Graph Transformer Networks with Syntactic and Semantic Structures for Event Argument Extraction

Amir Pouran Ben Veyseh[1], Tuan Ngo Nguyen[1] and Thien Huu Nguyen[1,2]
[1] Department of Computer and Information Science, University of Oregon,
Eugene, OR 97403, USA
[2] VinAI Research, Vietnam
`{apouranb,tnguyen,thien}@cs.uoregon.edu`

Abstract

The goal of Event Argument Extraction (EAE) is to find the role of each entity mention for a given event trigger word. It has been shown in the previous works that the syntactic structures of the sentences are helpful for the deep learning models for EAE. However, a major problem in such prior works is that they fail to exploit the semantic structures of the sentences to induce effective representations for EAE. Consequently, in this work, we propose a novel model for EAE that exploits both syntactic and semantic structures of the sentences with the Graph Transformer Networks (GTNs) to learn more effective sentence structures for EAE. In addition, we introduce a novel inductive bias based on information bottleneck to improve generalization of the EAE models. Extensive experiments are performed to demonstrate the benefits of the proposed model, leading to state-of-the-art performance for EAE on standard datasets.

1 Introduction

Event Extraction (EE) is an important task of Information Extraction that aims to recognize events and their arguments in text. In the literature, EE is often divided into two sub-tasks: (1) Event Detection (ED) to detect the event trigger words, and (2) Event Argument Extraction (EAE) to identity the event arguments and their roles for the given event triggers. In recent years, ED has been studied extensively with deep learning while EAE is relatively less explored (Wang et al., 2019b). As EAE is necessary to accomplish EE and helpful for many downstream applications (Yang et al., 2003; Cheng and Erk, 2018), further studies are required to improve the performance of EAE. This work focuses on EAE to meet this requirement for EE.

The current state-of-the-art methods for EAE have involved deep learning models that compute an abstract representation vector for each word in the input sentences based on the information from the other context words. The representation vectors for the words are then aggregated to perform EAE (Chen et al., 2015; Nguyen et al., 2016). Our main motivation in this work is to exploit different structures in the input sentences to improve the representation vectors for the words in the deep learning models for EAE. In this work, a sentence structure (or view) refers to an importance score matrix whose cells quantify the contribution of a context word for the representation vector computation of the current word for EAE. In particular, we consider two types of sentence structures in this work, i.e., syntactic and semantic structures. As such, the importance score for a pair of words in the syntactic structures is determined by the syntactic connections of the words in the dependency parsing trees while the contextual semantics of the words in the input sentences are exploited to compute the importance scores in the semantic structures. Consider the following sentence as an example:

Iraqi Press constantly report interviews with **<u>Hussain Molem</u>**, *the* **Hanif Bashir's son-in-law**, *while US officials confirmed* **all Bashir's family members** *were* **<u>killed</u>** *last week.*

In this sentence, an EAE system should be able to realize the entity mention "*Hussain Molem*" as the *Victim* of the *Attack* event triggered by "*killed*". As "*Hussain Molem*" and "*killed*" are far away from each other in the sentence as well as its dependency tree, the EAE models might find it challenging to make the correct prediction in this case. In order for the models to be successful in this case, our intuition is that the models should first rely on the direct connections between "*killed*" and "*all Bashir's family members*" in the dependency tree to capture the role of "*all Bashir's family members*" in the representation vectors for "*killed*". Afterward, the models can rely on the close semantic similarity

between "*all Bashir's family members*" and "*the Hanif Bashir's son-in-law*" to further connect "*the Hanif Bashir's son-in-law*" to "*killed*" so the role information of "*the Hanif Bashir's son-in-law*" can be recorded in the representation vector for "*killed*". Finally, the direct apposition relation between "*the Hanif Bashir's son-in-law*" and "*Hussain Molem*" can be exploited to connect "*Hussain Molem*" with "*killed*" to obtain the necessary representations to perform argument prediction for "*Hussain Molem*". On the one hand, this example suggests that both syntactic and semantic structures are necessary for the EAE models. On the other hand, the example also hints that it is not enough to apply the syntactic and semantic structures separately. In fact, these structures should be explicitly combined to complement each other on identifying important context words to obtain effective representations for EAE. To our knowledge, this is the first work to explore syntactic and semantic structures for EAE.

How should we combine the syntactic and semantic structures to aid the learning of effective representations for EAE? In this work, we propose to employ Graph Transformer Networks (GTN) (Yun et al., 2019) to perform the syntax-semantic merging for EAE. GTNs facilitate the combination of multiple input structures via two steps. The first step obtains the weighted sums of the input structures, serving as the intermediate structures that are able to capture the information from different input perspectives (i.e., structures). In the second step, the intermediate structures are multiplied to generate the final structures whose goal is to leverage the multi-hop paths/connections between a pair of nodes/words (i.e., involving the other words) to compute the importance score for the final structures. As the multi-hop paths with heterogeneous types of connections along the way (i.e., syntax or semantic) has been illustrated to be helpful in our running example (i.e., between '*Hussain Molem*' and '*killed*'), we expect that GTNs can help to combine the syntactic and semantic structures to produce effective representations for EAE.

Finally, in order to further boost the performance for EAE, we propose a novel inductive bias for the proposed GTN model, aiming to improve the generalization of GTNs using the Information Bottleneck idea (Tishby et al., 2000; Belghazi et al., 2018). In particular, the use of the rich combined structures from syntax and semantics might augment GTNs with high capacity to encode the detailed information in the input sentences. Coupled with the generally small training datasets for EAE, the GTN models could learn to preserve all the context information in the input sentences, including the irrelevant information for EAE. This likely leads to the overfitting of GTNs on the training data. In order to overcome this issue, we propose to treat the GTN model in this work as an information bottleneck in which the produced representations of GTNs are trained to not only achieve good prediction performance for EAE but also minimize the mutual information with the input sentences (Belghazi et al., 2018). To this end, we introduce the mutual information between the generated representations of GTNs and the input sentences as an additional term in the overall loss function to improve the generalization of GTNs for EAE. Our extensive experiments on two benchmark datasets for EAE show that the proposed model can achieve the state-of-the-art performance for EAE.

2 Related Work

EAE is one of the two subtasks in EE (the other one is ED) that has been approached early by the feature-based models (Ahn, 2006; Ji and Grishman, 2008; Patwardhan and Riloff, 2009; Liao and Grishman, 2010a,b; Riedel and McCallum, 2011; Hong et al., 2011; McClosky et al., 2011; Li et al., 2013; Miwa et al., 2014; Yang and Mitchell, 2016). The recent work on EE has focused on deep learning to improve the models' performance (Chen et al., 2015; Sha et al., 2018; Zhang et al., 2019; Yang et al., 2019; Nguyen and Nguyen, 2019; Zhang et al., 2020). Among the two subtasks of EE, while ED has been studied extensively by the recent deep learning work (Nguyen and Grishman, 2015; Chen et al., 2015; Nguyen et al., 2016g; Chen et al., 2017; Liu et al., 2017, 2018a; Zhao et al., 2018; Wang et al., 2019a; Lai et al., 2020c), EAE has been relatively less explored. The closest work to ours is (Wang et al., 2019b) that focuses on EAE and exploits the concept hierarchy of event argument roles to perform the task. Our work differs from (Wang et al., 2019b) in that we employ the syntactic and semantic structures of the sentences to better learn the representations for EAE. We also note some new directions on EE based on zero-shot learning (Huang et al., 2018), few-shot learning (Lai et al., 2020a,b) and multimodal learning (Zhang et al., 2017).

3 Model

EAE can be formulated as a multi-class classification problem in which the input involves a sentence $W = w_1, w_2, \ldots, w_N$ (w_i is the i-th word/token in the sentence of length N), and an argument candidate and event trigger at indexes a and t in the sentence (i.e., the words w_a and w_e) respectively. The goal in this problem is to predict the role that the argument candidate w_a plays in the event triggered by w_e. Note that the set of the possible roles also include a special type *None* to indicate that the argument candidate is not an actual argument of the event (i.e., no roles). In order to achieve a fair comparison, following the prior work on EAE (Wang et al., 2019b), we take as inputs the event triggers that are detected by an independent model (i.e., the model in (Wang et al., 2019a)) and separate from our proposed model. We also consider each entity mention in the sentence as a candidate argument for the role prediction task. Our model for EAE in this work involves four major components: (i) sentence encoding, (ii) structure generation, (iii) structure combination, and (iv) model regularization as described in the following.

3.1 Sentence Encoding

To represent the sentence, we encode each word w_i with a real-valued vector x_i that is the concatenation of the two following vectors: (i) the embeddings of the relative distances of the word w_i to the argument candidate (i.e., $i - a$) and event trigger (i.e., $i - e$) (these embeddings are initialized randomly and updated during training), and (ii) the BERT embedding of the word w_i. In particular, to achieve a fair comparison with (Wang et al., 2019b), we run the BERT base cased model (Devlin et al., 2019) over W and use the hidden vector for the first wordpiece of w_i in the last layer of BERT as the embedding vector (of 768 dimensions) for w_i.

The word encoding step then produces a sequence of vectors $X = x_1, \ldots, x_N$ to represent the input sentence W. In order to better combine the BERT embeddings and the relative distance embeddings, we further feed X into a Bidirectional Long-short Term Memory network (BiLSTM), resulting in the hidden vector sequence $H = h_1, \ldots, h_N$ as the representation vectors for the next steps.

3.2 Structure Generation

As presented in the introduction, the motivation for our EAE model is to employ the sentence structures to guide the computation of effective representation vectors for EAE with deep learning. These sentence structures would involve the score matrices of size $N \times N$ in which the score at the (i, j) cell is expected to capture the importance of the contextual information from w_j with respect to the representation vector computation of w_i in the deep learning models for EAE (called the importance score for the pair (w_i, w_j)). In this work, we consider two types of sentence structures for EAE, i.e., the syntactic structures and the semantic structures.

Syntactic Structures: It has been shown in the prior work that the dependency relations in the dependency trees can help to connect a word to its important context words to obtain effective representation vectors for EAE (Sha et al., 2018). To this end, we use the adjacency matrix A^d of the dependency tree for W as one of the syntactic structures for EAE in this work. Note that A^d here is a binary matrix whose cell (i, j) is only set to 1 if w_i and w_j are linked in the dependency tree for W.

One problem with the A^d structure is that it is agnostic to the argument candidate w_a and event trigger w_e for our EAE task. As the argument candidate and event trigger are the most important words in EAE, we argue that the sentence structures should be customized for those words to produce more effective representation vectors in W for EAE. In order to obtain the task-specific syntactic structures for EAE, our intuition is that the closer words to the argument candidate w_a and event trigger w_e in the dependency tree would be more informative to reveal the contextual semantics of w_a and w_e than the farther ones in W (Nguyen and Grishman, 2018). These syntactic neighboring words of w_a and w_e should thus be assigned with higher importance scores in the sentence structures, serving as the mechanism to tailor the syntactic structures for the argument candidate and event trigger for EAE in this work. Consequently, besides the general structure A^d, we propose to generate two additional customized syntactic structures for EAE based on the lengths of the paths between w_a, w_e and the other words in the dependency tree of W for EAE (i.e., one for the argument candidate and one for the event trigger). In particular, for the argument candidate w_a, we first compute the length d_i^a of the path from w_a to every other word w_i in W. The length d_i^a is then converted to an embedding vector $\hat{d}_i^a$ by looking up a length embedding table D (initialized ran-

domly and updated during training): $\hat{d}_i^a = D[d_i^a]$. Afterward, we generate the argument-specific syntactic structure $A^a = \{s_{i,j}^a\}_{i,j=1..N}$ via: $s_{i,j}^a = sigmoid(FF([\hat{d}_i^a, \hat{d}_j^a, \hat{d}_i^a \odot \hat{d}_j^a, |\hat{d}_i^a - \hat{d}_j^a|, \hat{d}_i^a + \hat{d}_j^a]))$ where $[]$ is the vector concatenation, $\odot$ is the element-wise multiplication, and FF is a two-layer feed-forward network to convert a vector to a scalar. We expect that learning the syntactic structures in this way would introduce the flexibility to infer effective importance scores for EAE.

The same procedure can then be applied to generate the trigger-specific syntactic structure $A^e = \{s_{i,j}^e\}_{i,j=1..N}$ for the event trigger w_e for the EAE model in this work. Finally, the general structure A^d and the task-specific syntactic structures A^a and A^e would be used as the syntactic structures for our structure combination component in the next step.

Semantic Structure The semantic structure aims to learn the importance score for a pair of words (w_i, w_j) by exploiting the contextual semantics of w_i and w_j in the sentence. As mentioned in the introduction, the semantic structure is expected to provide complementary information to the syntactic structures, that once combined, can lead to effective representation vectors for EAE. In particular, we employ the Bi-LSTM hidden vectors $H = h_1, \ldots, h_N$ in the sentence encoding section to capture the contextual semantics of the words for the semantic structure in this work. The semantic importance scores $s_{i,j}^s$ for the semantic structure $A^s = \{s_{i,j}^s\}_{i,j=1..N}$ can then be learned via $s_{i,j}^s = f(h_i, h_j)$ where f is some function to produce a score for h_i and h_j. Motivated by the self-attention scores in (Vaswani et al., 2017), we use the following key and query-based function for the f function for $s_{i,j}^s$:

$$k_i = U_k h_i, q_i = U_q h_i$$
$$s_{i,j}^s = \exp(k_i q_j) / \sum_{v=1..N} \exp(k_i q_v) \quad (1)$$

where U_k and U_q are trainable weight matrices, and the biases are omitted in this work for brevity.

Similar to the general syntactic structure A^d, a problem for this function is that the semantic scores $s_{i,j}^s$ are not aware of the argument candidate and the event trigger words, the two important words for EAE. To this end, we propose to involve the contextual semantics of the argument candidate w_a and event trigger w_e (i.e., h_a and h_t) in the computation of the semantic structure score $s_{i,j}^s$ for EAE using:

$$c^k = \sigma(V_k[h_a, h_t]), k_i' = c^k \odot k_i$$
$$c^q = \sigma(V_q[h_a, h_t]), q_i' = c^q \odot q_i \quad (2)$$
$$s_{i,j}^s = \exp(k_i' q_j') / \sum_{v=1..N} \exp(k_i' q_v')$$

The rationale in this formula is to use the hidden vectors for the argument candidate w_a and event trigger w_e to generate the task-specific control vectors c^k and c^q. These control vectors are then employed to filter the information in the key and query vectors (i.e., k_i and q_i) so only the relevant information about w_a and w_e is preserved in k_i and q_i via the element-wise products $\odot$. The resulting key and query vectors (i.e., k_i' and q_i') would then be utilized to compute the task-specific importance score $s_{i,j}^s$ for the semantic structure A^s in this work.

3.3 Structure Combination

The four initial structures in $\mathcal{A} = [A^d, A^a, A^e, A^s]$ can be interpreted as four different types of relations between the pairs of words in W (i.e., using the syntactic and semantic information) (called the word relation types). The cell (i, j) in each initial structure is deemed to capture the degree of connection between w_i and w_j based on their direct interaction/edge (i.e., the one-hop path (w_i, w_j)) and the corresponding relation type for this structure. Given this interpretation, this component seeks to combine the four initial structures in $\mathcal{A}$ to obtain richer sentence structures for EAE. On the one hand, we expect the importance scores between a pair of words (w_i, w_j) in the combined structures to be able to condition on the possible interactions between w_i, w_j and the other words in the sentence (i.e., the multi-hop paths between w_i and w_j that involve the other words). On the other hand, the multi-hop paths between w_i and w_j for the importance scores should also be able to accommodate the direct edges/connections between the words of different relation types (i.e., the heterogeneous edge types). Note that both the multi-hop paths and the heterogeneous edge types along the paths (i.e., for syntax and semantics) have been demonstrated to be helpful for EAE in the introduction. Consequently, in this work, we propose to apply Graph Transformer Networks (GTNs) (Yun et al., 2019) to simultaneously achieve these two goals for EAE.

In particular, following (Yun et al., 2019), we first add the identity matrix I (of size $N \times N$) into the set of structures in $\mathcal{A}$ to enable GTNs to

3654

learn the multi-hop paths at different lengths, i.e., $\mathcal{A} = [A^d, A^a, A^e, A^s, I] = [\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4, \mathcal{A}_5]$. Given the initial structures in $\mathcal{A}$ and inspired by the transformers in (Vaswani et al., 2017), the GTN model is organized into C channels for which the i-the channel involves M intermediate structures $Q_1^i, Q_2^i, \ldots, Q_M^i$ of size $N \times N$ ($1 \leq i \leq C$) (i.e., corresponding to $M - 1$ layers in GTNs). In GTNs, each intermediate structure Q_j^i is computed via a weighted sum of the individual input structures with learnable weights $\alpha_{j,v}^i$: $Q_j^i = \sum_{v=1..5} \alpha_{j,v}^i \mathcal{A}_v$. The weighted sums enable the intermediate structures to reason with any of the four initial word relation types depending on the context, thus offering the structure flexibility for the model. Afterward, in order to capture the multi-hop paths for the importance scores in the i-th channel, the intermediate structures are multiplied to obtain a single sentence structure Q^i for this channel: $Q^i = Q_1^i \times Q_2^i \times \ldots \times Q_M^i$ (called the final structures). It has been shown in (Yun et al., 2019) that Q^i is able to model any multi-hop paths between the words with lengths up to M. Such multi-hop paths can also host heterogeneous word relation types in the edges (due to the flexibility of the intermediate structures Q_j^i), thus introducing rich sentence structures for our EAE problem.

In the next step, the final structures $Q^1, Q^2, \ldots, Q^C$ of GTN are treated as different adjacency matrices for the fully connected graph between the words in W. These matrices would then be consumed by a Graph Convolutional Network (GCN) model (Kipf and Welling, 2017) to produce more abstract representation vectors for the words in our EAE problem. In particular, the GCN model in this work consists of several layers (i.e., G layers in our case) to compute the representation vectors at different abstract levels for the words in W. For the k-th final structure Q^k, the representation vector for the word w_i in the t-th GCN layer would be computed via:

$$\bar{h}_i^{k,t} = ReLU(U^t \sum_{j=1..N} \frac{Q_{i,j}^k \bar{h}_j^{k,t-1}}{\sum_{u=1..N} Q_{i,u}^k}) \quad (3)$$

where U^t is the weight matrix for the t-th GCN layer and the input vectors $\hat{h}_i^{k,0}$ for the GCN model are obtained from the Bi-LSTM hidden vectors (i.e., $\hat{h}_i^{k,0} = h_i$ for all $1 \leq k \leq C, 1 \leq i \leq N$).

Given the outputs from the GCN model, the hidden vectors in the last GCN layer (i.e., the G-th layer) of the word w_i for all the final struc-

tures (i.e., $\bar{h}_i^{1,G}, \bar{h}_i^{2,G}, \ldots, \bar{h}_i^{C,G}$) are then concatenated to form the final representation vector h_i' for w_i in the proposed GTN model: $h_i' = [\bar{h}_i^{1,G}, \bar{h}_i^{2,G}, \ldots, \bar{h}_i^{C,G}]$. Finally, in order to predict the argument role for w_a and w_e in W, we assemble a representation vector R based on the hidden vectors for w_a and w_e from the GCN model via: $R = [h_a', h_e', MaxPool(h_1', h_2', \ldots, h_N')]$. This vector is then sent to a two-layer feed-forward network with softmax in the end to produce a probability distribution $P(.|W, a, t)$ over the possible argument roles. We would then optimize the negative log-likelihood L_{pred} to train the model in this work: $\mathcal{L}_{pred} = -P(y|W, a, t)$ where y is the golden argument role for the input example.

3.4 Model Regularization

As presented in the introduction, the high representation learning capacity of the GTN model could lead to memorizing the information that is only specific to the training data (i.e., overfitting). In order to improve the generalization, we propose to regularize the representation vectors obtained by GTN so only the effective information for EAE is preserved in the GTN representations for argument prediction and the nuisance information of the training data (i.e., the irrelevant one for EAE) can be avoided. To this end, we propose to treat the GTN model as an Information Bottleneck (IB) (Tishby et al., 2000) in which the GTN-produced representation vectors $H' = h_1', h_2', \ldots, h_N'$ would be trained to both (1) retain the effective information to perform argument prediction for EAE (i.e., the high prediction capacity) and (2) maintain a small Mutual Information (MI)[1] with the representation vectors of the words from the earlier layers of the model (i.e., the minimality of the representations) (Belghazi et al., 2018). In this work, we follow the common practice to accomplish the high prediction capacity by using the GTN representation vectors to predict the argument roles and minimizing the induced negative log-likelihood in the training phase. However, for the minimality of the representations, we propose to achieve this by explicitly minimizing the MI between the GTN-produced vectors $H' = h_1', h_2', \ldots, h_N'$ and the BiLSTM hidden vectors $H = h_1, h_2, \ldots, h_N$ from sentence encoding. By encouraging a small MI between H and H', we expect that only the relevant information for EAE

---

[1]In information theory, MI evaluates how much information we know about one random variable if the value of another variable is revealed.

in H is passed through the GTN bottleneck to be retained in H' for better generalization.

As H and H' are sequences of vectors, we first transform them into the summarized vectors h and h' (respectively) to facilitate the MI estimation via the max-pooling function: $h = MaxPool(h_1, h_2, \ldots, h_N)$ and $h' = MaxPool(h'_1, h'_2, \ldots, h'_N)$. Afterward, we seek to compute the MI between h and h' and include it in the overall loss function for minimization. However, as h and h' are high-dimensional vectors, their MI estimation is prohibitively expensive in this case. To this end, we propose to apply the mutual information neural estimation (MINE) method in (Belghazi et al., 2018) to approximate the MI with its lower bound. In particular, motivated by (Hjelm et al., 2019), we propose to further approximate the lower bound of the MI between h and h' via the adversarial approach using the loss function of a variable discriminator. As the MI between h and h' is defined as the KL divergence between the joint and marginal distributions of these two variables, the discriminator aims to differentiate the vectors that are sampled from the joint distribution and those from the product of the marginal distributions for h and h'. In our case, we sample from the joint distribution for h and h' by simply concatenating the two vectors (i.e., $[h', h]$) and treat it as the positive example. To obtain the sample from the product of the marginal distribution, we concatenate h' with $\hat{h}$ that is the aggregated vector (via max-pooling) of the BiLSTM hidden vectors for another sentence (obtained from the same batch with the current sentence during training) (i.e., $[h', \hat{h}]$ as the negative example). These positive and negative examples are then fed into a two-layer feed-forward network D (i.e., the discriminator) to produce a scalar score, serving as the probability to perform a binary classification for the variables. Afterward, the logistic loss $\mathcal{L}_{disc}$ of D is proposed as an estimation of the MI between h and h' and added into the overall loss function for minimization: $\mathcal{L}_{disc} = \log(1 + e^{(1 - D([h', h]))}) + \log(1 + e^{D([h', \hat{h}])})$.

Finally, the overall loss function $\mathcal{L}$ to train the model in this work would be: $\mathcal{L} = \mathcal{L}_{pred} + \alpha_{disc}\mathcal{L}_{disc}$ where α_{disc} is a trade-off parameter.

4 Experiments

Datasets & Parameters: We evaluate the models on two benchmark datasets, i.e., ACE 2005 and TAC KBP 2016 (Ellis et al., 2016). ACE 2005 is a widely used EE dataset, involving 599 documents, 33 event subtypes and 35 argument roles. We use the same data split with the prior work (Chen et al., 2015; Wang et al., 2019b) for a fair comparison (i.e., 40 documents for the test data, 30 other documents for the development set, and the remaining 529 documents for the training data). For TAC KBP 2016, as no training data is provided, following (Wang et al., 2019b), we use the ACE 2005 training data to train the models and then evaluate them on the TAC KBP 2016 test data. To evaluate the models' performance, for a fair comparison with the previous work (Chen et al., 2015; Wang et al., 2019b), we consider an argument classification as correct if its predicted event subtype, offsets and argument role match the golden data.

We fine-tune the hyper-parameters for our model on the ACE 2005 development set, leading to the following values: 30 dimensions for the relative distance and length embeddings (i.e., D), 200 hidden units for the feed-forward network, BiLSTM and GCN layers, 2 layers for the BiLSTM and GCN modules ($G = 2$), $C = 3$ channels for GTN with $M = 3$ intermediate structures in each channel, and 0.1 for the parameter α_{disc}. Finally, besides BERT, we also evaluate the proposed model when BERT is replaced by the `word2vec` embeddings (of 300 dimensions) (Mikolov et al., 2013) to make it comparable with some prior works. Note that as in (Wang et al., 2019b), the proposed model with BERT takes as inputs the predicted event triggers from the BERT-based ED model in (Wang et al., 2019a) while the proposed model with `word2vec` utilizes the predicted event triggers from the `word2vec`-based ED model in (Chen et al., 2015) for compatibility.

Comparison with the State of the Art: To evaluate the effectiveness of the proposed model (called **SemSynGTN**), we first compare it with the baselines on the ACE 2005 dataset. Following (Wang et al., 2019b), we use the following baselines in our experiments: (i) the feature-based models (i.e., **Li's Joint** (Li et al., 2013) and **RBPB** (Sha et al., 2016)), (ii) the deep sequence-based models that run over the sequential order of the words in the sentences (i.e., **DMCNN** (Chen et al., 2015), **JRNN** (Nguyen et al., 2016), **PLMEE** (Yang et al., 2019), and **DMBERT** (i.e., DMCNN with BERT) (Wang et al., 2019b)), (iii) the deep structure-based models that employ dependency trees for BiLSTM or GCNs (i.e., **dbRNN** (Sha et al., 2018) and **JMEE**

(Liu et al., 2018b)), (iv) the models with Generative Adversarial Imitation Learning (**GAIL** (Zhang et al., 2019a)), and (v) the deep learning model that exploits the hierarchical concept correlation among argument roles (i.e., **HMEAE** (Wang et al., 2019b)). **HMEAE** is a BERT-based model with the current best EAE performance on ACE 2005.

Model	P	R	F1
Li's joint (Li et al., 2013)	64.7	44.4	52.7
RBPB (Sha et al., 2016)	54.1	53.5	53.8
DMCNN (Chen et al., 2015)	62.2	46.9	53.5
JRNN (Nguyen et al., 2016)	54.2	56.7	55.4
dbRNN (Sha et al., 2018)	66.2	52.8	58.7
GAIL (Zhang et al., 2019a)	-	-	59.7
JMEE (Liu et al., 2018b)	66.8	54.9	60.3
SemSynGTN (ours)	68.4	55.4	**61.2**
PLMEE (Yang et al., 2019)*	62.3	54.2	58.0
DMBERT (Wang et al., 2019b)*	58.8	55.8	57.2
HMEAE (Wang et al., 2019b)*	62.2	56.6	59.3
SemSynGTN (BERT) (ours)*	69.3	55.9	**61.9**

Table 1: EAE Performance on the ACE 2005 test set. * indicates the models that use BERT.

Table 1 presents the performance of the models on the ACE 2005 test set. Note that we distinguish between the models that employ BERT for the pre-trained word embeddings and those that do not for a clear comparison in the table. The most important observation is that the proposed model SemSynGTN significantly outperforms all the baseline models (with $p < 0.01$) no matter if BERT is used as the pre-traind word embeddings or not. SemSynGTN achieves the state-of-the-art performance on ACE 2005 when BERT is applied in the model, thus demonstrating the benefits of the proposed model with the syntactic and semantic structure combination for EAE in this work.

In order to further demonstrate the effectiveness of the proposed model, following the previous work (Wang et al., 2019b), we evaluate the models on the TAC KBP 2016 dataset. In particular, we compare SemSynGTN with the top four systems in the TAC KBP 2016 evaluation (Dubbin et al., 2016; Hsi et al., 2016; Ferguson et al., 2016), the DMCNN model in (Chen et al., 2015), and the DMBERT and HMEAE models in (Wang et al., 2019b). Note that HMEAE is also the state-of-the-art model for EAE on this dataset. The results are shown in Table 2 that corroborates our findings from Table 1. Specifically, SemSynGTN significantly outperforms all the baseline models with large margins ($p < 0.01$) (whether BERT is used or not), thus confirming the

advantages of the SemSynGTN model in this work.

Model	P	R	F1
DISCERN-R (Dubbin et al., 2016)	7.9	7.4	7.7
Washington4 (Ferguson et al., 2016)	32.1	5.0	8.7
CMU CS Event 1 (Hsi et al., 2016)	31.2	4.9	8.4
Washington1 (Ferguson et al., 2016)	26.5	6.8	10.8
DMCNN (Chen et al., 2015)	17.9	16.0	16.9
SemSynGTN (ours)	39.4	15.3	**22.0**
DMBERT (Wang et al., 2019b)*	22.6	24.7	23.6
HMEAE (Wang et al., 2019b)*	24.8	25.4	25.1
SemSynGTN (BERT) (ours)*	51.1	19.8	**28.5**

Table 2: Performance on the TAC KBP 2016 dataset. * indicates the models that use BERT.

Ablation Study: This part analyzes the effectiveness of the components in the proposed model for EAE by removing each of them from the overall model and evaluating the performance of the remaining models on the ACE 2005 development dataset. In particular, the first major component in SemSynGTN involves structure customization that seeks to tailor the initial syntactic and semantic structures for the argument candidate and event trigger. We evaluate two ablated models for this component: (i) eliminating the task-specific syntactic customization from SemSynGTN that amounts to excluding the customized syntactic structures A^a and A^e from the initial structure set $\mathcal{A}$ (called **SemSynGTN - SynCustom**), and (ii) removing the task-specific semantic customization from SemSynGTN that leads to the use of the simple key-query version (i.e., Equation 1) to compute the importance scores in the semantic structure A^s (i.e., instead of using Equation 2 as in the full model) (called **SemSynGTN - SemCustom**).

The second major component is structure combination that aims to generate the mixed structures from the initial structures in $\mathcal{A}$ via GTN. We consider two ablated models for this component: (i) completely excluding the GTN model and directly applying the GCN model on the initial structures in $\mathcal{A}$ (called **SemSynGTN - GTN**) (the representation vectors from the last GCN layer for the same word with different initial structures are also concatenated in this case), and (ii) keeping the intermediate structures in the channels of the GTN model, but avoiding the intermediate structure multiplications for multi-hop paths in each channel of GTN (called **SemSynGTN - Multi-hop**) (so the GCN model is directly applied over the intermediate structures whose outputs are also concatenated).

Finally, the third component corresponds to the

regularization loss based on information bottleneck L_{disc} in Section 3.4. The removal of $\mathcal{L}_{disc}$ from the overall loss $\mathcal{L}$ leads to the ablated model **SemSynGTN - IB**. As this component relies on the MI between the hidden vectors returned by the BiLSTM and GTN models, we evaluate another variant for SemSynGTN in this case where the regularization loss is eliminated, but the hidden vectors from the BiLSTM model $h_1, h_2, \ldots, h_N$ are included in the final representation vector R for argument role prediction (i.e., $R = [h_a, h_e, h, h'_a, h'_e, h']$) (called **SemSynGTN - IB + LSTM in** R). Table 3 provides the performance of the ablated models on the ACE 2005 development set.

Model	P	R	F1
SemSynGTN	70.2	57.2	**63.0**
SemSynGTN - GTN	66.0	55.5	60.3
SemSynGTN - Multi-hop	68.1	55.2	61.0
SemSynGTN - SynCustom	62.5	57.5	59.9
SemSynGTN - SemCustom	64.9	58.1	61.3
SemSynGTN - IB	68.2	56.3	61.7
SemSynGTN - IB + LSTM in R	67.5	55.8	61.1

Table 3: The ablation study on the ACE 2005 dev set.

The table clearly shows that all the components are necessary for SemSynGTN to achieve the highest performance. In particular, the GTN model, the syntactic and semantic structure customization, and the structure multiplication are all important as eliminating any of them would hurt the performance significantly. These evidences highlight the importance of combining the customized sentence structures for EAE in this work. In addition, "SemSynGTN - Bottleneck" and "SemSynGTN - IB + LSTM in R" are also significantly worse than SemSynGTN, suggesting the effectiveness of information bottleneck to regularize the model for better generalization for GTNs in EAE.

Structure Analysis: The proposed model generates four initial sentence structures in $\mathcal{A}$ (i.e., A^d, A^a, A^e, and A^s) to capture the general and task-specific structures for EAE based on the syntactic and semantic information. In order to evaluate their contribution for SemSynGTN, Table 4 presents the performance of the remaining models when each of these structures is eliminated from the model (i.e., from $\mathcal{A}$). It is clear from the table that the model performance is significantly worse when we remove any of the initial structures in $\mathcal{A}$, thus demonstrating the benefits of such structures for EAE.

Information Bottleneck Analysis: In order to

Model	P	R	F1
SemSynGTN	**70.2**	**57.2**	**63.0**
SemSynGTN - A^d	68.3	56.4	61.8
SemSynGTN - A^a	65.2	57.1	60.9
SemSynGTN - A^e	64.6	56.9	60.5
SemSynGTN - A^s	65.2	55.9	60.2

Table 4: The structure contribution on the ACE 2005 dev set.

prevent overfitting for the GTN model in this work, we propose to cast GTN as an information bottleneck that seeks to minimize the mutual information between the GTN-produced vectors $H' = h'_1, h'_2, \ldots, h'_N$ and the representation vectors of the words from the earlier layers of the model (i.e., prior to GTN for the minimality of the representations). In particular, in the implementation of this idea, we propose to achieve the minimality of the representations by minimizing the mutual information between the vectors in H' and the BiLSTM hidden vectors $H = h_1, h_2, \ldots, h_N$ in the sentence encoding. However, there are other prior layers of GTN whose hidden vectors can also be used for this MI minimization, including (1) the BERT-generated vectors for the words in the input sentence (i.e., $E = e_1, \ldots, e_N$ where e_i is the hidden vector of the first wordpiece of w_i in the last layer of the BERT model), and (2) the input vectors $X = x_1, \ldots, x_N$ for the BiLSTM layer in the sentence encoding where x_i is the concatenation of e_i and the relative distance embeddings for w_i toward w_a and w_e. In this analysis, we aim to evaluate the performance of SemSynGTN when the BiLSTM vectors H are replaced by the vectors in E and X in the computation of $\mathcal{L}_{disc}$ for the MI minimization. Table 5 presents the performance of these variants of SemSynGTN on the ACE 2005 development dataset.

Model	P	R	F1
SemSynGTN with H for MI (proposed)	70.2	57.2	**63.0**
SemSynGTN with X for MI (BERT + distance)	69.1	56.6	62.2
SemSynGTN with E for MI (BERT)	67.4	55.7	61.0

Table 5: Performance of the models on the ACE 2005 development set using different configurations for MI in the information bottleneck.

The first observation from the table is that SemSynGTN with the vectors in X for MI performs better than those with the BERT-generated vectors E. We attribute this to the fact that in addition to the BERT-generated vectors in E, the vectors in X also include the relative distance embeddings

of the words (i.e., for the argument candidate and trigger word). In principle, this makes X more compatible with H' than E as both X and H' have access to the relative distances of the words to capture the positions of the argument candidate and trigger word in the sentences. Such compatible nature of information sources enables more meaningful comparison between X and H' for the MI minimization, providing more effective training signals to improve the representation vectors for EAE. More importantly, we see that the proposed MI minimization mechanism between H' and H helps SemSynGTN to achieve significantly better performance than those with the other variants (i.e., with X or E for the MI). This clearly helps to justify our proposal of employing the BiLSTM hidden vectors H to compute $\mathcal{L}_{disc}$ in this work. In fact, the advantage of H over X for the MI minimization demonstrates the benefits of the BiLSTM layer to better combine the BERT-generated vectors e_i and the relative distance embeddings for w_i in x_i to generate effective hidden vectors for the MI-based comparisons with H' for EAE.

Performance Analysis: To understand how the proposed model improves the performance over the baselines, we examine the outputs of SemSynGTN and the two major baseline models, i.e., (1) HMEAE (Wang et al., 2019b), the most related work that ignores the syntactic and semantic structures and previously has the best BERT-based performance for EAE, and (2) SemSynGTN - GTN that considers the syntactic and semantic structures but does not model their interactions to capture multi-hop paths with GTN. Our investigation suggests that while SemSynGTN outperforms HMEAE and SemSynGTN - GTN in general, the performance gaps between the models become substantially larger for the sentences with large numbers of words (i.e., distances) between the argument candidates and event triggers (called $\#bw$). In particular, Table 6 presents the performance of the three models on two subsets of the ACE 2005 development set, i.e., one with $\#bw \leq 10$ and one with $\#bw > 10$. As we can see, the performance gaps between SemSynGTN and the two baselines on the subset with $\#bw > 10$ are much larger than those with $\#bw \leq 10$. We attribute this better performance of SemSynGTN to its abilities to employ the combined structures based on syntax and semantic, and to model the multi-hop paths between words to compute the importance scores in the fi-

nal structures of GTN. These abilities essentially enable SemSynGTN to capture longer and more flexible paths between words to compute effective representations for EAE. SemSynGCN is then able to perform better for the sentences with large $\#bw$ where encoding more context words is necessary to achieve high performance.

Model	$\#bw \leq 10$	$\#bw > 10$
SemSynGTN	**64.1**	**61.3**
SemSynGTN - GTN	61.8	55.6
HMEAE (2019b)	60.9	53.3

Table 6: F1 scores of the models on the ACE 2005 dev set.

5 Conclusion

We propose a novel deep learning model for EAE that combines the syntactic and semantic structures of the input sentences for effective representation learning. The proposed model introduces Graph Transformer Networks and Graph Convolutional Networks to effectively perform the structure combination. A novel inductive bias is presented to improve the model generalization based on information bottleneck. The extensive experiments demonstrate the effectiveness of the proposed model. In the future, we plan to apply the proposed model to other related tasks in information extraction such as relation extraction.

Acknowledgement

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*.

Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. 2018. Mutual information neural estimation. In *ICML*.

Yubo Chen, Shulin Liu, Xiang Zhang, Kang Liu, and Jun Zhao. 2017. Automatically labeled data generation for large scale event extraction. In *ACL*.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *ACL-IJCNLP*.

Pengxiang Cheng and Katrin Erk. 2018. Implicit argument prediction with event knowledge. In *NAACL-HLT*.

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

Greg Dubbin, Archna Bhatia, Bonnie J Dorr, Adam Dalton, Kristy Hollingshead, Suriya Kandaswamy, Ian Perera, and Jena D Hwang. 2016. Improving discern with deep learning. In *Text Analysis Conference (TAC)*.

Joe Ellis, Jeremy Getman, Neil Kuster, Zhiyi Song, Ann Bies, and Stephanie Strassel. 2016. Overview of linguistic resources for the tac kbp 2016 evaluations: Methodologies and results. In *Text Analysis Conference (TAC)*.

James Ferguson, Colin Lockard, Natalie Hawkins, Stephen Soderland, Hannaneh Hajishirzi, and Daniel S Weld. 2016. University of washington tac-kbp 2016 system description. In *Text Analysis Conference (TAC)*.

Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. 2019. Learning deep representations by mutual information estimation and maximization. In *ICLR*.

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

Andrew Hsi, Jaime Carbonell, and Yiming Yang. 2016. Cmu cs event tac-kbp2016 event argument extraction system. In *Text Analysis Conference (TAC)*.

Lifu Huang, Heng Ji, Kyunghyun Cho, and Clare R. Voss. 2018. Zero-shot transfer learning for event extraction. In *ACL*.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *ACL*.

Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

Viet Dac Lai, Franck Dernoncourt, and Thien Huu Nguyen. 2020a. Exploiting the matching information in the support set for few shot event classification. In *Proceedings of the 24th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*.

Viet Dac Lai, Franck Dernoncourt, and Thien Huu Nguyen. 2020b. Extensively matching for few-shot learning event detection. In *Proceedings of the 1st Joint Workshop on Narrative Understanding, Storylines, and Events (NUSE) at ACL 2020*.

Viet Dac Lai, Tuan Ngo Nguyen, and Thien Huu Nguyen. 2020c. Event detection: Gate diversity and syntactic importance scores for graph convolution neural networks. In *EMNLP*.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *ACL*.

Shasha Liao and Ralph Grishman. 2010a. Filtered ranking for bootstrapping in event extraction. In *COLING*.

Shasha Liao and Ralph Grishman. 2010b. Using document level cross-event inference to improve event extraction. In *ACL*.

Jian Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2018a. Event detection via gated multilingual attention mechanism. In *AAAI*.

Shulin Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2017. Exploiting argument information to improve event detection via supervised attention mechanisms. In *ACL*.

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

David McClosky, Mihai Surdeanu, and Christopher Manning. 2011. Event extraction as dependency parsing. In *BioNLP Shared Task Workshop*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *ICLR*.

Makoto Miwa, Paul Thompson, Ioannis Korkontzelos, and Sophia Ananiadou. 2014. Comparable study of event extraction in newswire and biomedical domains. In *COLING*.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *NAACL-HLT*.

Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *ACL-IJCNLP*.

Thien Huu Nguyen and Ralph Grishman. 2018. Graph convolutional networks with argument-aware pooling for event detection. In *AAAI*.

Thien Huu Nguyen, Adam Meyers, and Ralph Grishman. 2016g. New york university 2016 system for kbp event nugget: A deep learning approach. In *Proceedings of Text Analysis Conference (TAC)*.

Trung Minh Nguyen and Thien Huu Nguyen. 2019. One for all: Neural joint modeling of entities and events. In *AAAI*.

Siddharth Patwardhan and Ellen Riloff. 2009. A unified model of phrasal and sentential evidence for information extraction. In *EMNLP*.

Sebastian Riedel and Andrew McCallum. 2011. Robust biomedical event extraction with dual decomposition and minimal domain adaptation. In *BioNLP Shared Task 2011 Workshop*.

Lei Sha, Jing Liu, Chin-Yew Lin, Sujian Li, Baobao Chang, and Zhifang Sui. 2016. Rbpb: Regularization-based pattern balancing method for event extraction. In *ACL*.

Lei Sha, 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*.

Naftali Tishby, Fernando C Pereira, and William Bialek. 2000. The information bottleneck method. In *arXiv preprint physics/0004057*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurNIPS*.

Xiaozhi Wang, Xu Han, Zhiyuan Liu, Maosong Sun, and Peng Li. 2019a. Adversarial training for weakly supervised event detection. In *NAACL-HLT*.

Xiaozhi Wang, Ziqi Wang, Xu Han, Zhiyuan Liu, Juanzi Li, Peng Li, Maosong Sun, Jie Zhou, and Xiang Ren. 2019b. Hmeae: Hierarchical modular event argument extraction. In *EMNLP-IJCNLP*.

Bishan Yang and Tom M. Mitchell. 2016. Joint extraction of events and entities within a document context. In *NAACL-HLT*.

Hui Yang, Tat-Seng Chua, Shuguang Wang, and Chun-Keat Koh. 2003. Structured use of external knowledge for event-based open domain question answering. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*.

Sen Yang, Dawei Feng, Linbo Qiao, Zhigang Kan, and Dongsheng Li. 2019. Exploring pre-trained language models for event extraction and generation. In *ACL*.

Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. 2019. Graph transformer networks. In *NeurIPS*.

Junchi Zhang, Yanxia Qin, Yue Zhang, Mengchi Liu, and Donghong Ji. 2019. Extracting entities and events as a single task using a transition-based neural model. In *IJCAI*.

Tongtao Zhang, Heng Ji, and Avirup Sil. 2019a. End-to-end entity and event extraction with generative adversarial imitation learning. In *Data Intelligence Vol 1 (2): 99-120*.

Tongtao Zhang, Spencer Whitehead, Hanwang Zhang, Hongzhi Li, Joseph Ellis, Lifu Huang, Wei Liu, Heng Ji, and Shih-Fu Chang. 2017. Improving event extraction via multimodal integration. In *MM*.

Yunyan Zhang, Guangluan Xu, Yang Wang, Daoyu Lin, Feng Li, Chenglong Wu, Jingyuan Zhang, and Tinglei Huang. 2020. A question answering-based framework for one-step event argument extraction. In *IEEE Access, vol 8, 65420-65431*.

Yue Zhao, Xiaolong Jin, Yuanzhuo Wang, and Xueqi Cheng. 2018. Document embedding enhanced event detection with hierarchical and supervised attention. In *ACL*.