CorED: Incorporating Type-level and Instance-level Correlations for Fine-grained Event Detection

Jiawei Sheng
Institute of Information Engineering, Chinese Academy of Sciences
School of Cyber Security, UCAS
shengjiawei@iie.ac.cn

Rui Sun
School of Computer Science and Engineering, Beihang University
sunrui@act.buaa.edu.cn

Shu Guo
National Computer Network Emergency Response Technical Team/Coordination Center of China
guoshu@cert.org.cn

Shiyao Cui, Jiangxia Cao
Institute of Information Engineering, Chinese Academy of Sciences
School of Cyber Security, UCAS
{cuishiyao,caojiangxia}@iie.ac.cn

Lihong Wang
National Computer Network Emergency Response Technical Team/Coordination Center of China
wlh@isc.org.cn

Tingwen Liu, Hongbo Xu
Institute of Information Engineering, Chinese Academy of Sciences
School of Cyber Security, UCAS
{liutingwen,hbxu}@iie.ac.cn

ABSTRACT

Event detection (ED) is a pivotal task for information retrieval, which aims at identifying event triggers and classifying them into pre-defined event types. In real-world applications, events are usually annotated with numerous fine-grained types, which often arises long-tail type nature and co-occurrence event nature. Existing studies explore the event correlations without full utilization, which may limit the capability of event detection. This paper simultaneously incorporates both the type-level and instance-level event correlations, and proposes a novel framework, termed as CorED. Specifically, we devise an adaptive graph-based type encoder to capture instance-level correlations, learning type representations not only from their training data but also from their relevant types, thus leading to more informative type representations especially for the low-resource types. Besides, we devise an instance interactive decoder to capture instance-level correlations, which predicts event instance types conditioned on the contextual typed event instances, leveraging co-occurrence events as remarkable evidence in prediction. We conduct experiments on two public benchmarks, MAVEN and ACE-2005 dataset. Empirical results demonstrate the unity of both type-level and instance-level correlations, and the model achieves effectiveness performance on both benchmarks.

CCS CONCEPTS

• **Computing methodologies** → **Information extraction**; • **Information systems** → *Content analysis and feature selection.*

KEYWORDS

Event detection, Event correlation, Information extraction

ACM Reference Format:
Jiawei Sheng, Rui Sun, Shu Guo, Shiyao Cui, Jiangxia Cao, Lihong Wang, Tingwen Liu and Hongbo Xu. 2022. CorED: Incorporating Type-level and Instance-level Correlations for Fine-grained Event Detection. In *Proceedings of the 45th Int'l ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '22), July 11–15, 2022, Madrid, Spain.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3477495.3531956

SIGIR '22, July 11–15, 2022, Madrid, Spain.

ACM ISBN 978-1-4503-8732-3/22/07.
https://doi.org/10.1145/3477495.3531956

1 INTRODUCTION

Retrieving events from texts is pivotal to various natural language processing applications [23], such as automatic question answering [34] and dialogue systems [33], and the first task to perform is event detection (ED). Generally, ED aims at detecting events in texts by identifying event triggers (the words or phrases evoking events in texts) and then classifying them into pre-defined event types. For example, in Figure 1(a), an ED model should recognize that the word "killing" is a trigger evoking a `Kill` type. There are several candidate event instances (a.k.a candidate triggers) within the sentence, thus the ED model ought to recognize all the true instances from them and identify their types. In real-world applications, events are usually annotated with numerous fine-grained types, making the task more challenging and informative. Such a fine-grained task is usually termed as fine-grained event detection [40].

Early ED studies exploit feature-based [1, 2, 15, 21] and neural-based [5, 12, 27, 31] methods, mostly regarding multiple event instances in one sentence as independent ones and recognize them separately. However, event instances can have interactive correlations within the sentence, which provide valuable feature for event detection. As shown in Figure 1(a), there are 3 true event instances with 6 candidate instances, from which we can observe explicit correlations between event instance "hit" and "killing". According to statistical analysis, there are nearly 30% sentences in ACE-2005 dataset [10] and 68% sentences in MAVEN dataset [40] having more than one event instance. Therefore, it is beneficial to exploit the instance correlations for event detection.

Recent efforts on ED mostly explore the instance correlations in textual-aspect, which can be roughly manifested into two groups: 1) dependency-based neural methods [8, 25, 28, 43], which elaborate syntactic dependency features upon the textual encoders, building implicit instance correlations along multi-hop syntactic shortcuts; 2) collection-based neural methods [6, 29, 30], which predict event types by collecting previous event instance representations, building explicit instance correlations in decoding process. However,

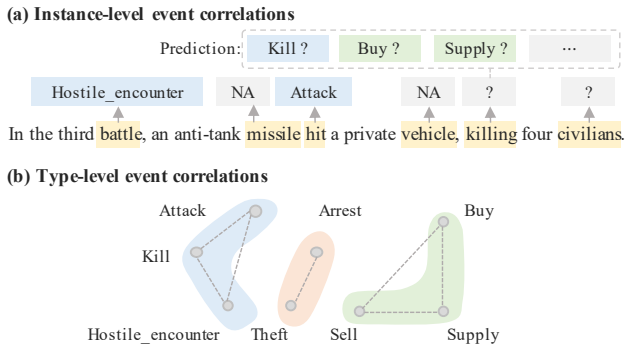


Figure 1: Illustration of instance-level and type-level event correlations discussed in this paper. Note that the blue, red and green shade marks event types with violence-, legality- and commerce-related meaning, respectively. The yellow shade marks the candidate instances within the sentence.

all these methods mostly leverage instance correlations in textual-aspect, neglecting the type-aspect instance correlations. Also in Figure 1(a), there exist redundant candidate instances evoking no event types (belonging to `NA` type), which may be intractable to discern valuable contextual instances for target instance prediction. Intuitively, the contextual instance types provide remarkable evidence for target instance type reasoning, such as an `Attack` type instance usually leading to a `Kill` type instance, naturally excluding the less valuable `NA` type candidate instances. We term such instance correlations in both textual-aspect and type-aspect as **instance-level event correlations** in this paper.

In addition, there also exist valuable **type-level event correlations**, which provide valuable prior knowledge for type representations, usually benefitting to instance-level event correlation incorporation and instance type prediction. Generally, the semantic relevance among pre-defined event types reflects type correlations. As shown in Figure 1(b), `Arrest` type is more related with `Theft` type than `Buy` type, since both `Arrest` type and `Theft` type have similar legality-related semantic meaning. We argue that it is crucial to build type-level correlations for fine-grained event detection. Concretely, fine-grained events often have long-tail issue in data annotation [40], where numerous event types only have few training instances to learn informative type representations. There are 28.6% types (48 in 168 types) having less than 100 training instances in the fine-grained MAVEN dataset. It would be beneficial to leverage type correlations beyond training data to learn informative type representations, especially for those low-resource event types. To the best of our knowledge, few studies have exploited the type-level correlations for event detection.

To tackle the above issues, we propose an Event Detection framework simultaneously incorporating both the type-level and instance-level event Correlations, termed as CorED. Specifically, we devise an adaptive graph-based type encoder to capture type-level correlations, which adaptively learns type correlations deployed as graph structure, and then refines type representations with graph neural networks. Such an encoder allows type representations access to valuable features from both training data and other related types, leading to more informative type representations especially for the low-resource event types. Besides, we devise an instance interactive decoder to capture instance-level correlations, which predicts event types explicitly conditioned on the other contextual instances with their types in the sentence. Such a decoder treats contextual instances with types as remarkable evidence, benefitting to the task for the sentence having multiple events. We conduct experiments on the challenging MAVEN dataset (168 types) and the well-studied ACE-2005 dataset (33 types) to demonstrate effectiveness of our model. Our contributions can be summarized as follows:

- We propose a novel framework[1] for fine-grained ED, which incorporates both the instance-level and type-level event correlations. In our knowledge, this paper is the first paper to simultaneously consider both correlations for ED task.
- We technically devise an adaptive graph-based type encoder to incorporate type-level correlations, and an instance interactive decoder to incorporate instance-level correlations.
- Experimental results indicate the unity of incorporating both type-level and instance-level correlations, and achieving effectiveness performance on two public benchmarks.

2 RELATED WORK

Event detection is a well-studied task with research effort [5, 7, 22, 30, 35, 36, 45] in the last decade. Early feature-based methods [1, 2, 15, 21] leverage linguistic features and manually designed feature for the task, usually suffering from cascading errors. Recent studies devise neural-based methods [5, 12, 27, 31, 45] with sophisticated neural structures to automatically extract contextual features, where several studies explore event instance correlation to improve performance: 1) the dependency-based neural methods [8, 25, 28, 43] elaborate syntactic dependency features upon the textual encoders, which building implicit instance correlations along multi-hop dependency shortcuts. For example, Cui et al.[2020] introduce syntactic labels in the dependency tree to enhance information flow, and adopts graph neural networks to leverage syntactic information for event detection. 2) the collection-based neural methods [6, 29, 30] predict events by collecting previous event instance representations, building explicit instance correlations in decoding process. As a typical method, Chen et al.[2018] propose a hierarchical tagging LSTM layer and tagging attention mechanism to model the event correlations within a sentence. Lou et al.[2021] further adopt Seq2seq manner to generate type sequence, which collects sequential dependency in generation process. Though these methods achieve promising performance, they mostly implicitly leverage the instance correlations, or neglect the instance correlations in the type-aspect. Besides, to the best of our knowledge, few studies exploit the type-level correlations for event detection. In contrast, this paper explicitly leverages both the type-level and instance-level correlations to improve event detection, especially for those low-resource event types.

Recent years have seen studies [3, 11, 24, 44] exploring other type information to improve performance for event detection. A promising way is to adopt machine reading comprehension paradigm [3, 11, 19, 26], which treats type description as questions,

[1]Our source code is available at https://github.com/JiaweiSheng/CorED.

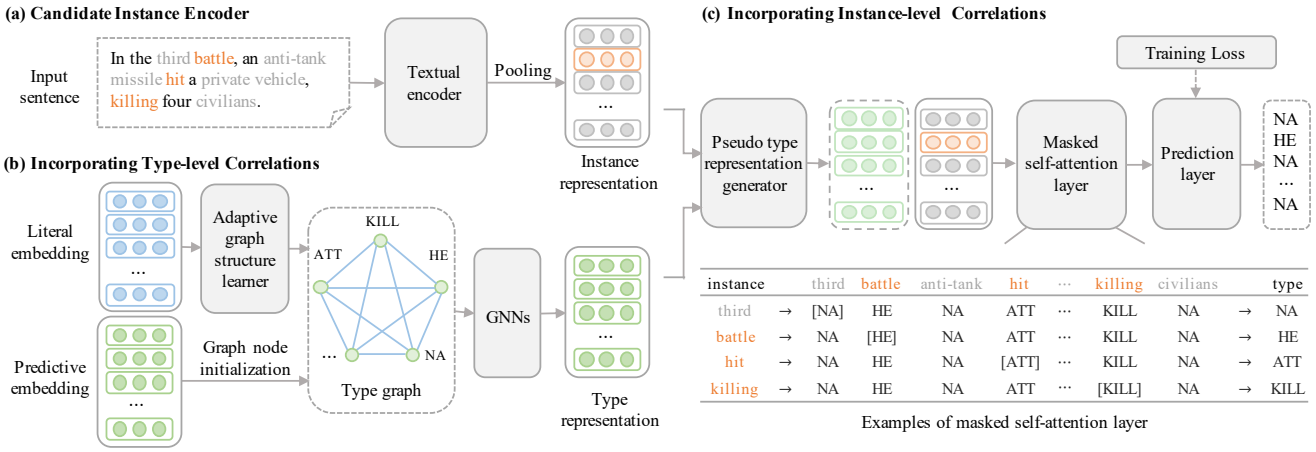


Figure 2: The overview of our proposed model, CorED. Part (a) encodes the input sentence, generating candidate instance (trigger) representations. Part (b) incorporates type-level correlations, generating type representations. Part (c) incorporates instance-level correlations, making type predictions for candidate instances. Note that `ATT`, `KILL`, and `HE` denote `Attack`, `Killing`, and `Hostile_encounter` type, respectively. The types surrounded by square brackets are replaced by symbol `[MASK]`.

and makes predictions with enriched type information. However, these works requires external information for event detection, and neglect the type relevance correlations. Actually, there exist methods [20, 42] exploring type correlations outside ED. They mostly attempt type correlations in fine-grained entity typing as multi-class classification, learning better type representation with manually constructed graph. Nevertheless, these methods mostly adopt static type graph, and thus inevitably introduce noise in graph construction. Besides, ED has quite different task definition, involving more complicated type or instance correlations among events.

3 PRELIMINARIES

Following previous ED studies [5, 30, 31, 40], we introduce the task definition in this paper:

Task Definition. Given a sentence s with several candidate triggers $\mathcal{C}_s$, our goal is to identify the true trigger $x \in \mathcal{C}_s$ and identify its event type $t \in \mathcal{T}$. Here $\mathcal{T}$ is the pre-defined event type set with $|\mathcal{T}| = N$. For those false triggers with no events, we classify them into an additional `NA` type, leading to $N + 1$ types.

Note that the candidate triggers are usually discovered with POS (part of speech) tags, such as nouns, verbs, adjectives, and adverbs. In this paper, we adopt the candidate triggers provided by the benchmark [40] with official evaluation, where the true event triggers are annotated within the candidate triggers.

4 METHODOLOGY

This section introduces our proposed model, CorED. The model contains a candidate instance encoder to generate candidate instance (i.e., candidate trigger) representations, an adaptive graph-based type encoder to generate type representations with type-level correlations, and an instance interactive decoder to predict event types with instance-level correlations. Figure 2 demonstrates the framework of our proposed model.

4.1 Candidate Instance Encoder

This section derives candidate instance representations based on the textual encoder, treating all the candidate triggers as candidate instances (by adding an `NA` type for the false triggers).

To derive token representations, we adopt LSTM/BERT as the textual encoder. BERT [9] is a bidirectional language representation model based on transformer architecture [37], which generates textual representations conditioned on token context and remains rich textual information. Formally, the sentence s has B tokens and M candidate triggers. We input the tokens into the textual encoder, and obtain hidden states $\boldsymbol{H} = [\boldsymbol{h}_1, \boldsymbol{h}_2, \cdots, \boldsymbol{h}_B]$ as the token representations. Then, we derive each candidate instance representation $\boldsymbol{x}_i$ with pooling operation upon the token representations over its candidate trigger span $[\text{sta}_i, \text{end}_i]$:

$$\boldsymbol{x}_i = \rho(\boldsymbol{W}\,\text{Pooling}(\boldsymbol{H}[\text{sta}_i : \text{end}_i])) \tag{1}$$

Here we adopt max-pooling operation. $\boldsymbol{W} \in \mathbb{R}^{d \times d}$ is the learnable parameter, d is the token representation dimension, and ρ is the ReLU activation function. For the sentence, the candidate instance encoder sequentially encodes each candidate instance, and outputs the sequential instance representations as $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_M]$.

4.2 Incorporating Type-level Correlations

As claimed in the introduction, event types can have semantic relevance correlations before fitting on the training instances. Thereby, this section leverages the literal semantics of event types with pre-trained word embeddings to generate the underlying type graph. Then, we refine another group of task predictive embeddings with graph neural networks upon the type graph, preserving type correlations as the prior knowledge for instance type prediction. The details are shown in Figure 2(b).

4.2.1 Adaptive type graph structure learner. Since event types are all written in textual phrases, we derive literal semantics of event

types, namely *type literal embeddings*, with off-the-shelf word embeddings such as GloVE [32]. Specifically, we simply take average[2] on the token embeddings within type literal name as the type embedding, and initialize a full-zero vector as the NA type embedding. The derived type literal embeddings are denoted as $\boldsymbol{V} \in \mathbb{R}^{(N+1)\times d'}$, where d' is the dimension of the word embedding. Then, we measure the similarity score between each of two type literal embeddings with a similarity function ϕ. Generally, ϕ can be achieved by cosine or Euclidean distance. Inspired by Chen et al.[2020b], we achieve ϕ with a parameterized metric function, namely weighted cosine, for robust similarity measurement:

$$\phi(v_i, v_j) = \frac{1}{K}\sum_{k=1}^{K} \cos(\boldsymbol{w}_k \odot \boldsymbol{v}_i, \boldsymbol{w}_k \odot \boldsymbol{v}_j) \tag{2}$$

where $\odot$ is the element-wise multiplication. $\boldsymbol{w}_k \in \mathbb{R}^{d'}$ is a non-negative learnable weight vector, used to highlight the important dimensions of the embedding $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$. Note that we keep type literal embeddings learnable, and adopt K heads of the weighted cosine to capture different aspects of embeddings. The final similarity score is obtained by taking average on the K cosine scores, grasping different aspects of the type literal embeddings.

Based on the similarity scores between types, we have built a fully-connected graph. Generally, many underlying graph structures are much more sparse than a fully connected graph [4], which is not only computationally expensive but also might introduce noise (i.e., redundant edges). Therefore, we adopt a simple yet effective ϵ-neighbor strategy [4] to prune edges, by only selecting the edges with larger similarity scores:

$$\mathbf{E}_{ij} = \begin{cases} \phi(v_i, v_j), & \text{if } \phi(v_i, v_j) > \epsilon \\ -\inf, & \text{otherwise} \end{cases} \tag{3}$$

where ϵ is a hyperparameter threshold. In this way, we have built undirected weighted type graph reflecting correlations of literal semantics. Besides, though the graph structure is initially decided by the type literal embeddings, it is then dynamically tuned by optimizing these learnable embeddings to fit the ED task.

4.2.2 Graph neural networks for type representation. Since the learned type graph structure reflects correlations of event types, we are going to refine type representations upon the graph structure. Notice that the type literal embeddings (from GloVE) may not be predictive on the instance representations (from LSTM/BERT) for event detection, we randomly initialize another embeddings $\boldsymbol{T}^{(0)} \in \mathbb{R}^{(N+1)\times d}$, termed as *type predictive embeddings*, to preserve the task predictive information.

Then, we adopt graph neural networks (GNNs) [17, 38] to refine the type predictive embeddings, which are usually adopted to transfer valuable information between correlative nodes [20, 42]. Specifically, we first normalize the edge weights with softmax normalization, and then stack L-layer graph convolutional layer [17]:

$$\begin{aligned} \mathbf{A} &= \text{softmax}(\mathbf{E}) \\ \boldsymbol{T}^{(l+1)} &= \rho\left(\boldsymbol{A}\boldsymbol{T}^{(l)}\boldsymbol{W}^{(l)}\right) \end{aligned} \tag{4}$$

where $\boldsymbol{W}^{(l)} \in \mathbb{R}^{d\times d}$ is the graph convolutional kernel, and ρ denotes ReLU activation. We adopt the same adjacent matrix through all the layers. Note that this design is inspired by GAT [38], but the adjacent matrix is generated from the learnable type literal embeddings, and keeps the graph structure dynamically learnable.

In this way, we refine type predictive embeddings upon the graph structure with type relevance correlations, thus deriving informative knowledge from related neighbors aside from fitting on the training instance. The final **type representation** $\boldsymbol{T}$ is obtained from the L-th layer output, omitting the superscript for simplicity.

4.3 Incorporating Instance-level Correlations

As claimed in the introduction, for any given instance (i.e., candidate trigger) in sentence, its contextual instances with types provide valuable evidence in prediction. Therefore, we propose an instance interactive decoder for prediction considering instance correlations. The instance interactive decoder borrows idea from masked language model (MLM) [9], which can be interpreted as a denoising auto-encoding mechanism [9, 46]. Generally, MLM aims to reconstruct corrupted input tokens (replaced by a learnable symbol [MASK] as noise) conditioned on the contextual tokens, which intrinsically models token correlations within contexts. In this paper, we take advantage of this property, and predict the target instance type with the [MASK] symbol conditioned on the contextual instances (including their texts and types), thus naturally model the instance-level correlations in both textual-aspect and type-aspect.

For better illustration, we here briefly introduce the overall process. Note that we have obtained the instance representation $\boldsymbol{X}$ (from §4.1) and the type representation $\boldsymbol{T}$ (from §4.2). The overall process can be summarized as follows:

$$\hat{\boldsymbol{Y}} = \text{PseudoTypeGenerator}(\boldsymbol{X}, \boldsymbol{T}), \tag{5}$$

$$\boldsymbol{Z} = \text{MaskedSelfAttentionLayer}(\boldsymbol{X}, \hat{\boldsymbol{Y}}), \tag{6}$$

$$Y = \text{PredictionLayer}(\boldsymbol{Z}), \tag{7}$$

where Eq. (5) generates pseudo type representations for all the instances within the sentence (see §4.3.1). Eq. (6) refines each target instance representation conditioned on its contextual instances, including their instance information and their type information (see §4.3.2). Eq. (7) predicts event types upon the refined instance representations, generating the final predictions (see §4.3.3).

4.3.1 Pseudo type representation generator. To catch the instance correlations, we generate pseudo type representations as the contextual instance type information. Specifically, we measure the type plausibility scores for each instance x_i by comparing the instance representation $\boldsymbol{x}_i$ to the type representation $\boldsymbol{t}_j$. Then, we aggregate all the type representations according to the plausibility scores:

$$\begin{aligned} p(t_j|x_i) &= \frac{\exp(\boldsymbol{x}_i \cdot \boldsymbol{t}_j)}{\sum_{o=1}^{N+1}(\boldsymbol{x}_i \cdot \boldsymbol{t}_o)}, \\ \hat{\boldsymbol{y}}_i &= \sum_{j=1}^{N+1} p(t_j|x_i)\boldsymbol{t}_j, \end{aligned} \tag{8}$$

where $p(t_j|x_i)$ reflects the plausibility of instance x_i occurring type t_j. In this way, the instance type representation aggregates all the possible types in accordance with their plausibility scores, preserving valuable knowledge [13] over event types, leading to robust

[2]This process can be easily extended to type description with other textual representation techniques, and we leave this incremental extension in the future work.

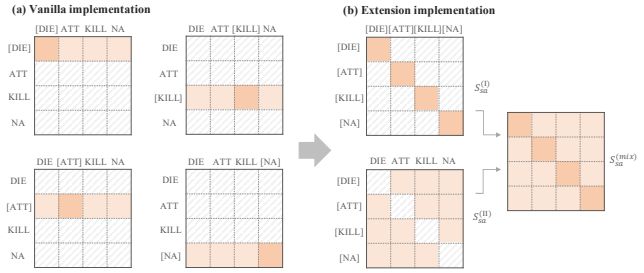


Figure 3: Example of self-attention score matrices. The vertical elements are the target instances x_i, and the horizontal elements are the attended instances x_j. The types surrounded by square brackets are replaced by the symbol [MASK].

instance type representations. The sequential instance type representations for the sequential instance textual representations $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_M]$ is respectively derived as $\hat{\boldsymbol{Y}} = [\hat{\boldsymbol{y}}_1, \hat{\boldsymbol{y}}_2, \cdots, \hat{\boldsymbol{y}}_M]$.

4.3.2 Masked self-attention layer. Based on the sequential instance textual representations and type representations, we will leverage the instance correlations for prediction. Specifically, our main idea is to predict types for each target instance with the contexts of other instance and type information within the sentence. Following this idea, for each target instance x_i, we first replace its type representation by the [MASK] embedding $\boldsymbol{e}_{mask}$, and the corrupted sequential instance type representations $\bar{\boldsymbol{Y}}^{(i)}$ for x_i are obtained as:

$$\begin{aligned} \bar{\boldsymbol{Y}}^{(i)} &= [\bar{\boldsymbol{y}}_1^{(i)}, \bar{\boldsymbol{y}}_2^{(i)}, \cdots, \bar{\boldsymbol{y}}_j^{(i)}, \cdots, \bar{\boldsymbol{y}}_M^{(i)}], \\ \bar{\boldsymbol{y}}_j^{(i)} &= \begin{cases} \boldsymbol{e}_{mask}, & \text{if } j = i \\ \hat{\boldsymbol{y}}_j, & \text{otherwise} \end{cases} \end{aligned} \tag{9}$$

Afterwards, we fuse the corrupted sequential instance type representations with instance textual representations, which captures both the instance and type information of contextual instances. Then, we input them into self-attention layer [37] to build interactive correlations among all the candidate instances:

$$\bar{\boldsymbol{X}}^{(i)} = \bar{\boldsymbol{W}}[\bar{\boldsymbol{Y}}^{(i)}; \boldsymbol{X}], \boldsymbol{Z}^{(i)} = \text{self-attention-layer}(\bar{\boldsymbol{X}}^{(i)}), \tag{10}$$

where $\bar{\boldsymbol{W}} \in \mathbb{R}^{d \times 2d}$ is the linear fusion parameter, and $[;]$ is the element-wise vector concatenation. Here we omit two-layer feed-forward networks [37] in the self-attention layer for illustration simplicity. The obtained instance representation $\boldsymbol{z}_i = \boldsymbol{Z}^{(i)}[i]$, which preserves its contextual instance and type information, is finally used for type prediction (in §4.3.3). Note that it takes M-times self-attention interactions since the instance representation $\boldsymbol{z}_i$ is separately generated conditioned on different corrupted sequential representations. Therefore, we introduce an extension implementation trick for the masked self-attention layer, which keeps the same results but reducing self-attention score computation times from M times to 2 times for sentences with any candidate instance number.

Extension Implementation. As shown in Figure 3(a), there exist redundant computation in self-attention score matrix. For each target instance $\bar{\boldsymbol{x}}_i$, it only utilizes the attention interactions to: 1) the instance itself $\bar{\boldsymbol{x}}_i$ with the [mask] symbol; and 2) the contextual instances $\bar{\boldsymbol{x}}_{j,j\neq i}$ with the non-masked types. Based on this observation, we calculate the attention interactions separately on the instances with the [mask] symbol and the instances with non-masked types. The implementation detail is shown in Figure 3(b).

Specifically, we first calculate the two self-attention score matrices separately utilizing all-masked and all-non-masked sequential instance representations as follows:

$$\bar{\boldsymbol{X}}^{(-)} = \bar{\boldsymbol{W}}[\boldsymbol{E}_{mask}; \boldsymbol{X}], \ \bar{\boldsymbol{X}}^{(+)} = \bar{\boldsymbol{W}}[\hat{\boldsymbol{Y}}; \boldsymbol{X}], \tag{11}$$

$$\begin{aligned} \boldsymbol{S}_{sa}^{(\text{I})} &= \text{self-attention-scores}(\bar{\boldsymbol{X}}^{(-)}, \bar{\boldsymbol{X}}^{(-)}), \\ \boldsymbol{S}_{sa}^{(\text{II})} &= \text{self-attention-scores}(\bar{\boldsymbol{X}}^{(-)}, \bar{\boldsymbol{X}}^{(+)}), \end{aligned} \tag{12}$$

where $\boldsymbol{E}_{mask}$ consists of M repeated $\boldsymbol{e}_{mask}$ vectors, and the self-attention-scores is calculated the same as the original paper [37]:

$$\text{self-attention-scores}(\boldsymbol{Q}, \boldsymbol{K}) = (\boldsymbol{Q}\boldsymbol{W}^{Q})^{T}\boldsymbol{K}\boldsymbol{W}^{K}, \tag{13}$$

where $\boldsymbol{W}^{Q} \in \mathbb{R}^{d \times d}$ and $\boldsymbol{W}^{K} \in \mathbb{R}^{d \times d}$ are learnable parameters.

Then, we generate the mixed self-attention score matrix $\boldsymbol{S}_{sa}^{(\text{mix})}$ by putting the instance-itself attention scores (from $\boldsymbol{S}_{sa}^{(\text{I})}$) on the diagonal and the contextual instance attention scores (from $\boldsymbol{S}_{sa}^{(\text{II})}$) on the non-diagonal in their corresponding positions, and composite the whole score matrix as follows:

$$\begin{aligned} \boldsymbol{S}_{sa}^{(\text{mix})}[i,j] &= \begin{cases} \boldsymbol{S}_{sa}^{(\text{I})}[i,j], & \text{if } j = i \\ \boldsymbol{S}_{sa}^{(\text{II})}[i,j], & \text{otherwise} \end{cases} \\ \boldsymbol{A}_{sa} &= \text{softmax}(\boldsymbol{S}_{sa}^{(\text{mix})}) \end{aligned} \tag{14}$$

Finally, with the mixed self-attention score matrix, we aggregate the sequential instance representations $\bar{\boldsymbol{X}}^{(i)}$ and generate the refined instance representation $\boldsymbol{z}_i$. In this way, we keep the same results but only conduct self-attention score computation from M times to 2 times, improving the computation efficiency.

4.3.3 Prediction Layer. Generally, the softmax function can be utilized for type prediction on each target instance. In order to better discern true event instances from all the candidate instances, we recommend a simple yet effective prediction layer as extension. Specifically, for each target instance x_i, we separately calculate the probability of the NA type and the other task types, and the final predicted probabilities are derived as:

$$p(t_j|x_i) = (1 - p(t_o|x_i))p(t_j|x_i) \tag{15}$$

where t_j denotes task event types except NA, and t_o denotes the NA type. The probabilities for event types and NA type are obtained as:

$$\begin{aligned} p(t_j|x_i) &= \sigma(\boldsymbol{t}_j \cdot \boldsymbol{z}_i) \\ p(t_o|x_i) &= \sigma(\boldsymbol{t}_o \cdot \boldsymbol{z}_i) \end{aligned} \tag{16}$$

where $\boldsymbol{t}_j$ and $\boldsymbol{t}_o$ are representations of the task types and the NA type from $\boldsymbol{T}$, respectively. In this way, the formula provides valuable clues in model training. When the true type is NA type, the model tends to predict higher $p(t_o|x_i)$ as a shortcut, relieving the model to learn all-zero task type probabilities. When the true type is task type, any task type will lead to lower $p(t_o|x_i)$, correcting the model prediction. Empirical results reflect the effectiveness of this design.

For inference strategy, we first select the type with the highest predicted score. If the score is larger than threshold ξ, we return the type as the final prediction, otherwise return the NA type.

4.4 Training Objective

To optimize the overall model, we adopt binary cross-entropy loss on the predictions (in Eq.(15)):

$$\mathcal{L} = -\sum_{s\in\mathcal{D}}\sum_{x_i\in s}\sum_{t_j\in\mathcal{T}} y_{ij}\log p(t_j|x_i) + (1-y_{ij})\log(1-p(t_j|x_i)) \quad (17)$$

where $y_{ij} \in \{0, 1\}$ indicates whether the type t_j holds for instance x_i in training data. We train the model by minimizing $\mathcal{L}$ through Adam stochastic gradient descent [16] over the shuffled mini-batches.

5 EXPERIMENT

This section conducts experiments to evaluate the performance of our proposed model, CorED.

5.1 Dataset and Evaluation Metric

We conduct experiments on MAVEN dataset and ACE-2005 dataset. MAVEN [40] is a newly constructed large-scale fine-grained ED dataset[3], which contains 4, 480 documents with challenging 168 fine-grained event types. We use the official data split 2,913/710/857 documents for training/validation/testing. We also adopt the well-studied ACE-2005 [10] dataset[4] for evaluation, which contains 599 documents with 33 event types. We use the same data split 529/30/40 documents as previous works [40, 41], and preprocess the data with code[5] provided by Wang et al.[2019b]. As previous works [5, 40] recommended, we use each sentence token in ACE-2005 as candidate triggers, and use the provided candidate triggers in MAVEN dataset. Since MAVEN has larger data scale and more fine-grained event types, it can be used for better evaluation on the fine-grained ED. We report the official precision (P), recall (R) and F1 scores for evaluation, among which F1 is the most comprehensive metric.

5.2 Comparision Methods

Most existing ED methods can be roughly manifested into two groups: methods with implicit instance-level event correlation and methods with explicit instance-level event correlation. We adopt the following paradigmatic methods as baselines:

Methods with implicit instance-level event correlation. This group of methods generate event instance representations with contextual features, which can be seen as an implicit way to capture instance-level event correlation. (1) **DMCNN** [5] leverages CNNs to learn textual representations and a dynamic multi-pooling mechanism to aggregate event-related features for classification. (2) **DMBERT** [39] extends DMCNN with pre-trained language representation model BERT [9] and also adopts the dynamic multi-pooling mechanism to aggregate features for classification. (3) **BiLSTM+CRF** [14] adopts LSTM networks to learn textual representations, and adopts the conditional random field (CRF) [18] as the output layer for event detection. (4) **BERT+CRF** extends BiLSTM+CRF with BERT for better textual representations, and also adopts CRF as the output layer. (5) **MOGANED** [43] is a graph neural network (GNN) model, which devises a multi-order graph attention network to effectively model the multi-order syntactic contextual connections in dependency trees. (6) **EE-GCN** [8] further exploits syntactic edge labels in dependency trees, and devises edge-enhanced GNNs to refine textual representations with syntactic relations. In contrast to these methods, our method generates event instance representations by leveraging other event instance representations, thus explicitly capturing the instance-level event correlations.

Methods with explicit instance-level event correlation. This group of methods explore instance collective decoders, which make event instance predictions with explicit contextual instance representations. (1) **HBTNGMA** [6] devises a hierarchical LSTM tagger to decode event instances, collecting contextual event instance representations from both sides. (2) **MLBiNet** [29] formulates ED in a generative Seq2Seq manner, which decodes event instances by generating instance predictions conditioned on the former instance representations, naturally modeling the sequential instance dependency. It also derives bidirectional information by concatenating representations from both directions. In contrast to these methods, our method not only captures contextual instance representations from both sides, but also exploits their contextual instance types for better reasoning in MLM manner.

In contrast to the above baselines, our model also exploits type-level event correlations as prior knowledge to refine event type representations. To the best of our knowledge, few methods consider both the type-level and instance-level correlations for ED.

5.3 Implementation Details

For all the baselines, we adopt the best hyperparameters reported in their literature. For EE-GCN, HBTNGMA and MLBiNet, we reproduce the performance on ACE-2005 for the standard deviation, and conduct experiments on MAVEN data with official implementation. We conduct EE-GCN on BERT textual encoder. We also adopt official Stanford CoreNLP toolkit[6] to generate syntactic labels for those methods utilizing syntactic features. For other baselines, we adopt the source code[7] provided by Wang et al.[2020] and copy the existing results reported in the literature.

We implement BiLSTM and BERT version of our method to evaluate the performance. For both versions, we adopt 300-dimension GloVE embeddings[8] as the type literal embeddings. We simply set head number K to 8, and GNN layer number L to 1 on both datasets. The threshold ϵ is 0.2 and 0.4, and the dropout rate is 0.5 and 0.6 on MAVEN and ACE-2005, respectively. For BERT version, we adopt bert-base-uncased in huggingface[9], whose layer number is 12 and the embedding dimension is 768. The learning rate for BERT parameters is 5e-5 for both datasets, and the learning rate for other parameters is 1e-3. The threshold ξ is 0.7 on both datasets. For LSTM version, we use GloVE embeddings to initialize input word embeddings, and adopt one-layer BiLSTM with dimension 600 on both datasets. The learning rate is 1e-3, and the threshold ξ is 0.6 on both datasets. All the hyperparameter settings are tuned on the validation data by grid search with 5 trials according to F1-score. We release the source code for further extension.

[3] https://github.com/THU-KEG/MAVEN-dataset
[4] https://catalog.ldc.upenn.edu/LDC2006T06
[5] https://github.com/thunlp/HMEAE
[6] http://nlp.stanford.edu/software/stanford-english-corenlp-2018-10-05-models.jar
[7] https://github.com/THU-KEG/MAVEN-dataset/tree/main/baselines
[8] https://nlp.stanford.edu/projects/glove/
[9] https://github.com/huggingface/transformers

Encoder	Methods	MAVEN (168 event types)			ACE-2005 (33 event types)		
		P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)
BiLSTM	DMCNN‡ (2015)	**66.3** ± 0.89	55.9 ± 0.50	60.6 ± 0.20	73.7 ± 2.42	63.3 ± 3.30	68.0 ± 1.95
	BiLSTM+CRF‡ (2015)	63.4 ± 0.70	64.8 ± 0.69	64.1 ± 0.13	**77.2** ± 2.08	74.9 ± 2.62	75.4 ± 1.64
	MOGANED‡ (2019)	63.4 ± 0.88	64.1 ± 0.90	63.8 ± 0.18	70.4 ± 1.38	73.9 ± 2.24	72.1 ± 0.39
	HBTNGMA† (2018)	63.3 ± 1.37	62.6 ± 1.22	62.7 ± 1.46	68.5 ± 2.47	78.4 ± 1.45	72.8 ± 0.87
	MLBiNet† (2021)	63.5 ± 0.57	63.8 ± 0.47	63.6 ± 0.52	73.5 ± 1.25	78.1 ± 0.61	76.5 ± 0.66
	CorED-BiLSTM (Ours)	63.6 ± 0.80	**71.2** ± 0.82	**67.2** ± 0.35	72.6 ± 1.53	**78.8** ± 1.54	**77.5** ± 0.61
BERT	DMBERT‡ (2019a)	62.7 ± 1.01	**72.3** ± 1.03	67.1 ± 0.41	70.2 ± 1.71	78.9 ± 1.64	74.3 ± 0.81
	BERT+CRF‡ (2019)	65.0 ± 0.84	70.9 ± 0.94	67.8 ± 0.15	71.3 ± 1.77	77.1 ± 1.99	74.1 ± 1.56
	EE-GCN† (2020)	64.3 ± 1.67	69.8 ± 1.38	66.9 ± 0.57	76.9 ± 1.72	78.5 ± 1.62	77.2 ± 1.43
	CorED-BERT (Ours)	**68.7** ± 0.64	69.7 ± 0.66	**69.2** ± 0.17	**79.9** ± 1.52	**81.7** ± 1.03	**81.2** ± 0.92

Table 1: Results of event detection on the MAVEN and ACE2005 datasets. The mean and standard deviation of results are reported with 5 trials. † marks results produced with official implementation. ‡ marks results copied from the literature [40].

Encoder	Variants	Micro				Macro			
		P(%)	R(%)	F1(%)	ΔF1	P(%)	R(%)	F1(%)	ΔF1
BiLSTM	complete	64.3	**71.1**	**67.5**	-	61.0	**62.9**	**60.3**	-
	w/o type correlation	67.3	65.7	66.5	↓ 1.0	61.5	58.5	58.7	↓ 1.6
	w/o instance correlation	**70.7**	62.4	66.3	↓ 1.2	65.0	55.9	58.4	↓ 1.9
	w/o both correlation	70.6	62.1	66.1	↓ 1.4	**65.6**	54.3	57.4	↓ 2.9
	repl. sigmoid	69.8	64.6	67.1	↓ 0.4	63.7	55.3	59.2	↓ 1.1
	repl. softmax	65.1	68.3	66.7	↓ 0.8	60.7	62.8	59.3	↓ 1.0
BERT	complete	68.4	70.6	**69.5**	-	65.4	64.2	**62.8**	-
	w/o type correlation	70.0	67.4	68.7	↓ 0.8	65.3	61.9	61.7	↓ 1.1
	w/o instance correlation	69.8	68.1	68.9	↓ 0.6	64.8	63.5	62.2	↓ 0.6
	w/o both correlation	**70.5**	66.0	68.2	↓ 1.3	**67.7**	59.5	61.3	↓ 1.5
	repl. sigmoid	65.3	**73.1**	68.9	↓ 0.6	61.5	**68.9**	62.3	↓ 0.5
	repl. softmax	64.9	**73.1**	68.8	↓ 0.7	60.1	68.0	62.6	↓ 0.2

Table 2: Variant experiments on MAVEN validation data. Bold marks the highest score. "w/o" means removing corresponding module from the complete model. "repl." means replacing corresponding module with the other module.

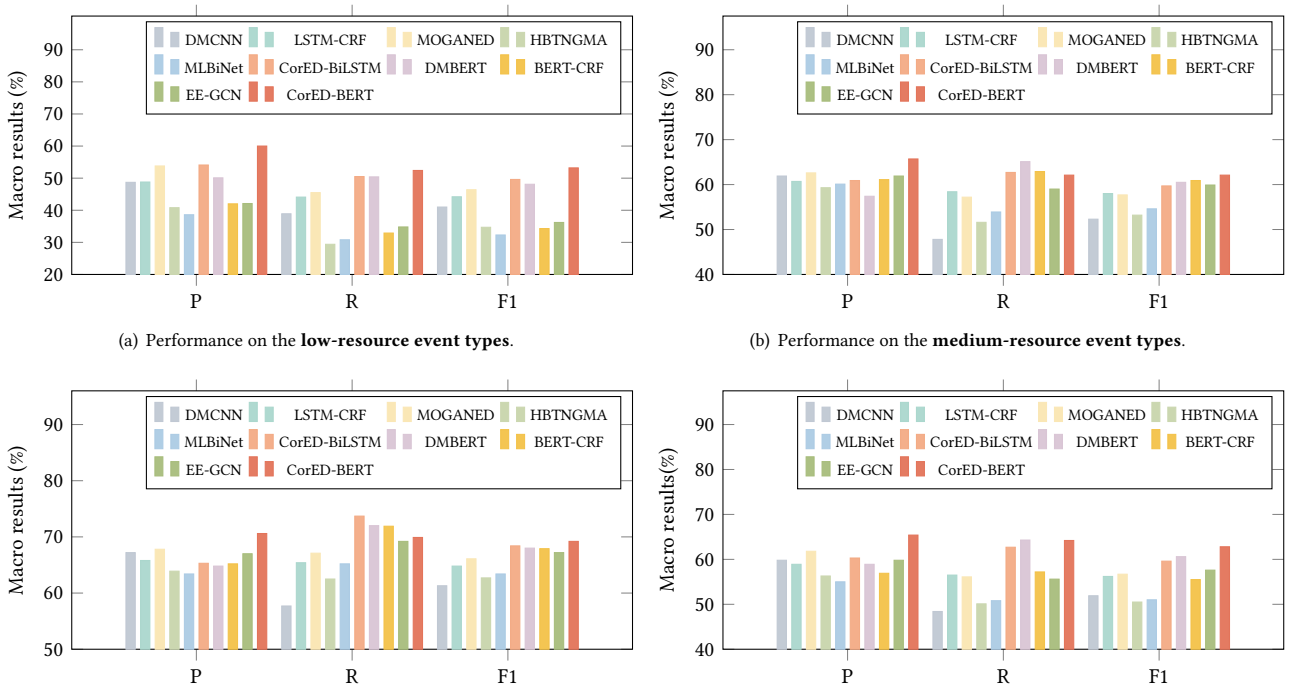
5.4 Main Results

To evaluate the effectiveness of CorED, we report overall results in Table. 1 with mean and standard deviation for exhaustive analysis.

From the table, we can observe that: 1) Our model outperforms all the baselines in both BERT group and BiLSTM group, in terms of F1-score on both datasets. Especially, CorED-BiLSTM improves 3.1 and 1.0 percentages, and CorED-BERT improves 1.4 and 4.0 percentages on MAVEN and ACE-2005 over the best performance baselines, respectively. It demonstrates the effectiveness and superiority of our proposed model, validating the benefit of incorporating both the type-level and instance-level event correlations. 2) Comparing to ACE-2005 dataset, all the baseline methods tend to have lower performance on the MAVEN dataset, whereas our model still achieves robust performance on the MAVEN dataset. Considering MAVEN contains numerous extremely fine-grained event types (168 vs. 33 types for MAVEN vs. ACE-2005), it demonstrates that our model performs well in the extremely fine-grained scenario, reflecting the effectiveness of incorporating type and instance correlations. 3) Comparing CorED-BiLSTM with HBTNGMA and MLBiNet that explicitly consider instance correlations, our model also achieves better performance. Though these methods consider correlations among instances, they generate instance representations without explicit type information, and fuse bidirectional information with simple integration. Besides, they may not be suitable for extremely fine-grained ED since there exist a number of low-resource event types. All the observation demonstrates that our model can achieve effective and reliable performance for the fine-grained ED task.

5.5 Discussion for Model Variants

To investigate the effectiveness of each module in CorED, we conduct variant experiments, showcasing the results in Table. 2. Besides the micro metrics, we also report the macro metrics, which calculate the mean scores of each event type, reflecting the average performance on detailed event types. Note that "w/o" denotes removing the corresponding module from the complete model, "repl. sigmoid" and "repl. softmax" denotes replacing the prediction layer with the simple sigmoid and softmax function, respectively.



(a) Performance on the **low-resource event types**.

(b) Performance on the **medium-resource event types**.

(c) Performance on the **high-resource event types**.

(d) Performance on the **overall event types**.

Figure 4: Detailed results on different event type groups on MAVEN validation data. For event type t with training sample number N_t, we assign the event type t in low-resource group, medium-resource group and high-resource group with $0 < N_t \leq 100$, $100 < N_t \leq 500$ and $N_t > 500$, respectively. The overall group contains all the event types from the three groups.

From the table, we can observe that: 1) The variants without correlations significantly decline on both micro F1 and macro F1, which demonstrates that the type-level correlations and instance-level corrections are both benefit to the fine-grained ED. 2) The impact of instance correlations tends to be more significant on LSTM version than BERT version. We believe the reason is that the powerful BERT encoder actually implicitly captures some clues of instance correlations within textual context. 3) The impact of type correlations tends to be more significant on macro metrics. Since the type correlations build connections among relevant types, the type representations learn effective features from both training instances and relevant types, thus benefiting to the low-resource event types. 4) The devised prediction layer achieves effective performance. By separately considering the `NA` type and event types, the model can better identify true triggers among candidate triggers, thus benefiting to final predictions. All the observations demonstrate the effectiveness of modules in our model.

5.6 Impact of Type-level Correlations

To further investigate the impact of the type-level correlations, we report the results on the detailed event types on MAVEN dataset. According to the training sample number N_t, we divide the event types into three groups[10]: the low-resource type ($0 < N_t \leq 100$, containing 48 types), medium-resource type ($100 < N_t \leq 500$, containing 67 types), and high-resource type ($N_t > 500$, containing 53 types). We also regard the overall event types as an individual group. The results are shown in Figure 4. From the figure, we can observe that: 1) Our model consistently outperforms other baselines on all the event type groups. It demonstrates our model is robust to different scenarios of training resource, achieving reliable performance on event detection. 2) Our model tends to achieve more significant improvements in the low-resource scenarios. The reason we think is that by incorporating the type-level event correlations, the type representations can derive effective features not only from the training instances, but also the other relevant event type representations. The advantages on the low-resource types and the consistent improvements on the medium-resource and high-resource types lead to better results on the overall performance.

To further understand the improvement on different event types, we conduct analysis on the adaptively learned type graph. Figure. 5 showcases example types with their top 5 most similar neighbors. Note that the similarity scores are calculated by the learnable similarity function Eq. 2, and we also report detailed training instance number of each event type. From the figure, we can observe that: 1) For each example type, the neighbor types with higher similarity score tend to reflect higher semantic relevance. Specifically, the type `Attack`'s top 1 neighbor is `Military_operation`, which can be comprehensible by human since they both have violence-related

[10]The phenomenon is similar in another group division of 0-30, 30-300 and 300-.

#Type	#Number	Top 5 of the most similar neighbor types
Attack	2,920	Military_operation (0.743; 1,022), Response (0.740; 299), Change_position (0.725; 903), Escaping (0.715; 764), Killing (0.714; 1,625)
Killing	1,625	Attack (0.715; 2,920), Death (0.713; 1,001), Arrest (0.702; 307), Kidnapping (0.660; 87), Releasing (0.653; 114)
Death	1,001	Cause_included (0.765; 678), Preventing_or_letting (0.758; 772), Arrest (0.757; 307), Change_position (0.756; 903), Coming_to_believe (0.755; 203)
Arrest	307	Releasing (0.856; 114), Being_operation (0.792; 199), Escaping (0.764; 764), Prison (0.759; 55), Death (0.758; 1,001)
Theft	19	Robbery (0.698; 57), Commit_crime (0.670; 224), Criminal_investigation (0.668; 228), Kidnapping (0.638; 87), Arrest (0.597; 307)
Incident	15	Attack (0.643; 2,920), Response (0.617; 299), Change_position (0.616; 903), Statement (0.614; 1,392), Criminal_investigation (0.611; 228)

Figure 5: Details of the learned type graph on MAVEN dataset. # Type is the node type. # Number is the training sample number of the node type. We report top 5 neighbor types, detailed with $(a; b)$, where a denotes the similarity, and b denotes the training sample number of the neighbor type.

semantic meaning. It demonstrates that the type graph learned by the model can effectively reflect the semantic meanings of event types, thus providing promising prior knowledge for type prediction. 2) With the learned type graph, the low-resource types can have connections to the high-resource types, capturing valuable information from the high-resource types. For instance, the top 1 neighbor of the low-resource type `Incident` is `Attack`, which has more training instances to learn effect representations. Such connections between high-resource types and low-resource types helps the low-resource types derive additional features aside from their own training instances. It also illustrates the improvements of our model on the low-resource event types. Benefit from the above properties, we evaluate the utility of our type encoder, and demonstrate the effectiveness of type-level event correlations for fine-grained event detection.

5.7 Impact of Instance-level Correlations

To demonstrate the effectiveness of the instance-level correlations, we evaluate the performance of our model in single event detection (1/1) and multiple event detection (1/n), where 1/1 denotes one sentence that has one event; otherwise, 1/n is used. The experimental results are presented in Table 3. From the table, we can observe that: 1) Both CorED-BERT and CorED-BiLSTM consistently outperform their baselines in 1/1, 1/N and the "all" group. Besides, it also shows that our model achieves more significant improvements on 1/N group than 1/1 group. We believe that our model leverages effective event correlations, thus derive better performance than previous methods. 2) Comparing to the ablated variant, our complete models of both textual encoder version derive better results on the 1/N and "all" group. We notice that our CorED-BiLSTM result slightly decline (↓ 0.3) on the 1/1 group. The reason may be that the instance interactive decoder makes prediction based on the contextual event instances, but for the sentences that only have one event, it can hardly derive additional contextual instance information for prediction. Nevertheless, CorED-BiLSTM achieves significant improvements (↑ 2.6) on 1/N group, and finally leads to better results (↑ 1.4) on the overall data. 3) The improvements on BiLSTM version are more significant on BERT version. The phenomenon also demonstrates that the powerful textual encoder may implicitly leverage the contextual instance information with the representative textual representations. Even though, our design achieves further improvements upon BERT encoder, which demonstrates that explicitly capturing instance correlations help the model easily access to the valuable evidence for prediction.

Data	Methods	1/1 (%)	1/N (%)	all (%)
BiLSTM	DMCNN	57.0	61.6	60.7
	LSTM-CRF	56.9	66.3	64.1
	MOGANED	58.0	67.0	65.6
	HBTNGMA	51.3	61.3	58.9
	MLBiNet	55.8	64.7	62.6
	CorED-BiLSTM w/o IC	**58.2**	67.8	66.3
	CorED-BiLSTM (ours)	57.9	**70.4**	**67.7**
BERT	DMBERT	59.0	70.0	67.6
	BERT-CRF	60.1	70.6	68.4
	EE-GCN	60.4	69.8	67.7
	CorED-BERT w/o IC	61.9	70.6	68.9
	CorED-BERT (ours)	**62.1**	**71.5**	**69.5**

Table 3: Results (F1) on MAVEN validation data, including single event sentences (1/1), multiple event sentences (1/N) and overall sentences. 1/1 means one sentence that has one event, otherwise, 1/N is used. "all" means all the validation data is included. "w/o IC" denotes the ablated variant without the explicit instance-level correlations.

Id	#Trigger	#Type	Sentence
1	killed	Kill	One thousand Knights Hospitaller from Strakonice led[Causation] by Jindřich of Hradec, killed[Mask] in battle[Hostile_encounter], attacked[Attack] war[Hostile_encounter] wagons placed[Placing] on a slim dam, with huge casualties[Catastrophe] but no success.
2	battle	Hostile_encounter	One thousand Knights Hospitaller from Strakonice led[Causation] by Jindřich of Hradec, killed[Kill] in battle[Mask], attacked[Attack] war[Hostile_encounter] wagons placed[Placing] on a slim dam, with huge casualties[Catastrophe] but no success.
3	attacked	Attack	One thousand Knights Hospitaller from Strakonice led[Causation] by Jindřich of Hradec, killed[Kill] in battle[Hostile_encounter], attacked[Mask] war[Hostile_encounter] wagons placed[Placing] on a slim dam, with huge casualties[Catastrophe] but no success.
4	burned	Destroy	Despite this, two Swedish ships were burned[Mask] soon.
5	created	Create	Texxas Jam was created[Mask] by Louis Messina, promoter of Pace Concerts in Houston and David Krebs, manager of the rock acts Ted Nugent and Aerosmith.

Figure 6: Case study of masked self-attention layer on MAVEN validation data. #Trigger is the target instance, #Type is the predicted type. Here the candidate instances are underlined, $[\cdot]$ marks their event types, and we omit the `NA` type of false instances for simplicity. The red, blue and orange color marks candidate instances with attention score $\alpha \geq 1e-1$, $5e-5 \leq \alpha < 1e-1$ and $\alpha < 5e-5$, respectively.

To further understand the inference process, we showcase examples in the masked self-attention layer on CorED-BERT version, and depict the attended degree of contextual instances in Figure 6. In the figure, the red and blue color marks the relatively important contextual candidate instances in prediction, and we notate the true type for each candidate instance for better comprehension. From the figure, we can observe that: 1) The model tends to focus on the relevant candidate instances for target instance prediction in most cases. For example, case 1 aims to predict the target type `Kill`. The mainly attended contextual instance is the candidate trigger "battle" with type `Hostile_encounter`, which provides comprehensible evidence for inference. Besides, the case 1-3 showcase three target instances within the same sentence. Though the three target instances have similar meanings, they tend to attend on different contextual instances, thus leveraging different evidences in detail for different target instance inference. 3) For the sentence only having one event, the model also perceives valuable information from the candidate instance with `NA` type. Specifically, case 4-5 showcase sentences having only one true instance. Interestingly, the model focuses on the false candidate instances relevant to the target instance, which seems to be the event arguments of the target event though MAVEN doesn't provide argument annotations. All the observation indicate that our model can exploit remarkable evidence for target instance prediction within the candidate instances.

6 CONCLUSION

In this paper, we propose a novel event detection framework, CorED, incorporating both the type-level and instance-level event correlations. We devise an adaptive graph-based type encoder to exploit the type correlations, and devise an instance interactive decoder to leverage the instance correlations. With the type-level correlations, the model derives valuable prior knowledge not only from the training data, but also from the other relative event types. With the instance-level correlations, the model makes instance type prediction conditioned on the other contextual instances within the same sentence, perceiving contextual instances as remarkable evidence in prediction. Experimental results demonstrate the unity of both type and instance correlations, and achieve effectiveness performance on the public benchmarks. In the future, we would investigate more advanced techniques to derive type graph structure, and alleviate the impact of the provided candidate triggers.

ACKNOWLEDGMENTS

Corresponding authors: Lihong Wang and Shu Guo. We sincerely thank all the anonymous reviewers for their insightful comments and constructive suggestions. This work is supported by the National Key Research and Development Program of China (Grant No.2021YFB3100600), the National Natural Science Foundation of China (No.62106059), the National Social Science Foundation of China (Grant No.19BSH022), and the Youth Innovation Promotion Association of CAS (Grant No. 2021153).

REFERENCES

[1] David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*. 1–8.
[2] Jun Araki and Teruko Mitamura. 2015. Joint Event Trigger Identification and Event Coreference Resolution with Structured Perceptron. In *Proceedings of EMNLP*. 2074–2080. https://doi.org/10.18653/v1/d15-1247
[3] Yunmo Chen, Tongfei Chen, Seth Ebner, Aaron Steven White, and Benjamin Van Durme. 2020. Reading the Manual: Event Extraction as Definition Comprehension. In *Proceedings of EMNLP*. https://doi.org/10.18653/v1/2020.spnlp-1.9
[4] Yu Chen, Lingfei Wu, and Mohammed J. Zaki. 2020. Iterative Deep Graph Learning for Graph Neural Networks: Better and Robust Node Embeddings. In *Proceedings of NeurIPS*. https://proceedings.neurips.cc/paper/2020/hash/e05c7ba4e087beea9410929698dc41a6-Abstract.html
[5] Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event Extraction via Dynamic Multi-Pooling Convolutional Neural Networks. In *Proceedings of ACL*. https://doi.org/10.3115/v1/p15-1017
[6] Yubo Chen, Hang Yang, Kang Liu, Jun Zhao, and Yantao Jia. 2018. Collective Event Detection via a Hierarchical and Bias Tagging Networks with Gated Multi-level Attention Mechanisms. In *Proceedings of EMNLP*. https://doi.org/10.18653/v1/d18-1158
[7] Xin Cong, Shiyao Cui, Bowen Yu, Tingwen Liu, Yubin Wang, and Bin Wang. 2021. Few-Shot Event Detection with Prototypical Amortized Conditional Random Field. In *Findings of ACL*. 28–40. https://doi.org/10.18653/v1/2021.findings-acl.3
[8] Shiyao Cui, Bowen Yu, Tingwen Liu, Zhenyu Zhang, Xuebin Wang, and Jinqiao Shi. 2020. Edge-Enhanced Graph Convolution Networks for Event Detection with Syntactic Relation. In *Findings of EMNLP*. https://doi.org/10.18653/v1/2020.findings-emnlp.211
[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL*. https://doi.org/10.18653/v1/n19-1423
[10] George R. Doddington, Alexis Mitchell, Mark A. Przybocki, Lance A. Ramshaw, Stephanie M. Strassel, and Ralph M. Weischedel. 2004. The Automatic Content Extraction (ACE) Program - Tasks, Data, and Evaluation. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation*.
[11] Xinya Du and Claire Cardie. 2020. Event Extraction by Answering (Almost) Natural Questions. In *Proceedings of EMNLP*. https://doi.org/10.18653/v1/2020.emnlp-main.49
[12] Xiaocheng Feng, Lifu Huang, Duyu Tang, Heng Ji, Bing Qin, and Ting Liu. 2016. A Language-Independent Neural Network for Event Detection. In *Proceedings of ACL*. https://doi.org/10.18653/v1/p16-2011
[13] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the Knowledge in a Neural Network. *CoRR* abs/1503.02531 (2015). arXiv:1503.02531 http://arxiv.org/abs/1503.02531
[14] Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF Models for Sequence Tagging. *CoRR* abs/1508.01991 (2015). arXiv:1508.01991 http://arxiv.org/abs/1508.01991
[15] Heng Ji and Ralph Grishman. 2008. Refining Event Extraction through Cross-Document Inference. In *Proceedings of ACL*. 254–262. https://aclanthology.org/P08-1030/
[16] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of ICLR*. http://arxiv.org/abs/1412.6980
[17] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of ICLR*. https://openreview.net/forum?id=SJU4ayYgl
[18] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of ICML*. 282–289.
[19] Fayuan Li, Weihua Peng, Yuguang Chen, Quan Wang, Lu Pan, Yajuan Lyu, and Yong Zhu. 2020. Event Extraction as Multi-turn Question Answering. In *Proceedings of EMNLP*. https://doi.org/10.18653/v1/2020.findings-emnlp.73
[20] Jinqing Li, Xiaojun Chen, Dakui Wang, and Yuwei Li. 2021. Enhancing Label Representations with Relational Inductive Bias Constraint for Fine-Grained Entity Typing. In *Proceedings of IJCAI*. https://doi.org/10.24963/ijcai.2021/529
[21] Qi Li, Heng Ji, and Liang Huang. 2013. Joint Event Extraction via Structured Prediction with Global Features. In *Proceedings of ACL*. 73–82. https://aclanthology.org/P13-1008/
[22] Qi Li, Heng Ji, and Liang Huang. 2013. Joint Event Extraction via Structured Prediction with Global Features. In *Proceedings of ACL*. https://www.aclweb.org/anthology/P13-1008/
[23] Jinzhi Liao, Xiang Zhao, Xinyi Li, Lingling Zhang, and Jiuyang Tang. 2021. Learning Discriminative Neural Representations for Event Detection. In *Proceedings of SIGIR*. ACM, 644–653. https://doi.org/10.1145/3404835.3462977
[24] Hongyu Lin, Yaojie Lu, Xianpei Han, and Le Sun. 2019. Cost-sensitive Regularization for Label Confusion-aware Event Detection. In *Proceedings of ACL*. 5278–5283. https://doi.org/10.18653/v1/P19-1521
[25] Anan Liu, Ning Xu, and Haozhe Liu. 2021. Self-Attention Graph Residual Convolutional Networks for Event Detection with dependency relations. In *Findings of EMNLP*. 302–311. https://doi.org/10.18653/v1/2021.findings-emnlp.28
[26] Jian Liu, Yubo Chen, Kang Liu, Wei Bi, and Xiaojiang Liu. 2020. Event Extraction as Machine Reading Comprehension. In *Proceedings of EMNLP*. https://doi.org/10.18653/v1/2020.emnlp-main.128
[27] Shulin Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2017. Exploiting Argument Information to Improve Event Detection via Supervised Attention Mechanisms.

In *Proceedings of ACL*. 1789–1798. https://doi.org/10.18653/v1/P17-1164

[28] Xiao Liu, Zhunchen Luo, and Heyan Huang. 2018. Jointly Multiple Events Extraction via Attention-based Graph Information Aggregation. In *Proceedings of EMNLP*. https://doi.org/10.18653/v1/d18-1156

[29] Dongfang Lou, Zhilin Liao, Shumin Deng, Ningyu Zhang, and Huajun Chen. 2021. MLBiNet: A Cross-Sentence Collective Event Detection Network. In *Proceedings of ACL*. https://doi.org/10.18653/v1/2021.acl-long.373

[30] Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint Event Extraction via Recurrent Neural Networks. In *Proceedings of NAACL*. https://doi.org/10.18653/v1/n16-1034

[31] Thien Huu Nguyen and Ralph Grishman. 2015. Event Detection and Domain Adaptation with Convolutional Neural Networks. In *Proceedings of ACL*. 365–371. https://doi.org/10.3115/v1/p15-2060

[32] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of EMNLP*. 1532–1543. https://doi.org/10.3115/v1/d14-1162

[33] Antoine Raux and Maxine Eskenazi. 2007. A multi-layer architecture for semi-synchronous event-driven dialogue management. In *2007 IEEE Workshop on Automatic Speech Recognition & Understanding (ASRU)*. IEEE, 514–519.

[34] Barry Schiffman, Kathleen R. McKeown, Ralph Grishman, and James Allan. 2007. Question Answering Using Integrated Information Retrieval and Information Extraction. In *Proceedings of NAACL*. 532–539. https://aclanthology.org/N07-1067/

[35] Jiawei Sheng, Shu Guo, Bowen Yu, Qian Li, Yiming Hei, Lihong Wang, Tingwen Liu, and Hongbo Xu. 2021. CasEE: A Joint Learning Framework with Cascade Decoding for Overlapping Event Extraction. In *Findings of ACL*. https://doi.org/10.18653/v1/2021.findings-acl.14

[36] Jinghui Si, Xutan Peng, Chen Li, Haotian Xu, and Jianxin Li. 2021. Generating disentangled arguments with prompts: A simple event extraction framework that works. In *Proceedings of ICASSP*.

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Proceedings of NeurIPS*.

[38] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *Proceedings of ICLR*. https://openreview.net/forum?id=rJXMpikCZ

[39] Xiaozhi Wang, Xu Han, Zhiyuan Liu, Maosong Sun, and Peng Li. 2019. Adversarial Training for Weakly Supervised Event Detection. In *Proceedings of NAACL*. https://doi.org/10.18653/v1/n19-1105

[40] Xiaozhi Wang, Ziqi Wang, Xu Han, Wangyi Jiang, Rong Han, Zhiyuan Liu, Juanzi Li, Peng Li, Yankai Lin, and Jie Zhou. 2020. MAVEN: A Massive General Domain Event Detection Dataset. In *Proceedings of EMNLP*. https://doi.org/10.18653/v1/2020.emnlp-main.129

[41] Xiaozhi Wang, Ziqi Wang, Xu Han, Zhiyuan Liu, Juanzi Li, Peng Li, Maosong Sun, Jie Zhou, and Xiang Ren. 2019. HMEAE: Hierarchical Modular Event Argument Extraction. In *Proceedings of EMNLP*. https://doi.org/10.18653/v1/D19-1584

[42] Wenhan Xiong, Jiawei Wu, Deren Lei, Mo Yu, Shiyu Chang, Xiaoxiao Guo, and William Yang Wang. 2019. Imposing Label-Relational Inductive Bias for Extremely Fine-Grained Entity Typing. In *Proceedings of NAACL*. https://doi.org/10.18653/v1/n19-1084

[43] Haoran Yan, Xiaolong Jin, Xiangbin Meng, Jiafeng Guo, and Xueqi Cheng. 2019. Event Detection with Multi-Order Graph Convolution and Aggregated Attention. In *Proceedings of EMNLP*. https://doi.org/10.18653/v1/D19-1582

[44] Pan Yang, Xin Cong, Zhenyun Sun, and Xingwu Liu. 2021. Enhanced Language Representation with Label Knowledge for Span Extraction. In *Proceedings of EMNLP*. https://aclanthology.org/2021.emnlp-main.379

[45] Sen Yang, Dawei Feng, Linbo Qiao, Zhigang Kan, and Dongsheng Li. 2019. Exploring Pre-trained Language Models for Event Extraction and Generation. In *Proceedings of ACL*. https://doi.org/10.18653/v1/p19-1522

[46] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *Proceedings of NeurIPS*. 5754–5764. https://proceedings.neurips.cc/paper/2019/hash/dc6a7e655d7e5840e66733e9ee67cc69-Abstract.html