparable most of the times. In the end, we use the ensemble model $\mathcal{P}+\mathcal{Q}$ which produces the best result on Laptop-14, ACE04 and ACE05.

We further verify the effect of each component within the framework via ablation studies. As shown in Table 11 and 12, the first column indicates different model variations. DNN is the deep learning component we adopt, which corresponds to module $\mathcal{Q}$ in VDLN. DNN w/o BiGRU removes the BiGRU layer on top of the BERT model and DNN+CRF further connects a linear-chain CRF with structured loss as the last layer. Clearly, the performances with and without BiGRU are similar. However, BiGRU brings

Computational Linguistics Just Accepted MS.
https://doi.org/10.1162/coli_a_00415

Model	Res14		Lap14		CoNLL04		ACE04		ACE05	
	Aspect	Opinion	Aspect	Opinion	Entity	Relation	Entity	Relation	Entity	Relation
DNN w/o BiGRU	86.5	85.8	79.0	79.2	87.2	70.8	86.2	56.7	86.9	61.5
DNN	86.2	86.1	80.2	79.5	87.6	69.3	85.8	55.5	87.4	61.3
DNN+CRF	87.2	86.2	80.7	80.3	88.3	70.8	87.4	57.2	87.6	62.4
VDLN (seg)	87.1	87.0	82.0	**82.5**	88.8	70.7	87.6	57.6	88.0	62.8
VDLN (rel)	87.4	87.2	80.6	81.7	88.4	71.9	86.9	58.2	87.7	63.5
VDLN w/o BiGRU	87.1	86.6	81.9	80.7	88.3	71.9	86.1	54.2	87.8	61.5
VDLN	87.5	87.1	**82.7**	81.3	**89.1**	72.4	**87.9**	57.8	**88.3**	**63.8**
VDLN (softmax)	86.7	87.0	81.5	81.2	88.5	71.4	87.5	57.9	87.7	62.3
VDLN+CRF (rel)	87.2	86.9	81.9	81.4	88.2	72.1	87.6	58.4	87.8	63.7
VDLN+CRF	**87.8**	**87.3**	82.3	82.2	88.5	**72.7**	**87.9**	58.6	88.0	63.5

Table 11
Ablation study for each component of the proposed model on aspect/opinion extraction and end-to-end relation extraction.

Model	Entity Extraction	Event trigger Identification	Event trigger Classification	Event argument Identification	Event argument Classification
DNN w/o BiGRU	86.1	75.4	72.7	54.4	51.0
DNN	86.5	75.0	72.7	54.7	51.2
DNN+CRF	87.2	74.6	72.8	54.3	50.9
VDLN (seg)	87.5	74.9	72.7	54.9	51.7
VDLN (rel)	86.8	74.7	71.8	55.2	52.0
VDLN w/o BiGRU	86.7	74.9	72.5	54.8	51.8
VDLN	**87.7**	75.6	**73.2**	**56.1**	**52.7**
VDLN+CRF (rel)	87.1	**75.8**	72.8	55.7	52.5
VDLN+CRF	87.4	75.4	72.8	55.4	52.2

Table 12
Ablation study for each component of the proposed model on end-to-end event extraction.

some performance gain when associated with the joint model VDLN, compared with VDLN w/o BiGRU which removes BiGRU in the joint model. A CRF layer is able to bring some performance gain compared to DNN alone for both aspect/opinion extraction and end-to-end relation extraction. However, it is not beneficial for event trigger and event argument prediction, most probably due to the fact that most event triggers are made from a single word. Another two variations, namely VDLN (seg) and VDLN (rel) refer to the model with only segmentation-based rule templates and relation-based rule templates, respectively, within module $\mathcal{P}$. Specifically, segmentation-based rule templates only associate token-level interactions, e.g., $q(\mathbf{y}_{i-1}), q(\mathbf{y}_{i+1}) \rightarrow p(\mathbf{y}_i)$. On the other hand, relation-based rule templates associate relational triplets with token predictions, e.g., $q(\mathbf{y}_j), q(\mathbf{y}_{r_{ij}}) \rightarrow p(\mathbf{y}_i)$. The results for these two variations demonstrate the contribution of each kind of interaction for the proposed model. From Table 11, we could observe that VDLN (seg) is beneficial for entity predictions, whereas VDLN (rel) mostly works for relation extraction. The last row VDLN+CRF takes DNN+CRF model as module $\mathcal{Q}$ in the joint model VDLN. It has similar performances compared to VDLN which shows VDLN already learns structured information in a CRF model. VDLN+CRF (rel) only adopts relation-based rule templates in module $\mathcal{P}$. By comparing it with VDLN, we could verify that the segmentation rule used in module $\mathcal{P}$ is more beneficial than using a simple graphical model. We also verify the effect of using sparsemax for the rule learning process. The sparsemax operator explicitly constrains the number of atoms to be selected to form the body of a rule. By replacing it with softmax (VDLN

Computational Linguistics Just Accepted MS.
https://doi.org/10.1162/coli_a_00415

Model		GNN+$\mathcal{Q}$	GCN+$\mathcal{Q}$	VDLN	VDLN+rules
Res14	Aspect	86.2	86.8	87.5	**88.0**
	Opinion	86.5	86.4	87.1	**87.3**
Lap14	Aspect	81.3	81.8	82.7	**82.9**
	Opinion	81.0	81.3	81.3	**82.1**
CoNLL04	Entity	87.9	88.2	**89.1**	**89.1**
	Relation	71.3	71.4	72.4	**72.7**
ACE04	Entity	85.7	86.7	**87.9**	87.7
	Relation	56.2	56.7	57.8	**58.3**
ACE05	Entity	86.1	86.3	**88.3**	88.0
	Relation	62.3	62.6	63.8	**64.1**
ACE05	Entity	87.1	**87.9**	87.7	87.5
	Trigger (I)	73.9	75.1	**75.6**	**75.6**
	Trigger (C)	71.8	72.3	**73.2**	72.7
	Argument (I)	53.6	54.4	**56.1**	55.8
	Argument (C)	51.7	52.0	**52.7**	52.5

Table 13
Investigations of the logic network.

(softmax)), the results show that sparsemax provides better result and is semantically more meaningful.

To investigate the advantage of the logic-inspired network within module $\mathcal{P}$, we compare the proposed model with another popular and effective deep learning model, i.e., graph neural networks (GNNs) (Dai, Dai, and Song 2016) and graph convolutional networks (GCNs) (Kipf and Welling 2017) for information propagation. The results are shown in Table 13. Specifically, we replace the logic network in $\mathcal{P}$ with a GNN (or GCN), which takes the context $\mathrm{ctx}(m)$ of each target node m as the neighboring nodes to update its own feature via non-linear transformations (spectral-based graph convolutions). In a word, the graph structure of the GNN (GCN) is provided by the rule templates used in the logic network where two nodes are connected if they appear in the same rule. We denote this model by GNN+$\mathcal{Q}$ (GCN+$\mathcal{Q}$). GCN+$\mathcal{Q}$ is more expressive than GNN+$\mathcal{Q}$. Clearly, GCN+$\mathcal{Q}$ outperforms GNN+$\mathcal{Q}$ in general, but is still inferior than our proposed model across all except one experiment, indicating that the proposed logic module has better reasoning capabilities than graph-based models in this problem domain.

As mentioned in Section 5.2, the logic network is able to learn relevant knowledge automatically, as well as encode prior knowledge if provided. In all the previous experiments, we do not feed any manually-designed logic rules into the logic network for fair comparisons and a demonstration of our model's generality. To investigate how the given rules contribute to the actual task, we design some easily-acquired logic rules for each task and incorporate them into the learning of LNet. The manually-designed rules for each specific task and dataset are listed in Table 14. For aspect and opinion terms extraction, we design rules involving dependency relations and POS tags, as adopted in (Qiu et al. 2011; Yu, Jiang, and Xia 2019). For example, the FOL rule "$\mathrm{aspect}(x) \wedge \mathrm{pos}_{noun}(x) \wedge \mathrm{dep}_{amod}(y, x) \wedge \mathrm{pos}_{adj}(y) \Rightarrow \mathrm{opinion}(y)$" states that if x is an aspect word having POS tag *noun*, we can infer that y with POS tag *adj* is an opinion word when there is a dependency relation *amod* between x and y. The dependency structures and POS tags are generated using Stanford CoreNLP (Manning et al. 2014). For end-to-end relation extraction, we mainly adopt relational rules demonstrating the correlations between entity types and relation types. Lastly, for event extraction, we use event trigger types and entity types as preconditions to design the FOL rules in order

Computational Linguistics Just Accepted MS.
https://doi.org/10.1162/coli_a_00415

Task		FOL Formula
Aspect and Opinion Extraction		$\text{aspect}(x) \wedge \text{pos}_{noun}(x) \wedge \text{dep}_{nn}(x,y) \wedge \text{pos}_{noun}(y) \Rightarrow \text{aspect}(y)$ $\text{aspect}(x) \wedge \text{pos}_{noun}(x) \wedge \text{dep}_{conj}(x,y) \wedge \text{pos}_{noun}(y) \Rightarrow \text{aspect}(y)$ $\text{opinion}(x) \wedge \text{pos}_{adj}(x) \wedge \text{dep}_{conj}(x,y) \wedge \text{pos}_{adj}(y) \Rightarrow \text{opinion}(y)$ $\text{aspect}(x) \wedge \text{pos}_{noun}(x) \wedge \text{dep}_{nsubj}(x,y) \wedge \text{pos}_{adj}(y) \Rightarrow \text{opinion}(y)$ $\text{opinion}(x) \wedge \text{pos}_{adj}(x) \wedge \text{dep}_{nsubj}(y,x) \wedge \text{pos}_{noun}(y) \Rightarrow \text{aspect}(y)$ $\text{aspect}(x) \wedge \text{pos}_{noun}(x) \wedge \text{dep}_{amod}(y,x) \wedge \text{pos}_{adj}(y) \Rightarrow \text{opinion}(y)$ $\text{opinion}(x) \wedge \text{pos}_{adj}(x) \wedge \text{dep}_{amod}(x,y) \wedge \text{pos}_{noun}(y) \Rightarrow \text{aspect}(y)$
End-to-End Relation Extraction	CoNLL04	$\text{person}(x) \wedge \text{live\_in}(r_{(x,y)}) \Rightarrow \text{location}(y)$ $\text{location}(x) \wedge \text{live\_in}(r_{(y,x)}) \Rightarrow \text{person}(y)$ $\text{organization}(x) \wedge \text{org\_based\_in}(r_{(x,y)}) \Rightarrow \text{location}(y)$ $\text{location}(x) \wedge \text{org\_based\_in}(r_{(y,x)}) \Rightarrow \text{organization}(y)$ $\text{location}(x) \wedge \text{located\_in}(r_{(x,y)}) \Rightarrow \text{location}(y)$ $\text{person}(x) \wedge \text{kill}(r_{(x,y)}) \Rightarrow \text{person}(y)$ $\text{person}(x) \wedge \text{work\_for}(r_{(x,y)}) \Rightarrow \text{organization}(y)$ $\text{organization}(x) \wedge \text{work\_for}(r_{(y,x)}) \Rightarrow \text{person}(y)$
	ACE04	$\text{person}(x) \wedge \text{person} - \text{social}(r_{(x,y)}) \Rightarrow \text{person}(y)$ $\text{person}(x) \wedge \text{discourse}(r_{(x,y)}) \Rightarrow \text{person}(y)$ $\text{geographical}(x) \wedge \text{discourse}(r_{(x,y)}) \Rightarrow \text{geographical}(y)$ $\text{organization}(x) \wedge \text{discourse}(r_{(x,y)}) \Rightarrow \text{organization}(y)$ $\text{person}(x) \wedge \text{employment}(r_{(x,y)}) \Rightarrow \text{organization}(y) \vee \text{geographical}(y)$ $\text{geographical}(x) \wedge \text{employment}(r_{(y,x)}) \Rightarrow \text{organization}(y) \vee \text{person}(y)$ $\text{organization}(x) \wedge \text{GPE} - \text{affiliation}(r_{(x,y)}) \Rightarrow \text{geographical}(y)$ $\text{person}(x) \wedge \text{GPE} - \text{affiliation}(r_{(x,y)}) \Rightarrow \text{geographical}(y)$
	ACE05	$\text{person}(x) \wedge \text{person} - \text{social}(r_{(x,y)}) \Rightarrow \text{person}(y)$ $\text{vehicle}(x) \wedge \text{part} - \text{whole}(r_{(x,y)}) \Rightarrow \text{vehicle}(y)$ $\text{geographical}(x) \wedge \text{part} - \text{whole}(r_{(x,y)}) \Rightarrow \text{geographical}(y)$ $\text{organization}(x) \wedge \text{part} - \text{whole}(r_{(x,y)}) \Rightarrow \text{organization}(y)$ $\text{person}(x) \wedge \text{ORG} - \text{affilliation}(r_{(x,y)}) \Rightarrow \text{organization}(y) \vee \text{geographical}(y)$ $\text{organization}(x) \vee \text{geographical}(x) \wedge \text{ORG} - \text{affilliation}(r_{(y,x)}) \Rightarrow \text{person}(y)$ $\text{organization}(x) \wedge \text{GPE} - \text{affilliation}(r_{(x,y)}) \Rightarrow \text{location}(y)$ $\text{location}(x) \wedge \text{GPE} - \text{affilliation}(r_{(y,x)}) \Rightarrow \text{organization}(y)$
End-to-End Event Extraction		$\text{Movement\_Transport}(x) \wedge \text{person}(y) \Rightarrow \text{Destination}(r_{(x,y)})$ $(\text{Personnel\_Elect}(x) \vee \text{Personnel\_StartPosition}(x) \vee \text{Personnel\_EndPosition}(x)$ $\vee \text{Life\_Marry}(x) \vee \text{Justice\_Arrest} - \text{Jail}(x)) \wedge \text{person}(y) \Rightarrow \text{Position}(r_{(x,y)})$ $(\text{Personnel\_StartPosition}(x) \vee \text{Personnel\_EndPosition}(x))$ $\wedge \text{organization}(y) \Rightarrow \text{Attacker}(r_{(x,y)})$ $(\text{Contact\_Meet}(x) \vee \text{Contact\_PhoneWrite}(x) \vee \text{Conflict\_Demonstrate}(x))$ $\wedge \text{person}(y) \Rightarrow \text{Attacker}(r_{(x,y)})$ $(\text{Movement\_Transport}(x) \vee \text{Contact\_Meet}(x) \vee \text{Conflict\_Attack}(x) \vee \text{Life\_Die}(x))$ $\wedge \text{time}(y) \Rightarrow \text{Target}(r_{(x,y)})$ $(\text{Justice\_Sentence}(x) \vee \text{Justice\_ChargeIndict}(x) \vee \text{Justice\_Convict}(x))$ $\wedge \text{person}(y) \Rightarrow \text{Adjudicator}(r_{(x,y)})$ $\text{Justice\_Sentence}(x) \wedge \text{sentence}(y) \Rightarrow \text{Crime}(r_{(x,y)})$ $\text{Justice\_ChargeIndict}(x) \wedge \text{crime}(y) \Rightarrow \text{Prosecutor}(r_{(x,y)})$

Table 14
Pre-defined logic rules for each task and dataset.

to entail the target relations. The results are shown in the last column of Table 13, i.e., "VDLN+rules". It could be observed that the performances are improved on the task of aspect and opinion terms extraction. For end-to-end relation extraction, additional rules are more beneficial for relation extraction. However, we could observe a degradation in the performance of event extraction when inserting the manually-designed logic rules. This might be caused by inaccurate event trigger predictions as well as uncertain rules with sparse coverage in the given corpus.

We would like to emphasize that compared with the pre-defined logic rules, the learned logic rules are different in the way that the atoms in the rule bodies are rather abstract and composited. More concretely, if we define the set of all generated synthetic

Computational Linguistics Just Accepted MS.
https://doi.org/10.1162/coli_a_00415

DNN	VDLN
"The **ambience** is also more laid-back and *relaxed*."	"The **ambience** is also more *laid-back* and *relaxed*."
"The folding **chair** I was seated at was *uncomfortable*."	"The **folding chair** I was seated at was *uncomfortable*."
"It is *robust*, with a *friendly* use as all Apple products."	"It is *robust*, with a *friendly* **use** as all Apple products."
"... on duty with the **6th Fleet** in the **Mediterranean**,..." Entity: location, location; Relation: located_in (1,2)	"... on duty with the **6th Fleet** in the **Mediterranean**,..." Entity: organization, location; Relation: org_based_in(1,2)
"... get them all home, said **Ms. Say** in **Nashville**, **Tenn**." Entity: people, location, location; Relation: located_in(1,2)	"... get them all home, said **Ms. Say** in **Nashville**, **Tenn**." Entity: people, location, location; Relation: located_in(2,3)
"... the **legislature** of the **state of Florida** ..." Entity: facility, geographical; Relation: None	"... the **legislature** of the **state of Florida** ..." Entity: organization, geographical; Relation: emp-org(1,2)

Table 15
Example outputs by DNN alone and VDLN, respectively.

atoms in the atom layer as $D_{\text{ctx}(m)} = \{d_1, ..., d_{80}\}$[3], we are able to list a few learned logic rules for relations on CoNLL04.

- $d_{11} \wedge d_2 \wedge d_{58} \wedge d_{31} \wedge d_7 \Rightarrow (r = \text{located\_in})$.
- $d_{18} \wedge d_{34} \wedge d_{16} \wedge d_4 \wedge d_{10} \Rightarrow (r = \text{work\_for})$.
- $d_5 \wedge d_4 \wedge d_{31} \Rightarrow (r = \text{live\_in})$.

Here each atom d_n is a linear combination of all different classes for a word or a relation. As shown in (14) and (15), each atom d_n is either a 1-ary atom corresponding to a linear combination of the DNN predictions of an argument word, or a 2-ary atom corresponding to a bi-linear property of two arguments. For predictions of a relation $r_{(\epsilon_1,\epsilon_2)}$ in the logic network, the input is $\tilde{\mathbf{y}}_{\text{ctx}}(r_{(\epsilon_1,\epsilon_2)}) = (\mathbf{q}_{r_{(\epsilon_1,\epsilon_2)}}, \mathbf{q}_{\epsilon_1}, \mathbf{q}_{\epsilon_2})$. Then $\{d_1, ..., d_{20}\}$ are 1-ary atoms corresponding to different linear combinations of $\mathbf{q}_{r_{(\epsilon_1,\epsilon_2)}}$. $\{d_{21}, ..., d_{40}\}$ are 1-ary atoms corresponding to different linear combinations of $\mathbf{q}_{\epsilon_1}$ which is the entity class distribution of the head entity ϵ_1 from DNNs. Similarly, $\{d_{41}, ..., d_{60}\}$ are 1-ary atoms corresponding to linear combinations of $\mathbf{q}_{\epsilon_2}$ for tail entity ϵ_2. The last 20 atoms $\{d_{61}, ..., d_{80}\}$ are 2-ary atoms corresponding to bilinear interactions of $\mathbf{q}_{\epsilon_1}$ and $\mathbf{q}_{\epsilon_2}$. To interpret the example rule "$d_{11} \wedge d_2 \wedge d_{58} \wedge d_{31} \wedge d_7 \Rightarrow (r = \text{located\_in})$" for CoNLL04 dataset, suppose the pair of entities being queried for the relation is (*the White House*, *U.S.*), we will have d_{11}, d_2 and d_7 representing linear transformations of $\mathbf{q}_{r_{(\text{the White House,U.S.})}}$, d_{58} representing a linear transformation of $\mathbf{q}_{\text{U.S.}}$, and d_{31} representing a linear transformation of $\mathbf{q}_{\text{the White House}}$. To be more specific, when those learned linear transformation weights favor a particular entity/relation class, a more concrete interpretation of the above rule could be

$$\text{located\_in}(r_{(\text{the White House,U.S.})}) \wedge \text{live\_in}(r_{(\text{the White House,U.S.})}) \wedge \text{location}(\text{U.S.})$$
$$\wedge\text{organization}(\text{the White House}) \wedge \text{located\_in}(r_{(\text{the White House,U.S.})}) \Rightarrow (r = \text{located\_in}).$$

Note that these learned rules are all generic rules learned for each specific dataset, because the linear and bi-linear transformations to compose those atoms are identical across each training instance and are learned throughout the training process.

3 According to the experimental setting, we have 80 generated atoms

Computational Linguistics Just Accepted MS.
https://doi.org/10.1162/coli_a_00415

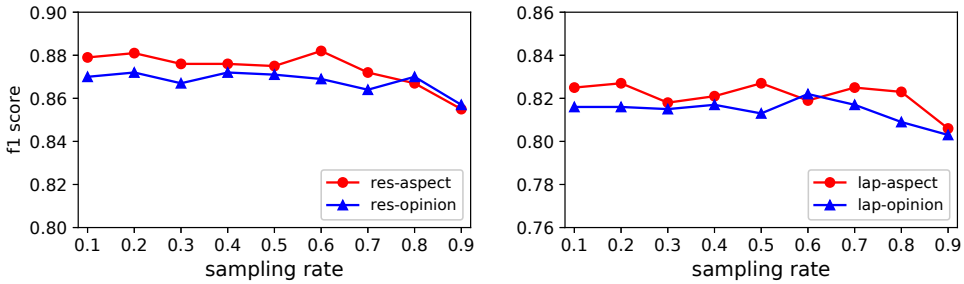

Figure 4
F1 with different sampling rates on the task of aspect and opinion extraction for Restaurant14 data (left) and Laptop14 data (right).

For qualitative analysis, we use Table 15 to list a few examples showing that the incorporation of logic reasoning is able to more correctly extract target terms/relations compared to pure neural networks $\mathcal{Q}$. Specifically, the words in **bold** indicate aspects or entities and the words in *italic* form indicate opinions. For entity and relation extraction, the second row in each example represents the predicted entity types and relations. The numbers in the relation indicate the indices of its corresponding entities. For aspect and opinion terms extraction, VDLN is able to identify target aspects or opinions with certain syntactic relations which are missed by pure DNN. For example, the opinion term *laid-back* can be extracted by associating it with the aspect term **ambience** and another opinion term *relaxed*. For entity and relation extraction, VDLN modifies incorrect predictions from DNN. For example, the output relation located_in(6th Fleet, Mediterranean) is corrected as org_based_in(6th Fleet, Mediterranean) by VDLN.

To demonstrate the model's robustness, we conduct experiments with varying hyper-parameters. We choose three parameters, namely the sampling rate ρ during the EM updates, the number of atoms T in the rule body for each rule formed in the logic network, and the number of rules in the rule set $\{R_1, ..., R_S\}$ that share the same head atom h. Specifically, we use different sampling rates ranging from $\rho = 0.1$ to $\rho = 0.9$ when updating both $\mathcal{P}$ and $\mathcal{Q}$. Here ρ is the probability of using the predictions from $\mathcal{P}$ or $\mathcal{Q}$ when learning the parameters of $\mathcal{Q}$ or $\mathcal{P}$, respectively during the variational EM updates. With probability $1 - \rho$, the ground-truth label is used to supervise each module. The results for aspect and opinion terms extraction are shown in Figure 4 and the results for CoNLL04 and ACE05 dataset are shown in Figure 5. Both two figures demonstrate the robustness of VDLN against different sampling rates. The performance drop for $\rho = 0.9$ is reasonable as only 10% of the ground-truth labels are used for supervision during the EM training procedure.

Figure 6, 7 and 8 correspond to the F1 scores for entity extraction and relation prediction on ACE05, ACE04 and CoNLL04 dataset, respectively. The x-axis on the left subfigure indicates the number of atoms T in the body of each rule, i.e., $d_{j_1} \wedge, ..., \wedge d_{j_T} \Rightarrow h$. The x-axis on the right subfigure indicates the number of rules S for each head atom h. As indicated in the figures, the final performance of our proposed framework is not sensitive to such hyper-parameters within the logic network. For ACE05 and ACE04, varying T from 1 to 10 results in more stable preformance for entity extraction compared with relation prediction. On the other hand, the model's performance is relatively less dependent on the number of rules S. When changing S from 10 to 50, the results stay within a small range. As for CoNLL04, the performance decreases when S is higher than

Computational Linguistics Just Accepted MS.
https://doi.org/10.1162/coli_a_00415

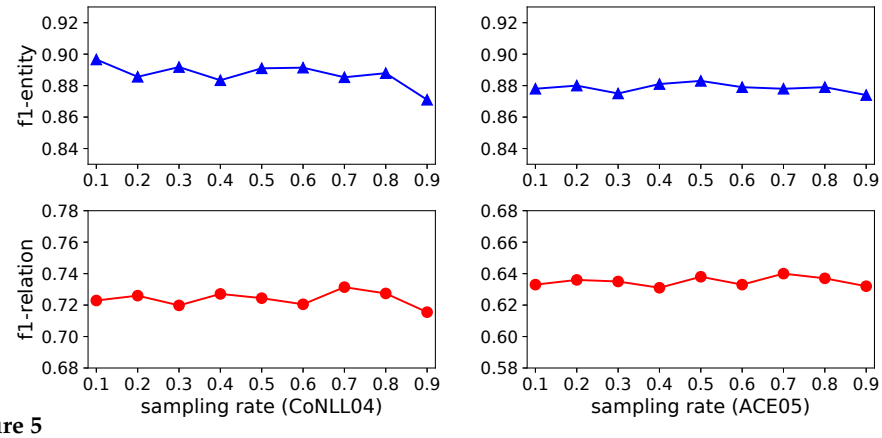

Figure 5
F1 with different sampling rates on the task of end-to-end relation extraction.

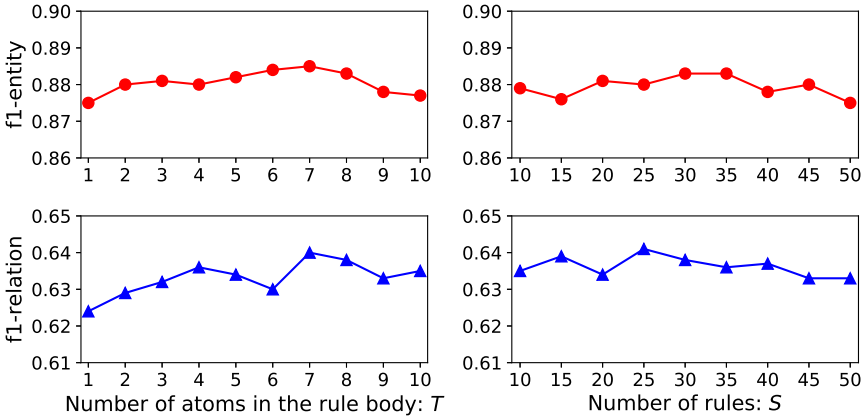

Figure 6
Sensitivity study for the logic network on ACE05 dataset.

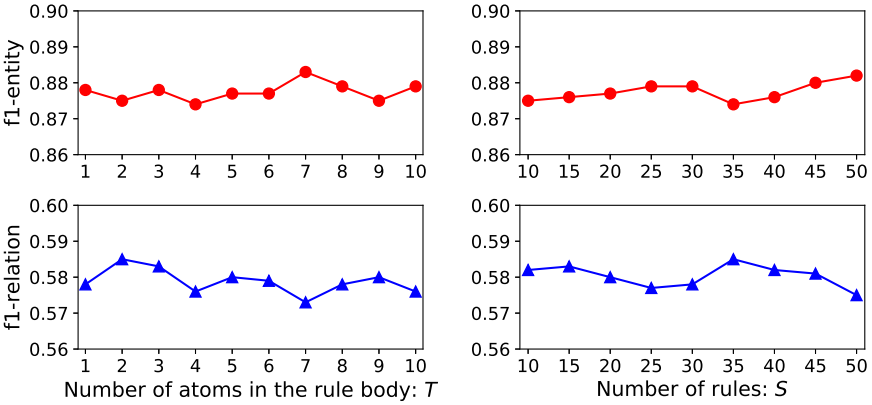

Figure 7
Sensitivity study for the logic network on ACE04 dataset.

Computational Linguistics Just Accepted MS.
https://doi.org/10.1162/coli_a_00415

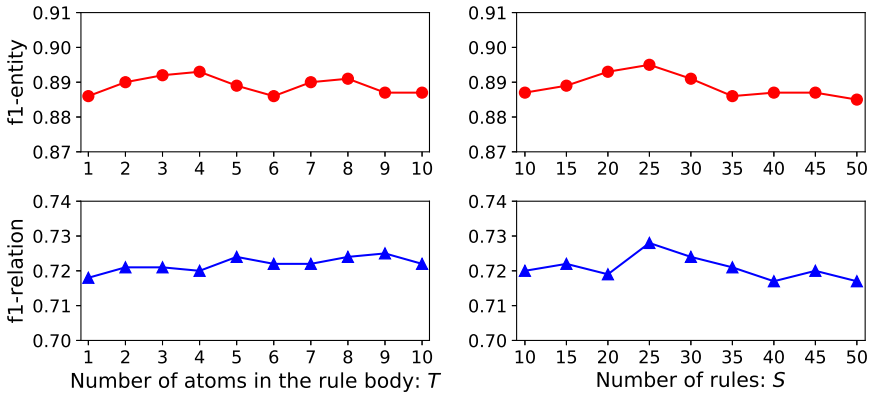

Figure 8
Sensitivity study for the logic network on CoNLL04 dataset.

25. This might result from the fact that the interactions between entities and relations are simpler in the smaller CoNLL04 dataset which becomes easy to be overfitted.

7.5 Error Analysis & Future Work

For error analysis, our model has limitations when the entities (triggers) are not correctly extracted. Specifically, if the entities (triggers) are not extracted in the entity (trigger) prediction phase in module $\mathcal{Q}$, i.e., generating predictions from $\mathbf{q}_i$ for each word, it becomes hard to rectify such predictions in the logic networks and during the EM training procedure. The reason is that the relations and rules are all based on the extracted candidate entities from $\mathcal{Q}$. Indeed, when some entities are not identified in $\mathcal{Q}$, there will be no bilinear interactions between the missed entities and other entities to be modeled in the logic network. Hence, it is difficult for VDLN to learn useful rules to correct its predictions.

In our future work, we plan to solve the above limitations by revising the extraction mechanism in the neural component where entities are first predicted followed by relation predictions. A table filling mechanism might be a good choice (Miwa and Sasaki 2014). We also plan to design more interpretable networks in terms of logic reasoning so that the learned rules can be explicitly explained. In terms of application, our future work may include generalizing our proposed framework to work with more challenging cases, e.g., cross-sentence correlations, cross-instance consistencies, in order to be applied on other application domains, e.g., document-level event extraction, cross-sentence relation extraction, etc.

8. Conclusion

We propose a variational deep logic network to inherit both the representational power of deep learning and the reasoning capabilities of logic systems for joint inference in IE. These two paradigms communicate through the variational EM algorithm. For knowledge reasoning, we introduce a novel logic network that transforms logic semantics to a deep hierarchical architecture to facilitate logic inference automatically. Meanwhile, the logic network enhances the expressiveness over manually-designed rules by learning

Computational Linguistics Just Accepted MS.
https://doi.org/10.1162/coli_a_00415

more effective atom combinations according to the training data. It is also flexible to incorporate pre-defined logic rules to further enhance the final performance.

Acknowledgement

This work is supported by NTU Nanyang Assistant Professorship (NAP) grant M4081532.020, 2020 Microsoft Research Asia collaborative research grant, and Singapore Lee Kuan Yew Postdoctoral Fellowship.

Computational Linguistics Just Accepted MS.
https://doi.org/10.1162/coli_a_00415

References

Adel, Heike and Hinrich Schütze. 2017. Global normalization of convolutional neural networks for joint entity and relation classification. In *EMNLP*, pages 1723–1729.

Bekoulis, Ioannis, Johannes Deleu, and Thomas Demeester. 2018. Joint entity recognition and relation extraction as a multi-head selection problem. *Expert Systems with Applications*, 114:34–45.

Bekoulis, Ioannis, Johannes Deleu, Thomas Demeester, and Chris Develder. 2018. Adversarial training for multi-context joint entity and relation extraction. In *EMNLP*, pages 1–7.

Campero, Andres, Aldo Pareja, Tim Klinger, Josh Tenenbaum, and Sebastian Riedel. 2018. Logical rule induction and theory learning using neural theorem proving. *CoRR*, abs / 1809.02193.

Chan, Yee Seng and Dan Roth. 2011. Exploiting syntactico-semantic structures for relation extraction. In *ACL-HLT*, pages 551–560.

Chen, Zhuang and Tieyun Qian. 2020. Relation-aware collaborative learning for unified aspect-based sentiment analysis. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3685–3694.

Ciravegna, Gabriele, Francesco Giannini, Stefano Melacci, Marco Maggini, and Marco Gori. 2020. A constraint-based approach to learning and explanation. In *AAAI*.

Dai, Dai, Xinyan Xiao, Yajuan Lyu, Shan Dou, Qiaoqiao She, and Haifeng Wang. 2019a. Joint extraction of entities and overlapping relations using position-attentive sequence labeling. In *AAAI*, pages 6300–6308.

Dai, Hanjun, Bo Dai, and Le Song. 2016. Discriminative embeddings of latent variable models for structured data. In *ICML*, pages 2702–2711.

Dai, Wang-Zhou, Qiuling Xu, Yang Yu, and Zhi-Hua Zhou. 2019b. Bridging machine learning and logical reasoning by abductive learning. In *NeurIPS*. pages 2815–2826.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pages 4171–4186.

Dixit, Kalpit and Yaser Al-Onaizan. 2019. Span-level model for relation extraction. In *ACL*, pages 5308–5314.

Doddington, George, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. 2004. The automatic content extraction (ACE) program – tasks, data, and evaluation. In *LREC*.

Dong, Honghua, Jiayuan Mao, Tian Lin, Chong Wang, Lihong Li, and Denny Zhou. 2019. Neural logic machines. In *ICLR*.

Evans, Richard and Edward Grefenstette. 2018. Learning explanatory rules from noisy data. *J. Artif. Intelligent Res.*, 61:1–64.

França, Manoel V., Gerson Zaverucha, and Artur S. D'avila Garcez. 2014. Fast relational learning using bottom clause propositionalization with artificial neural networks. *Mach. Learn.*, 94(1):81–104.

Gallaire, H. and J. Minker. 1978. Logic and data bases, symposium on logic and data bases, centre d'études et de recherches de toulouse, 1977. In *Advances in Data Base Theory*.

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

Garcez, Artur S. d'Avila, Krysia B. Broda, and Dov M. Gabbay. 2002. *Neural-symbolic learning systems - foundations and applications*. Perspectives in neural computing. Springer.

Guo, Shu, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. 2016. Jointly embedding knowledge graphs and logical rules. In *EMNLP*.

Gupta, Pankaj, Hinrich Schütze, and Bernt Andrassy. 2016. Table filling multi-task recurrent neural network for joint entity and relation extraction. In *COLING*, pages 2537–2547.

Hong, Yu, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. 2011. Using cross-entity inference to improve event extraction. In *ACL-HLT*, pages 1127–1136.

Hu, Minqing and Bing Liu. 2004. Mining and summarizing customer reviews. In *KDD*, pages 168–177.

Hu, Zhiting, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. 2016. Harnessing deep neural networks with logic rules. In *ACL*, pages 2410–2420.

Ji, Heng and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *Proceedings of ACL-08: HLT*, pages 254–262.

Computational Linguistics Just Accepted MS.
https://doi.org/10.1162/coli_a_00415

Judea, Alex and Michael Strube. 2016. Incremental global event extraction. In *COLING*, pages 2279–2289.

Kate, Rohit J. and Raymond Mooney. 2010. Joint entity and relation extraction using card-pyramid parsing. In *CoNLL*, pages 203–212.

Katiyar, Arzoo and Claire Cardie. 2017. Going out on a limb: Joint extraction of entity mentions and relations without dependency trees. In *ACL*, pages 917–928.

Kipf, Thomas N. and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations*, ICLR '17.

Klement, Erich Peter, Radko Mesiar, and Endre Pap. 2013. Triangular norms. *Springer Science and Business Media*.

Lafferty, John D., Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289.

Lamb, Luis C., Artur S. d'Avila Garcez, Marco Gori, Marcelo OR Prates, Pedro HC Avelar, and Moshe Y. Vardi. 2020. Graph neural networks meet neural-symbolic computing: A survey and perspective. In *IJCAI*, pages 4877–4884.

Li, Fangtao, Chao Han, Minlie Huang, Xiaoyan Zhu, Ying-Ju Xia, Shu Zhang, and Hao Yu. 2010. Structure-aware review mining and summarization. In *COLING*, pages 653–661.

Li, Qi and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *ACL*, pages 402–412.

Li, Qi, Heng Ji, Yu Hong, and Sujian Li. 2014. Constructing information networks using one single model. In *EMNLP*, pages 1846–1851.

Li, Tao and Vivek Srikumar. 2019. Augmenting neural networks with first-order logic. In *ACL*, pages 292–302.

Li, Xiaoya, Fan Yin, Zijun Sun, Xiayu Li, Arianna Yuan, Duo Chai, Mingxin Zhou, and Jiwei Li. 2019. Entity-relation extraction as multi-turn question answering. In *ACL*, pages 1340–1350.

Li, Xin and Wai Lam. 2017. Deep multi-task learning for aspect term extraction with memory interaction. In *EMNLP*, pages 2886–2892.

Liao, Shasha and Ralph Grishman. 2010. Using document level cross-event inference to improve event extraction. In *ACL*, pages 789–797.

Lin, Yankai, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *ACL*, pages 2124–2133.

Lin, Ying, Heng Ji, Fei Huang, and Lingfei Wu. 2020. A joint neural model for information extraction with global features. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7999–8009.

Liu, Pengfei, Shafiq Joty, and Helen Meng. 2015. Fine-grained opinion mining with recurrent neural networks and word embeddings. In *EMNLP*.

Liu, Xiao, Zhunchen Luo, and Heyan Huang. 2018. Jointly multiple events extraction via attention-based graph information aggregation. In *EMNLP*, pages 1247–1256.

Luan, Yi, Dave Wadden, Luheng He, Amy Shah, Mari Ostendorf, and Hannaneh Hajishirzi. 2019. A general framework for information extraction using dynamic span graphs. In *NAACL*, pages 3036–3046.

Manhaeve, Robin, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. 2018. Deepproblog: Neural probabilistic logic programming. In *NeurIPS*. pages 3749–3759.

Manning, Christopher D., Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60.

Martins, Andre and Ramon Astudillo. 2016. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *ICML*, volume 48, pages 1614–1623.

McClosky, David, Mihai Surdeanu, and Christopher Manning. 2011. Event extraction as dependency parsing. In *ACL-HLT*, pages 1626–1635.

Minervini, Pasquale, Matko Bosnjak, Tim Rocktäschel, Sebastian Riedel, and Edward Grefenstette. 2020. Differentiable reasoning on large knowledge bases and natural language. In *AAAI*, pages 5182–5190.

Minervini, Pasquale, Thomas Demeester, Tim Rocktäschel, and Sebastian Riedel. 2017. Adversarial sets for regularised neural link predictors. In *UAI*.

Minervini, Pasquale and Sebastian Riedel. 2018. Adversarially regularising neural NLI models to integrate logical background knowledge. In *CoNLL*, pages 65–74.

Computational Linguistics Just Accepted MS.
https://doi.org/10.1162/coli_a_00415

Miwa, Makoto and Mohit Bansal. 2016. End-to-end relation extraction using LSTMs on sequences and tree structures. In *ACL*.
Miwa, Makoto and Yutaka Sasaki. 2014. Modeling joint entity and relation extraction with table representation. In *EMNLP*.
Miwa, Makoto, Paul Thompson, Ioannis Korkontzelos, and Sophia Ananiadou. 2014. Comparable study of event extraction in newswire and biomedical domains. In *COLING*, pages 2270–2279.
Neal, Radford M. and Geoffrey E. Hinton. 1999. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in Graphical Models*. pages 355–368.
Nguyen, Thien Huu, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *NAACL*, pages 300–309.
Nguyen, Trung Minh and Thien Huu Nguyen. 2019. One for all: Neural joint modeling of entities and events.
Nilsson, Nils J. 1986. Probabilistic logic. *Artif. Intell.*, 28(1):71–87.
Patwardhan, Siddharth and Ellen Riloff. 2009. A unified model of phrasal and sentential evidence for information extraction. In *EMNLP*, pages 151–160.
Pontiki, Maria, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *SemEval*.
Poon, Hoifung and Lucy Vanderwende. 2010. Joint inference for knowledge extraction from biomedical literature. In *NAACL*, pages 813–821.
Qiu, Guang, Bing Liu, Jiajun Bu, and Chun Chen. 2011. Opinion word expansion and target extraction through double propagation. *Comput. Linguist.*, 37(1):9–27.
Qu, Meng, Yoshua Bengio, and Jian Tang. 2019. GMNN: Graph Markov neural networks. In *ICML*, volume 97, pages 5241–5250.
Qu, Meng and Jian Tang. 2019. Probabilistic logic neural networks for reasoning. In *NeurIPS*.
Richardson, Matthew and Pedro M. Domingos. 2006. Markov logic networks. *Machine Learning*, 62(1-2):107–136.
Riedel, Sebastian, Hong-Woo Chun, Toshihisa Takagi, and Jun'ichi Tsujii. 2009. A Markov Logic approach to bio-molecular event extraction. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 41–49.
Rocktäschel, Tim and Sebastian Riedel. 2017. End-to-end differentiable proving. In *Advances in Neural Information Processing Systems 30*. pages 3788–3800.
Rocktäschel, Tim, Sameer Singh, and Sebastian Riedel. 2015. Injecting logical background knowledge into embeddings for relation extraction. In *NAACL*, pages 1119–1129.
Roth, Dan and Wen-tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *HLT-NAACL 2004 Workshop: CoNLL-2004*, pages 1–8.
Roth, Dan, Wen-tau Yih, and Scott Wen-tau Yih. 2007. Global inference for entity and relation identification via a linear programming formulation. *Introduction to Statistical Relational Learning*.
Serafini, Luciano and Artur S. d'Avila Garcez. 2016. Logic tensor networks: Deep learning and logical reasoning from data and knowledge. *CoRR*, abs/1606.04422.
Sha, Lei, Feng Qian, Baobao Chang, and Zhifang Sui. 2018. Jointly extracting event triggers and arguments by dependency-bridge rnn and tensor-based argument interaction. In *AAAI*.
Shanahan, Murray, Kyriacos Nikiforou, Antonia Creswell, Christos Kaplanis, David Barrett, and Marta Garnelo. 2019. An explicitly relational neural network architecture. *CoRR*, abs/1905.10307.
Sun, Changzhi, Yuanbin Wu, Man Lan, Shiliang Sun, Wenting Wang, Kuang-Chih Lee, and Kewen Wu. 2018. Extracting entities and relations with joint minimum risk training. In *EMNLP*, pages 2256–2265.
Takanobu, Ryuichi, Tianyang Zhang, Jiexi Liu, and Minlie Huang. 2019. A hierarchical framework for relation extraction with reinforcement learning. In *AAAI*.
Tran, Son N. and Artur S. d'Avila Garcez. 2018. Deep logic networks: Inserting and extracting knowledge from deep belief networks. *IEEE Trans. Neural Networks Learn. Syst.*, 29(2):246–258.
Venugopal, Deepak, Chen Chen, Vibhav Gogate, and Vincent Ng. 2014. Relieving the computational bottleneck: Joint inference for event extraction with high-dimensional features. In *EMNLP*, pages 831–843.
Wadden, David, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 2019. Entity, relation, and event extraction with

Computational Linguistics Just Accepted MS.
https://doi.org/10.1162/coli_a_00415

contextualized span representations. 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 5784–5789.

Wang, Hai and Hoifung Poon. 2018. Deep probabilistic logic: A unifying framework for indirect supervision. In *EMNLP*, pages 1891–1902.

Wang, Jue and Wei Lu. 2020. Two are better than one: Joint entity and relation extraction with table-sequence encoders. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1706–1721.

Wang, Po-Wei, Priya L. Donti, Bryan Wilder, and J. Zico Kolter. 2019. Satnet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In *ICML*, pages 6545–6554.

Wang, Shaolei, Yue Zhang, Wanxiang Che, and Ting Liu. 2018. Joint extraction of entities and relations based on a novel graph scheme. In *IJCAI*, pages 4461–4467.

Wang, Wenya and Sinno Jialin Pan. 2020. Integrating deep learning with logic fusion for information extraction. In *AAAI*.

Wang, Wenya, Sinno Jialin Pan, Daniel Dahlmeier, and Xiaokui Xiao. 2016. Recursive neural conditional random fields for aspect-based sentiment analysis. In *EMNLP*.

Wang, Wenya, Sinno Jialin Pan, Daniel Dahlmeier, and Xiaokui Xiao. 2017. Coupled multi-layer tensor network for co-extraction of aspect and opinion terms. In *AAAI*.

Xu, Hu, Bing Liu, Lei Shu, and Philip S. Yu. 2018a. Double embeddings and CNN-based sequence labeling for aspect extraction. In *ACL*, pages 592–598.

Xu, Jingyi, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van den Broeck. 2018b. A semantic loss function for deep learning with symbolic knowledge. In *ICML*, pages 5502–5511.

Yang, Bishan and Claire Cardie. 2013. Joint inference for fine-grained opinion extraction. In *ACL*, pages 1640–1649.

Yang, Bishan and Tom M. Mitchell. 2016. Joint extraction of events and entities within a document context. In *NAACL*, pages 289–299.

Yang, Yuan and Le Song. 2020. Learn to explain efficiently via neural logic inductive learning. *ICLR*.

Yin, Yichun, Furu Wei, Li Dong, Kaimeng Xu, Ming Zhang, and Ming Zhou. 2016. Unsupervised word and dependency path embeddings for aspect term extraction. In *IJCAI*.

Yu, Jianfei, Jing Jiang, and Rui Xia. 2019. Global inference for aspect and opinion terms co-extraction based on multi-task neural networks. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 27(1):168–177.

Yu, Xiaofeng and Wai Lam. 2010. Jointly identifying entities and extracting relations in encyclopedia text via a graphical model approach. In *COLING*, pages 1399–1407.

Zeng, Xiangrong, Daojian Zeng, Shizhu He, Kang Liu, and Jun Zhao. 2018. Extracting relational facts by an end-to-end neural model with copy mechanism. In *ACL*, pages 506–514.

Zhang, Meishan, Yue Zhang, and Guohong Fu. 2017. End-to-end neural relation extraction with global optimization. In *EMNLP*.

Zhang, Tongtao, Heng Ji, and Avirup Sil. 2019. Joint entity and event extraction with generative adversarial imitation learning. *Data Intell.*, 1(2):99–120.

Zheng, Suncong, Feng Wang, Hongyun Bao, Yuexing Hao, Peng Zhou, and Bo Xu. 2017. Joint extraction of entities and relations based on a novel tagging scheme. In *ACL*, pages 1227–1236.

Computational Linguistics Just Accepted MS.
https://doi.org/10.1162/coli_a_00415