# Multi-View Consistency for Relation Extraction via Mutual Information and Structure Prediction

**Amir Pouran Ben Veyseh,[1] Franck Dernoncourt,[2] My Thai,[3,4] Dejing Dou,[1] Thien Huu Nguyen[1,5]**
[1]Department of Computer and Information Science, University of Oregon, Eugene, Oregon 97403, USA
[2]Adobe Research, San Jose, CA, USA
[3]Department of Computer and Information Science and Engineering, University of Florida
[4]VIN Bigdata Institute, Hanoi, Vietnam
[5]VinAI Research, Hanoi, Vietnam
{apouranb, dou, thien}@cs.uoregon.edu, mythai@cise.ufl.edu, dernonco@adobe.com

## Abstract

Relation Extraction (RE) is one of the fundamental tasks in Information Extraction. The goal of this task is to find the semantic relations between entity mentions in text. It has been shown in many previous work that the structure of the sentences (i.e., dependency trees) can provide important information/features for the RE models. However, the common limitation of the previous work on RE is the reliance on some external parsers to obtain the syntactic trees for the sentence structures. On the one hand, it is not guaranteed that the independent external parsers can offer the optimal sentence structures for RE and the customized structures for RE might help to further improve the performance. On the other hand, the quality of the external parsers might suffer when applied to different domains, thus also affecting the performance of the RE models on such domains. In order to overcome this issue, we introduce a novel method for RE that simultaneously induces the structures and predicts the relations for the input sentences, thus avoiding the external parsers and potentially leading to better sentence structures for RE. Our general strategy to learn the RE-specific structures is to apply two different methods to infer the structures for the input sentences (i.e., two views). We then introduce several mechanisms to encourage the structure and semantic consistencies between these two views so the effective structure and semantic representations for RE can emerge. We perform extensive experiments on the ACE 2005 and SemEval 2010 datasets to demonstrate the advantages of the proposed method, leading to the state-of-the-art performance on such datasets.

## Introduction

Relation Extraction (RE) is an important task in Information Extraction (IE) that aims to identify the semantic relations between entity mentions in text. Finding such semantic relations could be beneficial for many downstream applications in Natural Language Processing, including Question Answering and Knowledge Base Population. Due to its importance, RE has gained much attention from the natural language processing (NLP) community, demonstrated by a sheer amount of research on RE in the last decades.

The early methods for RE have involved the feature-based approach and the kernel-based approach (Zelenko, Aone, and Richardella 2003; Zhou et al. 2005) where extensive feature engineering is necessary to produce effective RE models. Recently, the research in RE has essentially transformed from such feature engineering approaches to the deep learning models that have helped to significantly advance our performance on the RE benchmark datasets. Among different techniques introduced by deep learning for RE, the syntactic tress (i.e., dependency trees and constituent trees) have been shown to be a useful resource to impose the structures over the computational graphs of the network architectures (Socher et al. 2012; Xu et al. 2015; Zhang, Qi, and Manning 2018). The major benefits of such syntactic structures involve the incorporation of the semantic/syntactic hierarchies in the sentence representations (Socher et al. 2012) and the capacity to directly capture the important context words for RE (i.e., via the dependency paths) (Zhang, Qi, and Manning 2018).

Despite such advantages of the syntactic structures for the deep learning models for RE, a major limitation in this approach is the reliance on some high-quality external parsers to produce the effective parse trees for the models. There are at least three issues originated from this parser reliance. First, the high-quality external parsers might only be available for some domains and languages, thus restricting the application of the RE models to those specific scenarios in practice. Second, as the external parsers are often trained only for the parsing purposes, the tree structures provided by these syntactic parsers might not be the optimal ones for RE, calling for more customized or task-specific structures for the RE problem. In fact, the syntactic trees generated by the external parsers cannot be used directly by the recent deep learning models for RE, necessitating some additional post-processing or controling mechanisms (i.e., pruning and graph attention) (Zhang, Qi, and Manning 2018; Guo, Zhang, and Lu 2019). Finally, the external parsers are often trained for some general or specific domains for which their performance might degrade if they are applied to new domains (i.e., the domain shift problem) (Plank 2011). Such performance loss of the parsers on the new domains might eventually propagate to the RE models that critically depends on the quality of the tree structures for the sentences to perform well.

In order to overcome the reliance on the external parsers, in this work, we propose to learn the implicit structures for the input sentences along with the relationship prediction for the entity metions for RE. This would help to avoid the external parsers for RE, making it possible to apply the RE models for different domains and languages, providing the task-specific sentence structures for RE, and potentially improving the RE performance on new domains. To this goal, we propose to learn the sentence structures for RE by applying two different methods to generate the dependencies/hierarchies between the words in the sentences (i.e., the two views). We would then introduce the constraints between the structures and representations learned by these two views to promote their consistencies. Our expectation is that such consistency constraints, once enforced under the relation prediction task in RE, can help to reveal the effective task-specific structures and representations for RE, a property that is impossible if the structures are pre-determined with some external parsers.

In particular, the two views for inducing the structures of the sentences involve Ordered Neurons Long-short Term Memory (ON-LSTM) (Shen et al. 2019) and self-attention (Vaswani et al. 2017) (i.e., Transformers). ON-LSTM is an extended version of the popular Long-short Term Memory networks (LSTM) that, via its internal forget and input mechanisms, can assign an importance score for each word in the sentence. Such importance scores indicate how close the words should be to the root of the tree structure induced by On-LSTM for the input sentence (i.e., the root would have highest importance score), thus implicitly forming a tree structure for the sentence. In order to perform its computation, ON-LSTM manipulates the life cycle of each neurons so the high-ranking neurons would be maintained for a larger number of steps (words) while the low-ranking neurons would be discarded more rapidly. In contrast to the word order schema for structure induction in ON-LSTM, the seft-attention mechanism induces the structure for an input sentence by estimating the connection scores between every pair of words in the sentence (i.e., a fully connected graph structure). The score for one pair of word reflects how influential one word is with respect to the semantic understanding for the other word. Finally, in order to encourage the structure consistency between ON-LSTM and seft-attention, we transforms the pairwise scores between the words in self-attention into the importance scores for the words and promote the similarity between the two importance score sequences (i.e., from ON-LSTM and self-attention) using the KL divergence in the loss function.

In the baseline version of the aforementioned similarity promotion, we consider the importance scores of every word in the input sentence for both ON-LSTM and self-attention. However, for RE, it is possible that only a subset of the words in the sentence are necessary to correctly recognize the relationships for the entity mentions. It is thus desirable to perform the similarity promotion only on the importance scores of those relevant context words to avoid any potential noise. Consequently, in this work, we propose a filtering technique that predicts which context words in the sentence are relevant for the RE problem, and incorporates such information into the similarity promotion process to further improve the induced structures for RE.

A potential issue with the word representations induced by ON-STM and self-attention is that such word representations are excessively constrained to achieve the importance score similarity for the structure consistency, thereby loosing the important semantic information for RE. In order to alleviate this problem, in addition to the structure consistency, we also introduce the constraints to preserve the important semantic information in the representation vectors produced by ON-LSTM and self-attention. In particular, we first use a bidirectional LSTM (BiLSTM) model to encode the semantic representations of the words in the its hidden vectors. We would then enrich the semantic content of the hidden vectors from ON-LSTM and self-attention via the semantic consistency between those vectors. In particular, we consider two mechanisms to achieve such semantic consistencies between the hidden/representation vectors in this work. The first mechanism is inspired by the control mechanism in (Veyseh, Nguyen, and Dou 2019) that retains the semantic content in the representation vectors of self-attention via the control vector computed from the BiLSTM vectors of the two entity mentions. The second mechanism (newly proposed in this work), on the other hand, exploits the mutual information (MI) between the high dimension representation vectors from ON-LSTM and BiLSTM to enable their semantic consistency.

We conduct extensive experiments on the ACE 2005 and SemEval 2010 datasets that demonstrate the effectiveness of the proposed model for RE. Our model significantly outperforms the competitive baselines and achieves the state-of-the-art performance on the datasets.

## Related Work

The early works have used the feature or kernel based approaches for RE (Zelenko, Aone, and Richardella 2003; Zhou et al. 2005; Sun, Grishman, and Sekine 2011; Chan and Roth 2010; Nguyen and Grishman 2014; Nguyen, Plank, and Grishman 2015c). These approaches often perform extensive feature engineering and might not generalize well on challenging datasets. Recently, deep learning models have been proposed to address such issues, leading to the state-of-the-art results for RE (Zeng et al. 2014; Wang et al. 2016; Nguyen and Grishman 2016; Zhang et al. 2017). These deep learning models can be further categorized into the sequence based or structure based models. In the sequence based models, the sequential order of the words are preserved in the processing flow of the models (e.g., CNN (Nguyen and Grishman 2015a), RNN (Zhang et al. 2017) or Transformer (Verga, Strubell, and McCallum 2018)). In contrast, the structure-based models utilize the syntactic trees of the input sentences to structure the computational graphs of the deep learning models, thereby being able to capture the longer-term dependencies for the words. (Peng et al. 2017; Tai, Socher, and Manning 2015; Song et al. 2018; Xu et al. 2015; Liu et al. 2015; Miwa and Bansal 2016; Nguyen and Grishman 2018a; Zhang, Qi, and Manning 2018). However, in practice, such syntactic trees often require post-processing step (i.e., rule-based or attention-based) to customize them for RE (Zhang, Qi, and Manning 2018; Guo, Zhang, and Lu 2019). Our work differs from these prior works as we introduce a new mechanism to automatically induce the structures for RE from the context. The learned structures are customized for RE and do not require post-processing steps.

## Method

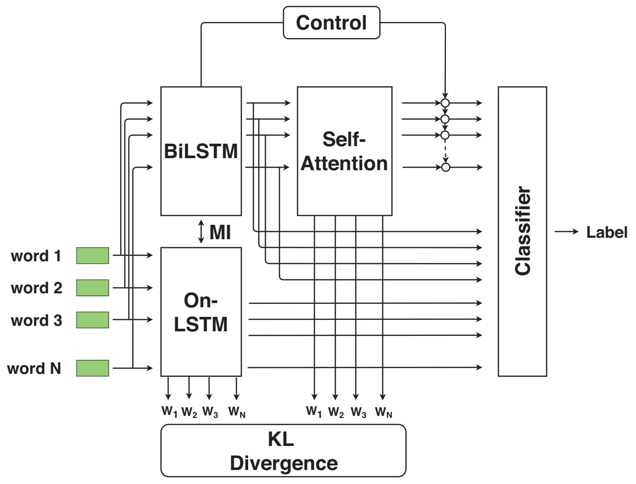

Figure 1: Model overview. The green vectors represent input word representations while the circles indicate the element-wise product.

The RE task can be seen as a multi-class classification problem. Given a sentence $W = w_1, w_2, \dots, w_N$ ($N$ is the number of words/tokens in $W$) and two entity mentions of interest located at tokens $w_s$ and $w_o$ ($1 \leq s < o \leq N$), we need to predict the semantic relationship between the two entity mentions in this case.

Motivated by the prior word on RE (Shi et al. 2018; Veyseh, Nguyen, and Dou 2019), we represent each word $w_i$ in the sentence using the concatenation vector $e_i$ of its pre-trained word embeddings, position embeddings (to indicate the positions of the two entity mentions), and entity type embeddings (to capture the entity mentions in the sentence)[1]. After this word-to-vector transformation, the input sentence $W$ is converted into a sequence of word representation vectors $E = e_1, e_2, \dots, e_N$ that would be used as the inputs for the next neural computations.

In order to avoid the external parsers, we propose to induce the task-specific structures during the relation prediction process for RE. Given the input sentence for RE, we employ two different network architectures to obtain the implicit structure and semantic representations for the words in the sentence. Several constraints are then introduced to ensure the consistencies between the structures and semantic representations learned by the two views. As mentioned in the introduction, the ON-LSTM and self-attention networks are used for the two views (called as the word-order view and the graph-based view respectively). Figure 1 shows the overall architecture of the proposed model. The details about the network architectures and consistency constraints are presented below.

**The word-order structure view** The strategy to induce a structure in the word-order view with ON-LSTM is to assign an importance score $w_i^{onlstm}$ for every word $w_i$ in the input sentence to implicitly form a binary tree structure for $W$. The words with higher importance scores would be closer to the root of the tree, reflecting the levels of the words in the tree. Consequently, the word $w_{i*}$ with the highest score $w_{i*}^s$ would be considered as the tree root, from which two subtrees are recursively constructed based on the words before $w_{i*}$ for the left child and the words after $w_{i*}$ for the right child.

ON-LSTM computes the importance scores for the words by introducing two additional master gates (i.e., *forget* and *input*) into the original computation of LSTM (Shen et al. 2019) (i.e., ON-LSTM is thus similar to LSTM in that both consume a sequence of vectors to produce a new sequence of hidden vectors). Basically, the hidden vectors for the *input* and *forget* gates (called the gate vectors) in both LSTM and ON-LSTM are computed for each word/step in the sentence and determine how much information of the context should be updated or forgotten respectively in the current step. However, the *forget* and *input* gate vectors in ON-LSTM differ from those in LSTM because the gates in LSTM treat every neurons/dimensions in their hidden vectors independently, imposing no hierarchy over such neurons and assuming the activity of each neuron for every word in the sentence/sequence. This is in contrast to the *forget* and *input* gates in ON-LSTM that enforce a hierarchy for the neurons/dimensions in their hidden vectors and only allow each neuron to be activated for a portion of words in the sentence. The rationale is that higher-ranking neurons would have a longer life time (i.e., being activated for more words in the sentence) to encode long-term information while lower-ranking neurons would be canceled more rapidly to focus on the short-term information (i.e., the structural bias).

In order to achieve such ranking mechanisms for the neurons in the master gates, ON-LSTM employs the *cummax* activation function in the computation for the hidden vectors of the *forget* and *input* gates: $cumax(x) = cumsum(softmax(x))$[2]. *cummax* essentially aggregates the softmax output of some input vector $x$ along the dimensions that can be seen as the expectation of some binary vector of the form $(0, \dots, 0, 1, \dots, 1)$ (i.e., divided into two consecutive segments: the 0-segment and the 1-segment). Similar to the *forget* and *input* gates in LSTM, the input for the *cummax* activation function to compute the gate vectors for ON-LSTM at the current step/word also involves the hidden vector from the previous step and the input vector for the current step. At one word/step, the 1-segments of the hidden vectors of the master gates cover the neurons

[1]Note that different from (Veyseh, Nguyen, and Dou 2019), we do not include the binary feature vectors for $w_i$ obtained from the dependency trees (i.e., from the dependency relations and paths) as we would like to avoid the parse trees in this work.

[2]$cumsum(u_1, u_2, \dots, u_n) = (u'_1, u'_2, \dots, u'_n)$ where $u'_i = \sum_{j=1..i} u_j$.

that are activated for the gates at that step. Consequently, in ON-LSTM, the lengths of the 1-segments, or more precisely the sums of the weights of the neurons in the 1-segments of the gate vectors for a word are used to determine the importance of that word in the sentence. Following (Shen et al. 2019), we use the hidden vectors of the master *forget* gate in ON-LSTM to obtain the importance scores the words. In particular, let $f_i = f_{i1}, f_{i2}, \ldots, f_{iD}$ be the hidden vector for the master *forget* gate at the $i$-th word $w_i \in W$ from ON-LSTM ($D$ is the dimension of the hidden vector), the importance score $w_i^{onlstm}$ for $w_i$ is computed by: $w_i^{onlstm} = D - \sum_{j=1..D} f_{ij}$.

In this work, we feed the input vector sequence $E = e_1, e_2, \ldots, e_N$ into two layers of ON-LSTM. We use the master *forget* gates of the second layer to generate the importance sores $w_i^{onlstm}$ for the words in $W$, serving as the encoding of the tree structure induced by ON-LSTM in this work. For convenience, we denote the output hidden vectors produced by the second layer of ON-LSTM for the words in the input sentence as $H' = h'_1, h'_2, \ldots, h'_N$.

**The graph-based structure view** Different from the word importance scores via the number of active neurons in ON-LSTM, the graph-based structure view represents the structure for the input sentence via a fully connected graph between the words. The weight $a_{ij}$ for the edge between $w_i$ and $w_j$ would indicate the level of connections/dependencies of these two words, thus implicitly defining a hierarchy among the words.

As presented previously, we obtain the graph-based structure for the input sentence in this work via the self-attention mechanisms in Transformers (Vaswani et al. 2017). In particular, starting from the input representation vectors for the words $E = e_1, e_2, \ldots, e_N$, we first feed them into a bidirectional LSTM layer (BiLSTM) that produces a sequence of hidden vectors $H = h_1, h_2, \ldots, h_N$ as the output. These representation vectors are expected to capture the semantic information for the whole input sentence. Afterward, the BiLSTM would be consumed by the self-attention layer to generate the connection scores for the pairs of words in the sentence. Specifically, for each BiLSTM vector $h_i$, we compute its corresponding key vector $k_i$, query vector $q_i$, and value vector $v_i$ via: $k_i = U_k h_i$, $q_i = U_q h_i$, and $v_i = U_v h_i$ where $U_k$, $U_q$ and $U_v$ are the weight matrices, and biases are omitted for brevity in this work. The connection scores $a_{ij}$ between the words $w_i$ and $w_j$ would then be obtained by the dot product between $k_i$ and $q_j$:

$$a_{i,j} = \exp(k_i \cdot q_j) / \sum_{t=1..N} \exp(k_i \cdot q_t) \tag{1}$$

We omit the normalization factor in this formula in (Vaswani et al. 2017) for brevity. The connection scores $a_{ij}$ of the graph structure induced by self-attention for the input sentence can be exploited for two purposes. First, they can be used to compute more abstract representation vectors $H'' = h''_1, h''_2, \ldots, h''_N$ for the words in the sentence via: $h''_i = \sum_{j=1..n} a_{ij} v_j$. Such vectors are expected to encode richer context information for the input sentence with a greater focus on the induced graph structure information. Second, the connection scores $a_{ij}$ can also be utilized to transform the induced graph structure into a tree structure by computing an importance score for every word in the sentence and following the same strategy to generate the binary tree as we do for the ON-LSTM model. In order to obtain the importance scores for the words with $a_{ij}$, we assume that a word would be more important if it has stronger connections with the other words. In particular, we compute the importance score $w_i^{satt}$ for the word $w_i$ by:

$$w_i^{satt} = \Sigma_{j=1}^{N} a_{i,j} / N \tag{2}$$

where the weights $a_{i,i}$ are set to be zero. We consider the scores $w_1^{satt}, w_2^{satt}, \ldots, w_N^{satt}$ as the encoding for the tree structure learned by seft-attention in this work.

**Structure consistency between views** As we do not have any supervision about the structure of the input sentence in this work, we seek to induce such structure automatically by promoting the similarity between the structures learned by the word-order and graph-based views. Intuitively, the word-order and graph-based structures should be similar/related as they are computed for the same input sentence $W$. We expect that such structure similarity enforcement would help to reveal the effective structures for RE. In order to achieve such structure similarity, we first transform the structure-encoding scores $w_1^{onlstm}, w_2^{onlstm}, \ldots, w_N^{onlstm}$ and $w_1^{satt}, w_2^{satt}, \ldots, w_N^{satt}$ from ON-LSTM and self-attention into the probability distributions $W^{onlstm}$ and $W^{satt}$ (respectively) via: $W^{onlstm} = softmax([w_1^{onlstm}, w_2^{onlstm}, \ldots, w_N^{onlstm}])$ and $W^{satt} = softmax([w_1^{satt}, w_2^{satt}, \ldots, w_N^{satt}])$. We would then incorporate the KL divergence between $W^{onlstm}$ and $W^{satt}$ into the overall loss function to minimize the distance between the two distribution:

$$KL(W^{onlstm} || W^{satt}) = -\Sigma_i W_i^{onlstm} log \frac{W_i^{onlstm}}{W_i^{satt}} \tag{3}$$

One potential issue with Equation 3 is that it enforces the structure similarity based on the importance scores for the words in the sentence equally (i.e., imposing the same weight for the word-specific term in the KL divergence). This implicitly assumes the equal contribution of the words for the structure of the sentence for RE. However, in RE, there might be only a subset of words in the sentence that are actually relevant or necessary for the semantic prediction (e.g., the words along the dependency path between the two entity mentions). This suggests a mechanism where the words are weighted differently in the KL divergence for the structure consistency based on their potential contribution for the relationship prediction of the two entity mentions. Consequently, we seek to estimate a contribution score $s_i$ for each word $w_i$ in the KL divergence based on the BiLSTM vectors $h_i$, $h_s$ and $h_o$ of $w_i$ and the two entity mentions of interest:

$$s_i = \sigma(W_1 \sigma(W_2 [h_i, h_s, h_o])) \tag{4}$$

where $W_1$ and $W_2$ are the weight matrices. Finally, we use such contribution scores to weight the word-specific terms in the KL divergence in Equation 3 in the overall loss:

$$L_{structure} = -\Sigma_i s_i W_i^{onlstm} log \frac{W_i^{onlstm}}{W_i^{satt}} \tag{5}$$

**Semantic consistency between views** Given the representation vectors learned ON-LSTM (i.e., $H' = h'_1, h'_2, \dots, h'_N$) and self-attention (i.e., $H'' = h''_1, h''_2, \dots, h''_N$), the natural approach to perform RE is to aggregate such representation vectors to create an overall feature vector for classification. However, the structure consistency constraint in Equation 5 might have filtered the information content in $H'$ and $H''$ excessively to encode mostly the structure information, erasing the important semantic information for RE. In order to enrich the representation vectors $H'$ and $H''$ with the semantic information, we propose to distill the semantic information from the BiLSTM vectors (i.e., $H = h_1, h_2, \dots, h_N$) and directly incorporate it into $H'$ and $H''$ for RE. As the representation vectors in $H$ are less involved with the structure constraint, we expect that they can still preserve important semantic information for RE in this case. In particular, we seek to employ mechanisms to promote the semantic consistencies between the BiLSTM representation vectors $H$ and those from ON-LSTM and self-attention (i.e., $H'$ and $H''$). We expect that such semantic consistencies can help to enrich the semantic information in the representation vectors $H'$ and $H''$.

First, for the semantic consistency between $H$ and $H'$, we propose to maximize the mutual information (MI) between their aggregated representations in the loss function. In information theory, MI evaluates how much information we know about one random variable if the value of another variable is revealed. Two random variables would be more dependent if they have larger mutual information. Consequently, if the semantic representations from $H$ and $H'$ are encouraged to have large mutual information, we expect them to share more semantic information. As $H$ already encodes the important semantic information for RE, this would help to enrich the semantic content in $H'$ as a by-product. In order to introduce this mutual information constraint, we first aggregate the representation vectors in $H$ and $H'$ into the overall representation vectors $\bar{h}$ and $\bar{h}'$ via the max-pooling function: $\bar{h} = Max\_Pooling(h_1, h_2, \dots, h_N)$ and $\bar{h}' = Max\_Pooling(h'_1, h'_2, \dots, h'_N)$. The mutual information would be computed between $\bar{h}$ and $\bar{h}'$ and introduced directly into the loss function for optimization.

One issue with this approach is that the computation of the MI for such high dimensional continuous vectors as $\bar{h}$ and $\bar{h}'$ is prohibitively expensive. In this work, we propose to address this issue by employing the mutual information neural estimation (MINE) in (Belghazi et al. 2018) that seeks to estimate the lower bound of the mutual information between the high dimensional vectors via adversarial training. To this goal, MINE attempts to compute the lower bound of the KL divergence between the joint and marginal distributions of the given high dimensional vectors/variables. Recently, in (Hjelm et al. 2019), the authors show that the Jensen-Shannon divergence can also used for this purpose, offering simpler methods to compute the lower bound for the MI. Consequently, following such methods, we apply the adversarial approach to obtain the MI lower bound via the binary cross entropy of a variable discriminator. This discriminator differentiates the vectors that are sampled from the joint distribution from those that are sampled from the product of the marginal distribution of the variables. In our case, the two variables are the semantic representations $\bar{h}$ and $\bar{h}'$. In order to sample from their joint distribution, we simply concatenate $\bar{h}$ and $\bar{h}'$ (i.e., the positive example). To sample from the product of the marginal distribution, we concatenate the representation $\bar{h}$ with $\hat{h}'$ where $\hat{h}'$ is the aggregated vector (with max-pooling) of the ON-LSTM representation vectors from another sentence in the same batch with the current sentence of interest $W$ (i.e., the negative example). These samples are fed into a 2-layer feed forward neural network $D$ (i.e., the discriminator) to perform a binary classification (i.e., coming from the joint distribution or the product of the marginal distributions). Finally, we use the following binary cross entropy loss to estimate the mutual information between $\bar{h}$ and $\bar{h}'$ to add into the overall loss function:

$$L_{disc} = -(log(D[\bar{h}, \bar{h}']) + log(1 - D([\bar{h}, \hat{h}']))) \quad (6)$$

Second, regarding $H$ and $H''$, we apply the control mechanism proposed in (Veyseh, Nguyen, and Dou 2019) to enforce the semantic consistency for these representation vectors. This control mechanism first obtains a control vector $c$ from the representation vectors in $H$, emphasizing on the representation vectors of the two entity mentions $h_s$ and $h_o$. This control vector is then applied directly to the representation vectors in $H''$, obtaining a new vector $\bar{h}''_i$ for each vector $h''_i \in H''$: $\bar{h}''_i = c \odot h''_i$.

For convenience, we use $\bar{h}''$ to denote the max-pooling aggregation vector for the representation vectors $\bar{h}''_i$: $\bar{h}'' = Max\_Pooling(\bar{h}''_1, \bar{h}''_2, \dots, \bar{h}''_N)$. Due to the direct incorporation, we expect that the semantic information in $H''$ would be consistent with those in $H$, thereby enriching the semantic content in $H''$. The control mechanism has been shown to work well for RE in (Veyseh, Nguyen, and Dou 2019). In the experiments, we will evaluate whether we can use the control mechanism and the new MI constraint interchangeably for the semantic consistency between $H$, $H'$ and $H''$.

**Training** Finally, in order to predict the relationships between the two entity mentions $w_s$ and $w_o$, we combine the representation vectors produced by BiLSTM, ON-LSTM, and self-attention to obtain an overall representation vector $R$ for the input sentence. This representation vector involves the max-pooling aggregation vectors from these three components (i.e., $\bar{h}$, $\bar{h}'$, and $\bar{h}''$) as well as their specific elements for the two entity mentions (i.e., $h_s$, $h_o$, $h'_s$, $h'_o$, $h''_s$ and $h''_o$): $R = [\bar{h}, \bar{h}', \bar{h}'', h_s, h_o, h'_s, h'_o, h''_s, h''_o]$. Due to the structure and semantic consistencies introduced in this work, we expect $R$ would contain effective information for the RE problem. In the final step, $R$ would be fed into a 2 layer feed-forward neural network followed by a softmax layer to compute the probability distribution $P(.|W, s, o)$ over the possible relations for RE. We use the negative log-likelihood as the training loss in this work: $L_{pred} = -P(y|W, s, o)$ where $y$ is the true relation label for the input sentence.

Overall, the loss function to train the model is:

$$L = L_{pred} + \alpha L_{disc} + \beta L_{structure} \quad (7)$$

where $\alpha$ and $\beta$ are the trade-off parameters.

| System | bc | cts | wl | Avg. |
|---|---|---|---|---|
| FCM (2015) | 61.90 | 52.93 | 50.36 | 55.06 |
| Hybrid FCM (2015) | 63.48 | 56.12 | 55.17 | 58.25 |
| LRFCM (2015) | 59.40 | - | - | - |
| Log-linear (2016) | 57.83 | 53.14 | 53.06 | 54.67 |
| CNN (2016) | 63.26 | 55.63 | 53.91 | 57.60 |
| Bi-GRU (2016) | 63.07 | 56.47 | 53.65 | 57.73 |
| Forward GRU (2016) | 61.44 | 54.93 | 55.10 | 57.15 |
| Backward GRU (2016) | 60.82 | 56.03 | 51.78 | 56.21 |
| CNN+DANN (2017) | 65.16 | - | - | - |
| GSN (2018) | 66.38 | 57.92 | 56.84 | 60.38 |
| AGGCN (2019) | 63.47 | 59.70 | 56.50 | 59.89 |
| SACNN (2019) | 65.06 | 61.71 | 59.82 | 62.20 |
| DRPC (2019) | 67.30 | 64.28 | 60.19 | 63.92 |
| MVC (ours) | **70.32** | **66.43** | **64.61** | **67.12** |

Table 1: F1 scores of the models on the ACE 2005 dataset over different target domains `bc`, `cts`, and `wl`.

## Experiments

**Datasets and Hyper-Parameters** We employ two widely used datasets (i.e., ACE 2005 and SemEval 2010) to evaluate the model in this work. For the ACE 2005 dataset, similar to the previous work (Fu et al. 2017; Shi et al. 2018; Veyseh, Nguyen, and Dou 2019), we use the dataset preprocessed and provided by (Yu, Gormley, and Dredze 2015) for compatible comparison. There are 6 different domains in this dataset, i.e., (`bc`, `bn`, `cts`, `nw`, `un`, and `wl`), covering text from news, conversations and web blogs. Following the the prior work, the union of the domains `bn` and `nw` (called `news`) is used as the training data (called the source domain); a half of the documents in `bc` is reserved for the development data, and the remainder (`cts`, `wl` and the other half of `bc`) serve as the test data (called the target domains). This data separation helps to evaluate the cross-domain generalization of the models due to the domain difference of the training data and test data. For the SemEval 2010 dataset (Hendrickx et al. 2010), there are 18 semantic relations that along with an *Other* class, leading to a 19-class classification problem. As validation data is not provided in SemEval 2018, we use the same model parameters as those used for the ACE 2005 dataset for consistency.

Based on the fine-tuning process on the validation data of the ACE 2005 dataset, we find the following values for the hyper-parameters for the proposed model: 50 dimensions for the position embeddings and entity type embeddings, 100 hidden units for the BiLSTM and ON-LSTM models, 200 dimensions for all the other hidden vectors in the model (i.e., the hidden vectors in self-attention and the layers of the feed-forward neural networks), 0.1 for the loss trade-off parameters $\alpha$ and $\beta$, and 0.3 for the learning rate with the Adam optimizer. Finally, we use the pre-trained word embedding `word2vec` with 300 dimension to represent the words.

**Comparison to the state of the art** We compare the performance of the proposed model (called MVC for multi-view consistency) with the following baselines:

• Feature based models: These models use linguistic features for RE, i.e., FCM, Hybrid FCM, LRFCM, and SVM (Yu, Gormley, and Dredze 2015; Hendrickx et al. 2010).

• Deep sequential models: These models employ deep learning architectures based on the sequential order of the sentence for RE, i.e., log-linear, CNN, Bi-GRU, Forward GRU, Backward GRU (Nguyen and Grishman 2016), and CNN+DANN (Fu et al. 2017).

• Adversarial learning model: This model, called GSN, is trained to learn the genre agnostic features for cross-domain RE. (Shi et al. 2018)

• Deep structure-based models: These models employ dependency trees either as the input features or graphs to form the computation flow for deep learning models. The state-of-the-art models of this type include: AGGCN (Guo, Zhang, and Lu 2019), SACNN (Tran et al. 2019) and DRPC (Veyseh, Nguyen, and Dou 2019). DRPC has the best reported performance on ACE 2005. Note that we obtain the performance of these models on the considered datasets using the actual implementation released by the original papers.

We report the F1 scores on all the ACE 2005 test sets in Table 1. From the table, we see that the deep structure-based models (i.e., AGGCN, SACNN and DRPC) are in general better than the deep sequential models, thus suggesting the benefits of the sentence structures (i.e., the dependency trees from the external parsers) for deep learning for RE. More importantly, the proposed model MVC is shown to significantly outperforms the baseline models on all the test sets with $p < 0.01$. The performance gap is substantial and clearly demonstrates the effectiveness of the proposed MVC in this work. In particular, MVC improves the average F1 score of the deep sequential models by almost 10% while this performance improvement for the structure-based model is at least 3%. We attribute the better performance of MVC over the structure-based models to the fact that the task-specific and context-dependent structures from MVC is better suited for RE than the pre-defined structures from the external parsers. Finally, due to the cross-domain nature of the evaluation on the ACE 2005 dataset, we can also conclude that the task-specific structures learned by MVC can be more robust against the domain shifts for RE.

We also compare the performance of MVC (i.e., using the macro F1 score) with the state-of-the-art structured-based RE models (i.e., using dependency trees) on the SemEval 2010 test set in Table 2. These models are also selected for comparison in (Veyseh, Nguyen, and Dou 2019). As we can see from the table, MVC can achieve significantly better or comparable performance with the other structure-based methods, further testifying to the advantages of the task-specific structure induction for RE proposed in this work.

**Ablation study on components** In this section, we report the performance of the proposed model on the ACE 2005 development set when the major components of the model is excluded. In particular, we seek to evaluate the contribution of four main components in this work, including the BiLSTM module, the ON-LSTM module, the self-attention module, and the contribution scores $s_i$ for the KL divergence constraint in Equation 4. The results are shown in Table 3. As we can see from the table, all the components are necessary for MVC as removing any of them would hurt the performance significantly. The largest performance loss comes

| System | F1 |
|---|---|
| SVM (2010) | 82.2 |
| SDP-LSTM (2015) | 83.7 |
| SPTree (2016) | 84.4 |
| PA-Tree (2017) | 82.7 |
| C-GCN (2018) | 84.8 |
| LISA (2018) | 83.9 |
| DRPC (2019) | 85.2 |
| AGGCN (2019) | 85.7 |
| SACNN (2019) | 85.8 |
| MVC (ours) | **86.1** |

Table 2: Performance on the SemEval 2010 dataset.

| System | P | R | F1 |
|---|---|---|---|
| MVC | 77.8 | 65.1 | 70.1 |
| MVC - LSTM | 75.7 | 62.9 | 68.0 |
| MVC - ON-LSTM | 72.1 | 63.4 | 67.5 |
| MVC - SA | 80.1 | 61.2 | 68.4 |
| MVC - Contribution Scores in (4) | 73.2 | 66.5 | 69.2 |

Table 3: Ablation study on the ACE 2005 dev set.

from excluding ON-LSTM that highlights the importance of ON-LSTM on inducing effective structure and semantic information for RE. Importantly, the performance loss due to the elimination of the contribution scores $s_i$ in Equation 4 suggests that in RE, learning the structure throughout the entire sentences would not be as helpful as restricting the structure induction to the relevant parts of the sentences.

**Semantic consistency** There are two mechanisms for semantic consistency in this work, i.e., the MI constraint and the control mechanism (Veyseh, Nguyen, and Dou 2019). In the model, the MI constraint is used to promote the semantic consistency between the representation vectors from BiLSTM and ON-LSTM (i.e., BiLSTM $\leftrightarrow$ ON-LSTM) while the control mechanism is applied for the vectors from BiLSTM and self-attention (i.e., BiLSTM $\leftrightarrow$ self-attention). The natural question is whether the MI and control mechanisms can be used interchangeably to achieve the semantic consistencies for the representation vectors for the two pairs BiLSTM $\leftrightarrow$ ON-LSTM and BiLSTM $\leftrightarrow$ self-attention. Table 4 shows the performance of the 4 possible combinations of the MI and control mechanisms for the two semantic consistency pairs BiLSTM $\leftrightarrow$ ON-LSTM and BiLSTM $\leftrightarrow$ self-attention. This table shows that when we use only one type of the semantic consistency mechanisms, i.e. both MI or both control, the performance drops more than the cases with two types of mechanisms. This demonstrates the complementary effects between the MI and control constraints for semantic consistency for RE. The best performance is achieved when the MI mechanism is used for BiLSTM $\leftrightarrow$ ON-LSTM and the control mechanism is reserved for BiLSTM $\leftrightarrow$ self-attention, testifying to our design of the proposed model in this work.

**Ablation study on consistency** This section investigates whether the consistency constraints (i.e., structure and semantics) are necessary. Table 5 presents the performance of MVC when different combinations of the consistency constraints are removed from the model. As we can see from the table, the model's performance would be reduced significantly if any of the constraints is excluded, among which the KL divergence constraint for the structure consistency between ON-LSTM and self-attention would lead to the most significant performance loss. Importantly, when all the three constraints are excluded from MVC (thus making it an ensemble model between ON-LSTM and self-attention), the performance would become the worst (i.e., 3% loss in the F1 score). These results clearly demonstrate the effectiveness of the proposed MVC in this work, highlighting the consistency constraints as the important mechanisms to achieve good performance for RE.

| Mechanisms | P | R | F1 |
|---|---|---|---|
| MI, Control (proposed) | 77.8 | 65.1 | 70.1 |
| MI, MI | 84.5 | 59.6 | 67.3 |
| Control, MI | 77.2 | 64.5 | 69.2 |
| Control, Control | 74.2 | 63.2 | 68.9 |

Table 4: Performance on the ACE 2005 dev set when the MI and control mechanisms are used interchangeably. The first and second mechanisms in each row corresponds to the constraints for BiLSTM $\leftrightarrow$ ON-LSTM and BiLSTM $\leftrightarrow$ self-attention respectively.

| Model | P | R | F1 |
|---|---|---|---|
| MVC | 77.8 | 65.1 | 70.1 |
| MVC - KL | 72.3 | 65.8 | 68.1 |
| MVC - MI | 74.0 | 67.4 | 69.5 |
| MVC - Control | 78.4 | 61.9 | 68.5 |
| MVC - KL - MI | 71.2 | 66.1 | 68.0 |
| MVC - MI - Control | 70.1 | 65.6 | 67.9 |
| MVC - KL - Control | 70.5 | 66.4 | 68.2 |
| MVC - KL - MI - Control | 72.1 | 63.2 | 67.1 |

Table 5: Performance on the ACE 2005 dev set when the consistency constraints are removed from the model.

## Conclusion

We propose a novel method for RE that seeks to automatically induce the task-specific structures for the input sentences, avoiding the external parsers. The experiments show that the induced structures are more effective for RE than the pre-defined structures from the external parsers. The key innovation of the proposed method is to use two views (i.e., ON-LSTM and self-attention) to learn the structures and semantic representations for the input sentences. We introduce several constraints to enforce the structure and semantic consistencies between the two views based on KL differences and mutual information. We achieve the state-of-the-art performance for RE on both the cross-domain and general settings, thereby demonstrating the effectiveness and the robustness of the proposed model.

## Acknowledgments

This work has been supported in part by Vingroup Innovation Foundation (VINIF) in project code VINIF.2019.DA18, IARPA BETTER, the NSF grant CNS-1747798 to the IU-

CRC Center for Big Learning, the NSF grant CNS-1814614, and Adobe Research Gift.

## References

Belghazi, M. I.; Baratin, A.; Rajeswar, S.; Ozair, S.; Bengio, Y.; Courville, A.; and Hjelm, R. D. 2018. Mine: mutual information neural estimation. In *ICML*.

Chan, Y. S., and Roth, D. 2010. Exploiting background knowledge for relation extraction. In *COLING*.

Fu, L.; Nguyen, T. H.; Min, B.; and Grishman, R. 2017. Domain adaptation for relation extraction with domain adversarial neural network. In *IJCNLP*.

Guo, Z.; Zhang, Y.; and Lu, W. 2019. Attention guided graph convolutional networks for relation extraction. In *ACL*.

Hendrickx, I.; Kim, S. N.; Kozareva, Z.; Nakov, P.; Ó Séaghdha, D.; Padó, S.; Pennacchiotti, M.; Romano, L.; and Szpakowicz, S. 2010. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *the 5th International Workshop on Semantic Evaluation*.

Hjelm, R. D.; Fedorov, A.; Lavoie-Marchildon, S.; Grewal, K.; Bachman, P.; Trischler, A.; and Bengio, Y. 2019. Learning deep representations by mutual information estimation and maximization. In *ICLR*.

Liu, Y.; Wei, F.; Li, S.; Ji, H.; Zhou, M.; and Wang, H. 2015. A dependency-based neural network for relation classification. In *ACL*.

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

Nguyen, T. H., and Grishman, R. 2014. Employing word representations and regularization for domain adaptation of relation extraction. In *ACL*.

Nguyen, T. H., and Grishman, R. 2015a. Relation extraction: Perspective from convolutional neural networks. In *Proceedings of the 1st NAACL Workshop on Vector Space Modeling for NLP (VSM)*.

Nguyen, T. H., and Grishman, R. 2016. Combining neural networks and log-linear models to improve relation extraction. In *Proceedings of IJCAI Workshop on Deep Learning for Artificial Intelligence*.

Nguyen, T. H., and Grishman, R. 2018a. Graph convolutional networks with argument-aware pooling for event detection. In *AAAI*.

Nguyen, T. H.; Plank, B.; and Grishman, R. 2015c. Semantic representations for domain adaptation: A case study on the tree kernel-based method for relation extraction. In *ACL-IJCNLP*.

Peng, N.; Poon, H.; Quirk, C.; Toutanova, K.; and Yih, W.-t. 2017. Cross-sentence n-ary relation extraction with graph lstms. In *TACL*.

Plank, B. 2011. Domain adaptation for parsing. In *Ph.D. thesis. University of Groningen*.

Shen, Y.; Tan, S.; Sordoni, A.; and Courville, A. 2019. Ordered neurons: Integrating tree structures into recurrent neural networks. In *ICLR*.

Shi, G.; Feng, C.; Huang, L.; Zhang, B.; Ji, H.; Liao, L.; and Huang, H. 2018. Genre separation network with adversarial training for cross-genre relation extraction. In *EMNLP*.

Socher, R.; Huval, B.; Manning, C. D.; and Ng, A. Y. 2012. Semantic compositionality through recursive matrix-vector spaces. In *EMNLP*.

Song, L.; Zhang, Y.; Wang, Z.; and Gildea, D. 2018. N-ary relation extraction using graph state lstm. In *EMNLP*.

Strubell, E.; Verga, P.; Andor, D.; Weiss, D.; and McCallum, A. 2018. Linguistically-informed self-attention for semantic role labeling. In *EMNLP*.

Sun, A.; Grishman, R.; and Sekine, S. 2011. Semi-supervised relation extraction with large-scale word clustering. In *ACL*.

Tai, K. S.; Socher, R.; and Manning, C. D. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *ACL*.

Tran, V.-H.; Phi, V.-T.; Shindo, H.; and Matsumoto, Y. 2019. Relation classification using segment-level attention-based cnn and dependency-based rnn. In *NAACL-HLT*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L. u.; and Polosukhin, I. 2017. Attention is all you need. In *NIPS*.

Verga, P.; Strubell, E.; and McCallum, A. 2018. Simultaneously self-attending to all mentions for full-abstract biological relation extraction. In *EMNLP*.

Veyseh, A. P. B.; Nguyen, T. H.; and Dou, D. 2019. Improving cross-domain performance for relation extraction via dependency prediction and information flow control. In *IJCAI*.

Wang, L.; Cao, Z.; de Melo, G.; and Liu, Z. 2016. Relation classification via multi-level attention cnns. In *EMNLP*.

Xu, Y.; Mou, L.; Li, G.; Chen, Y.; Peng, H.; and Jin, Z. 2015. Classifying relations via long short term memory networks along shortest dependency paths. In *EMNLP*.

Yu, M.; Gormley, M. R.; and Dredze, M. 2015. Combining word embeddings and feature embeddings for fine-grained relation extraction. In *NAACL-HLT*.

Zelenko, D.; Aone, C.; and Richardella, A. 2003. Kernel methods for relation extraction. In *Journal of machine learning research*.

Zeng, D.; Liu, K.; Lai, S.; Zhou, G.; and Zhao, J. 2014. Relation classification via convolutional deep neural network. In *COLING*.

Zhang, Y.; Zhong, V.; Chen, D.; Angeli, G.; and Manning, C. D. 2017. Position-aware attention and supervised data improve slot filling. In *EMNLP*.

Zhang, Y.; Qi, P.; and Manning, C. D. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *EMNLP*.

Zhou, G.; Su, J.; Zhang, J.; and Zhang, M. 2005. Exploring various knowledge in relation extraction. In *ACL*.