Document-Level Multi-Event Extraction with Event Proxy Nodes and Hausdorff Distance Minimization

Xinyu Wang[1,2], **Lin Gui**[2], **Yulan He**[1,2,3]
[1]Department of Computer Science, University of Warwick
[2]Department of Informatics, King's College London
[3]The Alan Turing Institute
Xinyu.Wang.11@warwick.ac.uk
{lin.1.gui, yulan.he}@kcl.ac.uk

Abstract

Document-level multi-event extraction aims to extract the structural information from a given document automatically. Most recent approaches usually involve two steps: (1) modeling entity interactions; (2) decoding entity interactions into events. However, such approaches ignore a global view of inter-dependency of multiple events. Moreover, an event is decoded by iteratively merging its related entities as arguments, which might suffer from error propagation and is computationally inefficient. In this paper, we propose an alternative approach for document-level multi-event extraction with event proxy nodes and Hausdorff distance minimization. The event proxy nodes, representing pseudo-events, are able to build connections with other event proxy nodes, essentially capturing global information. The Hausdorff distance makes it possible to compare the similarity between the set of predicted events and the set of ground-truth events. By directly minimizing Hausdorff distance, the model is trained towards the global optimum directly, which improves performance and reduces training time. Experimental results show that our model outperforms previous state-of-the-art method in F1-score on two datasets with only a fraction of training time. [1]

... [5] Shenkai Petrochemical Co., Ltd. received the receipt from the company's shareholder, **Yexiang Investment Management Co., Ltd.** on the evening of November 15, 2016 Regarding the notice of the shares being frozen. ... [8] On November 14, 2016, Yexiang Investment received the Notice of Litigation Preservation from the **People's Court of Binjiang District**, and granted a total of **47,577,481 shares** held by Yexiang Investment will be frozen, and the freezing period is from **October 31, 2016** to **October 30, 2019** ... [10] Yexiang Investment is … holding **47,577,481 shares** of the company, accounting for **13.07%** of the company's total share capital. ... [12] On **February 2, 2016**, the **42,000,000 shares** held by it are pledged to **Haitong Securities Co., Ltd.**, and the repurchase transaction date was **February 1, 2017**. ...

Event #1: **Equity Pledge**
Pledger: **Yexiang Investment Management Co., Ltd.**
Pledgee: **Haitong Securities Co., Ltd.**
TotalHoldingShares: **47,577,481 shares**
TotalHoldingRatio: **13.07%**
PledgedShares: **42,000,000 shares**
StartDate: **February 2, 2016**
EndDate: **February 1, 2017**

Event #2: **Equity Freeze**
EquityHolder: **Yexiang Investment Management Co., Ltd.**
LegalInstitution: **People's Court of Binjiang District**
TotalHoldingRatio: **13.07%**
FrozeShares: **47,577,481 shares**
StartDate: **October 31, 2016**
EndDate: **October 30, 2019**

Figure 1: An example of a document that contains two events. [·] denotes the sentence numbering. Words highlighted in colors denote different entities.

1 Introduction

Event extraction aims to identify event triggers with certain types and extract their corresponding arguments from text. Much research has been done on sentence-level event extraction (Du and Cardie, 2020; Lin et al., 2020; Lu et al., 2021). In recent years, there have been growing interests in tackling the more challenging task of document-level multi-event extraction, where an event is represented by a cluster of arguments, which may scatter across multiple sentences in a document. Also, multiple events in the same document may share some common entities. For example, as shown in Figure 1, the two events, *Equity Pledge* and *Equity Freeze*, have their arguments scattered across the document. The same entity mentions, *Yexiang Investment Management Co., Ltd.* and *13.07%*, are involved in both events, with the former taking different argument roles ('*Pledger*' and '*Equity Holder*'), while the latter having the same argument role ('*Total Holding Ratio*'). In such a setup, an event is not associated with a specific event trigger word or phrase, as opposed to the common setup in sentence-level event extraction. These challenges make it difficult to distinguish various events and link entities to event-specific argument roles.

Document-level multi-event extraction can be typically formulated as a table-filling task that fills

[1]Code is available at https://github.com/xnyuwg/procnet

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 10118–10133

the correct entities into a pre-defined event schema as shown in Figure 1. Here, an event is essentially represented by a cluster of arguments. Existing approaches (Zheng et al., 2019; Yang et al., 2021; Huang and Jia, 2021; Xu et al., 2021; Liang et al., 2022) usually involve two steps: (1) first model the entity interactions based on contextual representations; (2) then design a decoding strategy to decode the entity interactions into events and arguments. For example, Zheng et al. (2019) and Xu et al. (2021) transformed this task into sequential path-expanding sub-tasks. Each sub-task expands a path sequentially by gradually merging entities in a pre-defined order of event argument roles.

The aforementioned approaches suffer from the following limitations: (1) They decode events from entity information and tend to produce local optimal results without considering the interdependency of multiple events globally in a document. (2) Event decoding by iteratively merging entities suffers from error propagation that an event type or an entity that has been incorrectly classified cannot be corrected later. (3) Every decoding decision requires iterating all entity mentions in a document, which is computationally inefficient.

To address the above limitations, we propose an alternative approach for document-level multi-event extraction with event proxy nodes and Hausdorff distance minimization, named as **Pro**xy Nodes **C**lustering **Net**work (`ProCNet`). The event proxy nodes aim to capture the global information among events in a document. The Hausdorff distance makes it possible to optimize the training loss defined as the difference between the generated events and the gold standard event annotations directly. This is more efficient compared to existing decoding approaches.

Our method involves two main steps: *Event Representation Learning* and *Hausdorff Distance Minimization*. For *Event Representation Learning*, we create a number of proxy nodes, each of which represents a pseudo-event, and build a graph to update proxy nodes. Entities mentioned in text are treated as nodes connecting to the proxy nodes. All the proxy nodes are interconnected to allow information exchange among the potential events. We employ a Hypernetwork Graph Neural Network (GNN) (Ha et al., 2017) for updating proxy node representations. After *Event Representation Learning*, each proxy node essentially resides in a new event-level metric space by aggregating information from the entity-level space.

For *Hausdorff Distance Minimization*, we regard the predicted events as a set and the ground-truth events as another set, and compute the Hausdorff distance between these two sets, which simultaneously consider all events and all their arguments. We then minimize the Hausdorff distance via gradient descent, where the model is trained to directly produce a globally optimal solution without the need of using decoding strategies as in existing approaches.

In this way, our model learns globally and does not suffer from the problem of existing approaches that decode events based on local entity information. Each entity is linked to every proxy node, and the association between an entity and a proxy node is updated at each training iteration. As such, our model avoids the error propagation problem caused by the iterative decoding strategy. In addition, our approach naturally addresses the problem that the same entity mention may be involved in multiple events since the entity will be mapped to a different event-level metric space depending on its associated proxy node. Moreover, as our approach replaces iterative computation in decoding with parallel computation, it is computationally more efficient compared to existing path-expansion approaches, as will be shown in our experiments section. In summary, our main contributions are:

- We propose a new framework for document-level multi-event extraction by learning event proxy nodes in a new event-level metric space to better model the interactions among events.
- We propose to utilize the Hausdorff distance in our learning objective function to optimize the difference between the generated events and the gold standard events directly. The proposed mechanism not only simultaneously considers all events but also speeds up the training process.
- Experimental results show that our model outperforms previous state-of-the-art method in F1 on two datasets with only a fraction of training time.

2 Related Work

Early research on event extraction (EE) largely focused on sentence-level event extraction (SEE), aiming to classify the event trigger and arguments in a sentence. Chen et al. (2015) decomposes SEE

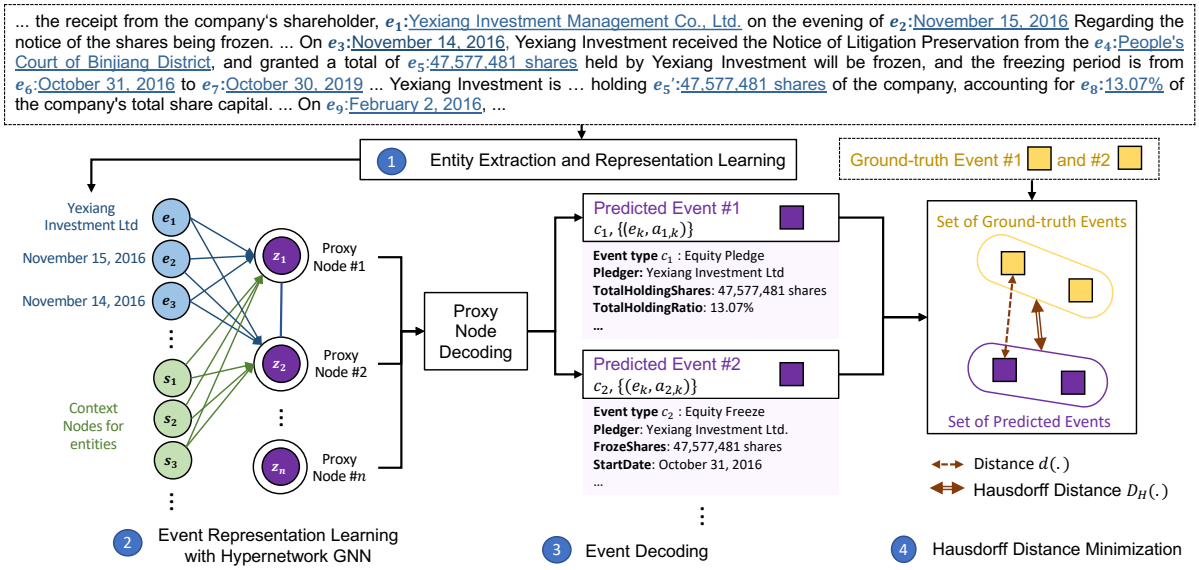

Figure 2: Overview of ProCNet with the example in Figure 1, where Entity 1, 5, 8, 9 are arguments of Event #1; Entity 1, 4, 5, 6, 7, 8 are arguments of Event #2; Entity 2, 3 do not belong to any events. Before training, proxy node embeddings are randomly initialized. Entities are first mapped to entity representations in the entity-level space by *Entity Representation Learning*. Then in *Event Representation Learning*, a hypernetwork heterogeneous graph is constructed with entity and context nodes connected with proxy nodes, and proxy nodes interconnected with each other. Proxy nodes are updated to represent pseudo-events. Afterwards, the proxy nodes and entity nodes are decoded into events, each of which is represented by an event types and a set of argument role-entity pairs in the *Event Decoding* step. Finally, *Hausdorff Distance Minimization* minimizes the distance between the set of predicted events and the set of ground-truth events to perform a global training in the new event-level metric space.

into two sub-tasks: *event trigger detection* and *event argument labeling*. More work has been done on joint-learning of the two sub-tasks (Nguyen and Nguyen, 2019; Lin et al., 2020). Recently, multi-turn Question-Answer (QA) methods have been investigated for EE with hand-designed or automatically generated questions (Du and Cardie, 2020; Li et al., 2020; Wang et al., 2020; Liu et al., 2020; Lyu et al., 2021). Apart from QA-based approaches, sequence-to-sequence learning has also been explored, where the event annotation is flattened as a sequence (Paolini et al., 2021; Lu et al., 2021; Li et al., 2021; Lu et al., 2022b). More recently, prompt-based learning has been explored using the knowledge in pre-trained language models (Lin et al., 2021; Hsu et al., 2021; Ma et al., 2022).

Compared to SEE, document-level event extraction (DEE) appears to be more challenging. DEE requires methods to model long-term dependencies among entities across multiple sentences. Simply employing SEE approaches for DEE may lead to incomplete and uninformative extractions (Li et al., 2021). To address the problem, conditional generation have been proposed, which are conditioned on pre-specified templates or prompts (Du et al., 2021; Huang et al., 2021; Ma et al., 2022).

DEE can also be formulated as a table-filling task where each event is represented as a cluster of arguments and an event type. In such a setup, it is usually not possible to associate a particular event trigger word or phrase with an event. Yang et al. (2018) proposed a key-event detection model. Zheng et al. (2019) transformed event tables into a directed acyclic graph with path expansion. Huang and Jia (2021) constructed a graph to build sentence communities. Lu et al. (2022a) captured event clues as a series of intermediate results. Xu et al. (2021) constructed a heterogeneous GNN with a tracker mechanism for partially decoded events. Liang et al. (2022) modeled the relation between entities with Relation-augmented Attention Transformer. These methods mainly focus on modeling entity inter-relations and rely on carefully-designed event decoding strategies. In contrast, we model events in the event-level metric space within a more global view and with less training time.

3 Methodology

3.1 Problem Setup

Different from the trigger-based event extraction task, where an event is represented by a trigger and

a list of arguments, in our task, an event is defined by an event type category c, a list of entities $\{e_i\}$ and their corresponding argument types $\{a_i\}$ as shown in Figure 1. Therefore, the target output is a list of "entity-argument" pairs $\{(e_i, a_i)\}$ and c as $\big(c, \{(e_i, a_i)\}\big)$. Proxy node is defined as z. An overview of `ProCNet` is shown in Figure 2. In what follows, we present each module in detail.

3.2 Entity Representation Learning

Given an input document, the first step is to identify the entities which might be potential arguments. This can be framed as a sequence labeling problem that, given a word sequence, the entity recognition model outputs a label sequence with the BIO (Beginning and Inside of an entity span, and Other tokens) tagging. We use BERT (Devlin et al., 2019) as a sequence labeler to detect entities at sentence-level. As an entity span may contain multiple tokens, we drive its representation by averaging the hidden states of its constituent tokens. For a document, a total of $|e|$ entity representations are extracted as $\{\boldsymbol{h}_{e_i}\}_{i=1}^{|e|}$. The loss of the BIO sequence tagging is defined as $\mathcal{L}_{\text{er}}$.

In order to make the entity representations encode the knowledge of entity associations, we introduce a simple auxiliary learning task to predict whether two entities belong to the same event, where entity representations will be updated during learning. Specifically, it is a binary classification task, with the predicted output computed as:

$$\hat{y}_{\text{epc}_{(i,j)}} = \phi\left(\text{MLP}([\boldsymbol{h}_{e_i}; \boldsymbol{h}_{e_j}])\right), \tag{1}$$

where ϕ denotes the sigmoid function, $[;]$ denotes the concatenation, and $\hat{y}_{\text{epc}_{i,j}}$ indicates the probability if entities i and j are from the same event. We use the binary cross-entropy (CE) loss here:

$$\mathcal{L}_{\text{epc}} = -\sum_i \sum_j \text{CE}(y_{\text{epc}_{(i,j)}}, \hat{y}_{\text{epc}_{(i,j)}}) \tag{2}$$

where $y_{\text{epc}_{i,j}}$ is the label. The loss for entity representation learning is defined as $\mathcal{L}_{\text{e}} = \mathcal{L}_{\text{er}} + \mathcal{L}_{\text{epc}}$.

3.3 Event Representation Learning with Proxy Nodes

In this section, we construct a graph to map entity representations in the entity-level space into event representations in a new event-level metric space.

We define n proxy nodes, which serve as pseudo-events, and randomly initialize their embeddings $\{\boldsymbol{h}_{z_i}^{(0)}\}_{i=1}^{n}$, which are only initialized once before training and will be updated during training. n is a hyper-parameter and can be simply set to a much larger value than the expected number of extracted events, as proxy nodes can also represent *null* events (see Section 3.4). We initialize entity node embeddings $\{\boldsymbol{h}_{e_i}\}_{i=1}^{|e|}$ and context node embeddings $\{\boldsymbol{h}_{s_i}\}_{i=1}^{|s|}$ by their corresponding entity and `[CLS]` representations, respectively.

We define the graph as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, and the node set $\mathcal{V}$ contains proxy nodes, entity nodes, and context nodes as: $\mathcal{V} = \{z_i\}_{i=1}^{n} \cup \{e_i\}_{i=1}^{|e|} \cup \{s_i\}_{i=1}^{|s|}$ with their embeddings $\{\boldsymbol{h}_{z_i}^{(0)}\}_{i=1}^{n} \cup \{\boldsymbol{h}_{e_i}\}_{i=1}^{|e|} \cup \{\boldsymbol{h}_{s_i}\}_{i=1}^{|s|}$. The edge set $\mathcal{E}$ includes three kinds of edges as follows:

Proxy↔Proxy Edge The bidirectional edge between all proxy nodes $\{z_i \rightarrow z_j : 0 < i \leq n, 0 < j \leq n\}$ allows the information exchange between proxy nodes.

Entity→Proxy Edge The directed edge from all entity nodes e to all proxy nodes z as $\{e_j \rightarrow z_i : 0 < i \leq n, 0 < j \leq |e|\}$ provides the entity information for pseudo-events.

Context→Proxy Edge The directed edge from all context node s to all proxy node z as $\{s_j \rightarrow z_i : 0 < i \leq n, 0 < j \leq |s|\}$ provides the contextual information.

In a typical setup for GNN, each node has its embedding updated by aggregating the neighborhood information. The aggregation weight matrix is shared across all nodes. But in our task here, each proxy node is expected to represent a distinct event. As such, we would like to have a unique aggregation function for each proxy node. To this end, we use the Graph Neural Network with Feature-wise Linear Modulation (GNN-FiLM) (Brockschmidt, 2020) to update the proxy node embeddings in $\mathcal{G}$. It introduces Hypernetwork to enable each proxy node to compute a unique aggregation function with different parameters. More concretely, given a node $v \in \mathcal{V}$ at the $(l+1)$-th layer, its hidden representation $\boldsymbol{h}_v^{(l+1)}$ is updated as:

$$\begin{aligned}
\boldsymbol{h}_v^{(l+1)} &= \sigma\Bigg(\sum_{u \xrightarrow{\varepsilon} v} \boldsymbol{\gamma}_{\varepsilon,v}^{(l)} \odot \boldsymbol{W}_{\varepsilon} \boldsymbol{h}_u^{(l)} + \boldsymbol{\beta}_{\varepsilon,v}^{(l)}\Bigg), \\
\boldsymbol{\gamma}_{\varepsilon,v}^{(l)} &= f_{\gamma}(\boldsymbol{h}_v^{(l)}; \boldsymbol{\theta}_{\gamma,\varepsilon}), \quad \boldsymbol{\beta}_{\varepsilon,v}^{(l)} = f_{\beta}(\boldsymbol{h}_v^{(l)}; \boldsymbol{\theta}_{\beta,\varepsilon}),
\end{aligned} \tag{3}$$

where $u \xrightarrow{\varepsilon} v$ denotes a neighboring node u connected with node v with the edge type ε. $\boldsymbol{W}_{\varepsilon} \in \mathbb{R}^{d_h \times d_h}$ is a learnable parameter for edge type ε. σ

and $\odot$ denote the activation function and Hadamard product, respectively. $\boldsymbol{\gamma}_{\varepsilon,v}^{(l)}$ and $\boldsymbol{\beta}_{\varepsilon,v}^{(l)}$ define the message-passing function of edge type ε and node v at layer l. They are computed by functions f_γ and f_β given $\boldsymbol{h}_v^{(l)}$ as the input. $\boldsymbol{\theta}_{\gamma,\varepsilon}$ and $\boldsymbol{\theta}_{\beta,\varepsilon}$ are learnable parameters of f_γ and f_β, respectively. To keep it simple, we only use one-layer GNN-FiLM with a single linear layer as the hyper-function in our experiments.

With the above formulation, each proxy node z has its unique message-passing function to aggregate information from entity nodes and context nodes in different ways. In summary, the representations of proxy nodes $\{\widehat{\boldsymbol{h}}_{z_i}\}_{i=1}^n$ are updated through GNN-FiLM learning:

$$\{\widehat{\boldsymbol{h}}_{z_i}\}_{i=1}^n = \text{GNN-FiLM}(\mathcal{V}, \mathcal{E}) \tag{4}$$

where z_i represents a pseudo-event. The training with proxy nodes is challenging, which will be addressed in Section 3.5.

3.4 Event Decoding

In this section, each proxy node representation $\widehat{\boldsymbol{h}}_{z_i}$ is decoded into an event, which is formulated into two parallel sub-tasks: *event type classification* and *event argument classification*.

Event Type Classification The event type of proxy node z_i is inferred from $\widehat{\boldsymbol{h}}_{z_i}$ with MLP as:

$$\boldsymbol{p}_{\hat{c}_i} = \text{softmax}\left(\text{MLP}(\widehat{\boldsymbol{h}}_{z_i})\right), \tag{5}$$

where $\boldsymbol{p}_{\hat{c}_i}$ denotes the event type probability distribution of z_i. Event type labels includes a *null* event type, denoting no correspondence between a proxy node and any events. The number of non-*null* proxy nodes is the number of predicted events.

Event Argument Classification In this task, we need to associate an entity with an event under an event-specific argument type. As the same entity (e.g., a company name) may have multiple mentions in a document, we aggregate their representations by a Multi-Head Attention (MHA) mechanism using a proxy node as the query. More concretely, assuming $\{\boldsymbol{h}_e\}_{e\in\bar{e}_k}$ denotes a set of mentions representations for the same entity $\bar{e}_k$, we use MHA to derive the aggregated entity representation for $\bar{e}_k$. The query, key and value are defined as $\boldsymbol{Q}_{z_i} = \widehat{\boldsymbol{h}}_{z_i}, \boldsymbol{K}_{\bar{e}_k} = \{\boldsymbol{h}_e\}_{e\in\bar{e}_k}, \boldsymbol{V}_{\bar{e}_k} = \{\boldsymbol{h}_e\}_{e\in\bar{e}_k}$. The representation of $\bar{e}_k$ is:

$$\widehat{\boldsymbol{h}}_{z_i,\bar{e}_k} = \text{MHA}(\boldsymbol{Q}_{z_i}, \boldsymbol{K}_{\bar{e}_k}, \boldsymbol{V}_{\bar{e}_k}), \tag{6}$$

where $\widehat{\boldsymbol{h}}_{z_i,\bar{e}_k}$ denotes the aggregated representation for entity $\bar{e}_k$ using the proxy node z_i as the query. Then the probability distribution $\boldsymbol{p}_{\hat{a}_{i,k}}$ of argument types of entity $\bar{e}_k$ with respect to proxy node z_i is:

$$\boldsymbol{p}_{\hat{a}_{i,k}} = \text{softmax}\left(\text{MLP}([\widehat{\boldsymbol{h}}_{z_i}; \widehat{\boldsymbol{h}}_{z_i,\bar{e}_k}])\right), \tag{7}$$

where $[;]$ denotes the concatenation. The argument type set includes a *null* argument type, denoting that entity $\bar{e}_k$ does not relate to proxy node z_i.

The final event type $\hat{c}_i$ for proxy node z_i and argument type for entity $\bar{e}_k$ under the event encoded by proxy node z_i are determined by:

$$\begin{aligned} \hat{c}_i &= \text{argmax}(\boldsymbol{p}_{\hat{c}_i}) \\ \hat{a}_{i,k} &= \text{argmax}(\boldsymbol{p}_{\hat{a}_{i,k}}) \end{aligned} \tag{8}$$

Each event is represented by an event type $\hat{c}_i$ and a list of arguments $\{\hat{a}_{i,k}\}$. Any predicted argument type which is not in the pre-defined schema for its associated event type will be removed. Proxy nodes classified as *null* event or entities classified as *null* arguments will be removed. If there are multiple entities predicted as the same argument, the one with the highest probability will be kept.

3.5 Hausdorff Distance Minimization

In this section, we construct a predicted pseudo-event set $\mathcal{U}_z$ represented by proxy node and a ground-truth event set $\mathcal{U}_y$. We define μ_{z_i} as the i-th pseudo-event, represented by z_i, with $\big(\hat{c}_i, \{(e_k, \hat{a}_{i,k})\}\big)$, and μ_{y_i} denotes the j-th ground-truth event $\big(c_j, \{(e_k, a_{j,k})\}\big)$. We further define the distance $d(\mu_{z_i}, \mu_{y_j})$ between predicted event μ_{z_i} and the ground-truth event μ_{y_i} as:

$$\begin{aligned} d(\mu_{z_i}, \mu_{y_j}) &= \text{CE}(\boldsymbol{p}_{\hat{c}_i}, c_j) \\ &\quad + \frac{1}{|\bar{e}|}\sum_{k=1}^{|e|} \text{CE}(\boldsymbol{p}_{\hat{a}_{i,k}}, a_{j,k}) \end{aligned} \tag{9}$$

where CE(.) is the cross-entropy loss; $|\bar{e}|$ denotes the number of unique entities; k indicates different entities. $d(\mu_z, \mu_y)$ is essentially computed by the total cross-entropy loss of event type classification and argument classification between the i-th proxy node and the j-th ground-truth event.

We aim to minimize the Hausdorff distance between sets $\mathcal{U}_z$ and $\mathcal{U}_y$ to learn the model by considering all events and their arguments simultaneously. As the standard Hausdorff distance is highly sensitive to outliers, we use the average Hausdorff

distance (Schütze et al., 2012; Taha and Hanbury, 2015):

$$D_H(\mathcal{U}_z,\mathcal{U}_y) = \frac{1}{|\mathcal{U}_z|}\sum_{\mu_z\in\mathcal{U}_z}\min_{\mu_y\in\mathcal{U}_y} d(\mu_z,\mu_y) + \frac{1}{|\mathcal{U}_y|}\sum_{\mu_y\in\mathcal{U}_y}\min_{\mu_z\in\mathcal{U}_z} d(\mu_z,\mu_y) \tag{10}$$

However, in our task, the average Hausdorff distance could suffer a problem that a predicted event, represented by a proxy node, may be guided to learn towards more than one different t event at the same training iteration when this proxy node is the closest neighbor of multiple ground-truth events.

To address this problem, we add a constraint to the average Hausdorff distance that the distance computation of $d(.)$ should only be performed no more than once on each μ_z and μ_y, and we modify the average Hausdorff distance as:

$$\widehat{D}_H(\mathcal{U}_z,\mathcal{U}_y) = \min\left\{\sum_{(\mu_z,\mu_p)\in\mathcal{U}_z\times\mathcal{U}_y} d(\mu_z,\mu_y)\right\} \tag{11}$$

For example, if $d(\mu_{z_1},\mu_{y_1})$ has been computed, then $d(\mu_{z_2},\mu_{y_1})$ is no longer allowed to perform, as μ_{y_1} has been used in $d(.)$ computation.

To this end, Eq. (11) with the constraint becomes a minimum loss alignment problem. To better solve Eq. (11) under the constraint, we construct an undirected bipartite graph $G = (\mathcal{U}_z,\mathcal{U}_y,\mathcal{T})$, where $\mu_z \in \mathcal{U}_z$ and $\mu_y \in \mathcal{U}_y$ are nodes of two parts representing the predicted events and the ground-truth events, respectively. $t \in \mathcal{T}$ denotes edge, which only exists between μ_z and μ_y. The weight of edge t between nodes μ_z and μ_y is defined as:

$$w(t_{z,y}) = d(\mu_z,\mu_y) \tag{12}$$

The first step is to find an edge set $\mathcal{T}$ that achieves the minimum value in the following equation:

$$\widehat{\mathcal{T}} = \text{argmin}\sum_{t_{z,y}\in\mathcal{T}} w(t_{z,y}), \tag{13}$$

where the edge $t \in \mathcal{T}$ must meet these conditions: (1) each μ_z has exactly one edge connected to it; (2) each μ_y has no more than one edge connected to it. Eq. (13) can be computed efficiently with (Ramakrishnan et al., 1991; Bertsekas, 1981). Then the final distance is computed by combining Eq. (11), (12), and (13) as:

$$\widehat{D}_H(\mathcal{U}_z,\mathcal{U}_y) = \sum_{t_{z,y}\in\widehat{\mathcal{T}}} w(t_{z,y}) \tag{14}$$

Finally, we use $\widehat{D}_H(\mathcal{U}_z,\mathcal{U}_y)$ to approximate average Hausdorff distance $D_H(\mathcal{U}_z,\mathcal{U}_y)$.

As n has been set to be a very large number, if the number of ground-truth events is less than the number of predicted events in a document, pseudo *null* events are added to the ground-truth event set as negative labels to make the number of ground-truth events equals to the number of predicted events.

In summary, $\widehat{D}_H(\mathcal{U}_z,\mathcal{U}_y)$ is the distance between the predicted events set and the ground-truth events set, which considers all events with all of their arguments at the same time, essentially capturing a global alignment.

3.6 Objective Function

The final loss is the sum of approximate Hausdorff distance and entity representation loss:

$$\mathcal{L} = \widehat{D}_H(\mathcal{U}_z,\mathcal{U}_y) + \mathcal{L}_{\text{e}} \tag{15}$$

4 Experiments

In this section, we present performance and runtime experiments in comparison with state-of-the-art approaches. We also discuss the ablations study. Entity and event visualisation results can be found in Appendix B.

4.1 Experimental Setup

Dataset We evaluate `ProCNet` on the two document-level multi-event extraction datasets: (1) ChFinAnn dataset[2] (Zheng et al., 2019) consists of 32,040 financial documents, with 25,632, 3,204, and 3,204 in the train, development, and test sets, respectively, and includes five event types. The dataset contains 71% of single-event documents and 29% of multi-event documents. (2) DuEE-Fin dataset[3] (Han et al., 2022) has around 11,900 financial documents and 13 event types. As the dataset has not released the ground truth annotations for the test set, we follow the setting of (Liang et al., 2022) and treat the original development set as the test set. We also set aside 500 documents from the training set as the development set. Our final dataset has 6,515, 500, and 1,171 documents in the train, development, and test set, respectively. There are 67% of single-event documents and 33% of multi-event documents. More details about the event types and their distributions are in Appendix A.1.

[2] https://github.com/dolphin-zs/Doc2EDAG

[3] https://aistudio.baidu.com/aistudio/competition/detail/46/0/task-definition

Model	ChFinAnn					DuEE-Fin				
	P.	R.	F1	F1 (S.)	F1 (M.)	P.	R.	F1	F1 (S.)	F1 (M.)
DCFEE-O	68.0	63.3	65.6	69.9	50.3	59.8	55.5	57.6	62.7	53.3
DCFEE-M	63.0	64.6	63.8	65.5	50.5	50.2	55.5	52.7	57.1	49.5
Greedy-Dec	82.5	53.7	65.1	80.2	36.9	66.0	50.6	57.3	67.8	47.4
Doc2EDAG	82.7	75.2	78.8	83.9	67.3	67.1	60.1	63.4	69.1	58.7
DE-PPN	83.7	76.4	79.9	85.9	68.4	69.0	33.5	45.1	54.2	21.8
PTPCG	83.7	75.4	79.4	88.2	-	71.0	61.7	66.0	-	-
GIT	82.3	78.4	80.3	87.6	72.3	69.8	65.9	67.8	73.7	63.8
ReDEE	83.9	79.9	81.9	88.7	74.1	77.0	72.0	74.4	78.9	70.6
ProCNet (Ours)	**84.1**	**81.9**	**83.0**	**89.6**	**75.6**	**78.8**	**72.8**	**75.6**	**80.0**	**72.1**

Table 1: Overall precision (P.), recall (R.), and F1-score (F1) on the ChFinAnn and DuEE-Fin datasets. F1 (S.) and F1 (M.) denote scores under Single-event (S.) and Multi-event (M.) sets.

Evaluation Metrics We follow the same metrics in (Zheng et al., 2019). For a predicted event of a specific event type, the most similar ground-truth event that is of the same event type is selected without replacement. Then the micro-averaged role-level precision, recall, and F1-score are calculated for the predicted event and the selected gold event.

Implementation Detail To keep it simple, we only use one-layer GNN-FiLM (Brockschmidt, 2020) with a single linear layer as the hyper-function. Specifically, we have $f_\gamma(\boldsymbol{h}_v^{(l)}; \boldsymbol{\theta}_{\gamma,\varepsilon}) = \boldsymbol{W}_{\gamma,\varepsilon}\boldsymbol{h}_v^{(l)}$ and $f_\beta(\boldsymbol{h}_v^{(l)}; \boldsymbol{\theta}_{\beta,\varepsilon}) = \boldsymbol{W}_{\beta,\varepsilon}\boldsymbol{h}_v^{(l)}$ in Eq. (3). The number of proxy nodes n is set to 16. More implementation details are in Appendix A.2

Baselines The baselines that we compare with are as follows: **DCFEE** (Yang et al., 2018) uses an argument-completion strategy in the table-filling task. Two variants of DCFEE are **DCFEE-O** for single-event and **DCFEE-M** for multi-event. **Doc2EDAG** (Zheng et al., 2019) utilizes a path-expansion decoding strategy to extract events like hierarchical clustering. **Greedy-Dec** is a variant of Doc2EDAG that decodes events greedily. **DE-PPN** (Yang et al., 2021) uses Transformer to encode sentences and entities. **GIT** (Xu et al., 2021) uses a Tracker module to track events in the path-expansion decoding. **PTPCG** (Zhu et al., 2022) combines event arguments together in a non-autoregressive decoding approach with pruned complete graphs, aiming to consume lower computational resources. **ReDEE** (Liang et al., 2022) is a Relation-augmented Attention Transformer to cover multi-scale and multi-amount relations.

Model	EF	ER	EU	EO	EP
DCFEE-O	51.1	83.1	45.3	46.6	63.9
DCFEE-M	45.6	80.8	44.2	44.9	62.9
Greedy-Dec	58.9	78.9	51.2	51.3	62.1
Doc2EDAG	70.2	87.3	71.8	75.0	77.3
DE-PPN	73.5	87.4	74.4	75.8	78.4
GIT	73.4	90.8	74.3	76.3	77.7
ReDEE	74.1	90.7	75.3	**78.1**	80.1
ProCNet (Ours)	**75.7**	**93.7**	**76.0**	72.0	**81.3**

Table 2: F1-score of five event types on ChFinAnn.

4.2 Overall Results

Table 1 shows the results on the ChFinAnn and the DuEE-Fin datasets. For ChFinAnn, the baseline results are reported in (Zheng et al., 2019; Yang et al., 2021; Xu et al., 2021; Zhu et al., 2022; Liang et al., 2022). For DuEE-Fin, the baseline results are either taken from (Liang et al., 2022) or by running the published source code of the baselines. We can observe that a simple argument completion strategy (DCFEE-O and DCFEE-M) produces the worst results. Greedy-Dec with the greedy decoding strategy improves upon DCEFF variants, but it reached an F1-score lower than Doc2EDAG by 13.7% on ChFinAnn and 6.3% on DuEE-Fin due to only modeling entity-level representations without a global view. DE-PPN which uses the Transformer to encode sentences and entities performs worse compared to Doc2EDAG which utilizes a path-expansion decoding strategy. Extending DocEDAG with a Track module (GIT) or using a relation-augmented attention transformer (ReDEE) achieves better results compared to earlier approaches. ProCNet gives the best overall

Model	WB	FL	BA	BB	CF	CL	SD	SI	SR	RT	PR	PL	EC
DCFEE-O	54.0	65.4	44.0	27.3	58.2	42.0	48.8	53.9	76.7	32.9	63.3	58.3	40.6
DCFEE-M	49.2	68.0	40.4	28.4	51.2	35.1	42.3	45.9	74.0	51.0	55.8	56.4	37.4
Greedy-Dec	53.7	71.8	49.5	41.1	61.3	42.1	49.7	57.4	74.4	29.2	60.8	50.5	39.4
Doc2EDAG	60.0	78.3	50.6	40.1	63.2	51.5	50.7	52.9	83.7	51.2	64.8	61.7	51.2
DE-PPN	50.7	62.7	41.3	21.4	36.3	23.0	32.9	31.3	67.8	25.8	42.1	36.3	23.4
GIT	58.8	77.6	56.6	44.7	68.5	55.1	58.8	**71.2**	86.4	45.0	66.4	71.3	53.8
ReDEE	72.2	81.2	58.9	53.4	76.7	56.7	68.2	56.6	**90.6**	49.9	75.0	**77.8**	56.6
ProCNet (Ours)	**76.0**	**85.0**	**69.8**	**63.5**	**79.0**	**60.5**	**69.3**	68.2	89.2	**50.0**	**77.4**	76.9	**56.9**

Table 3: F1-score on the DuEE-Fin dataset with 13 event types.

F1-score, outperforming the best baseline, ReDEE, by 1.1-1.2%, respectively on ChFinAnn and DuEE-Fin. It can also be observed that all models have better F1-scores for the single-event scenario than the multi-event one, verifying the difficulty of extracting multiple events from a document. When comparing results across the two datasets, we see better results achieved on ChFinAnn, possibly due to its larger training set and smaller set of event types compared to DuEE-Fin.

4.3 Per-Event-Type Results

Table 2 and Table 3 show the evaluation results on the 5 and 13 event types[4] on ChFinAnn and DuEE-Fin, respectively. On ChFinANN, ReDEE outperforms the others on EO. On DuEE-Fin, ReDEE gives the best results on SR and PL, while GIT outperforms the others on SI. Some documents of these event types contain more than 40 sentences. A possible reason for ProCNet not performing well on these event types is its limited capability of capturing long-term dependencies across sentences, since ProCNet does not directly model the relations between sentences. On the contrary, ReDEE and GIT model the inter-relations of sentences directly. Nevertheless, ProCNet achieves superior results on other event types, resulting in overall better performance compared to baselines.

4.4 Run-Time Comparison

We compare the training time of the five baselines, Doc2EDAG, DE-PPN, PTPCG, GIT, and ReDEE, with ProCNet on a GPU server with NVIDIA Quadro RTX 6000 and the same setting. We record the average per epoch training time and the total time to reach convergence in Table 4. DuEE-Fin contains fewer data than ChFinANN, as such, Doc2EDAGE, GIT, and ProCNet trained faster on DuEE-Fin. However, ReDEE took longer time to converge on DuEE-Fin, because ReDEE models the relations of all argument-argument pairs. As the number of event types and argument types in DuEE-Fin is more than that in ChFinANN, the training time of ReDEE increases exponentially. DE-PPN runs faster than Doc2EDAG, GIT, and ReDEE but slower than ProCNet. In contrast, ProCNet avoids the time-consuming decoding by introducing the proxy nodes and HDM. Besides, ProCNet can run all proxy nodes and their arguments in parallel, which is more GPU-friendly. PTPCG has a shorter per-epoch run time, but took a longer time to converge on ChFinAnn; though it appears to be more run time efficient on DuEE-Fin compared to our approach. In summary, ProCNet is 0.5x-44.8x times faster than baselines per epoch, and is 0.6x-45.4x times faster to reach convergence.

Model	Per Epoch		Convergence	
	Time	Ratio	Time	Ratio
ChFinANN				
Doc2EDAG	4:40	5.2x	327:09	10.7x
DE-PPN	1:54	2.1x	87:27	2.8x
PTPCG	0:26	0.5x	39:04	1.3x
GIT	4:48	5.3x	317:35	10.3x
ReDEE	8:12	9.1x	525:33	17.1x
ProCNet (Ours)	0:54	1.0x	30:34	1.0x
DuEE-Fin				
Doc2EDAG	1:53	11.3x	249:35	16.5x
DE-PPN	0:15	1.5x	24:36	1.6x
PTPCG	0:06	0.6x	9:28	0.6x
GIT	1:50	11.0x	178:38	11.8x
ReDEE	7:28	44.8x	687:14	45.4x
ProCNet (Ours)	0:10	1.0x	15:09	1.0x

Table 4: The GPU time (hh:mm) of each epoch and reaching convergence in average.

[4]Please refer to Appendix A.1 for event type descriptions.

... [9] Name of shareholder: Guangxin Holding Group Co., Ltd. ... [11] Guangxin Group disclosed its share increase plan on **October 23, 2018**. ... [21] As of **November 19, 2018**, Guangxin Group has accumulated 1653640 shares of the company through centralized bidding through the Shanghai Stock Exchange trading system, accounting for 0.08% of the company's total share capital, with an average increase price of 9.59 yuan per share, the increase amount is 15854061.0 yuan,... [22] On **November 20, 2018**, Guangxin Group increased its holdings of the company's shares by 1788000 shares through centralized bidding through the Shanghai Stock Exchange trading system, accounting for 10% of the company's total share capital. 0.08%, the average increase price is 9.36 yuan per share, and the increase amount is 16740153.0 yuan. ... [24] After the above-mentioned increase in holdings, Guangxin Group directly held **264558774 shares** of the company, accounting for 12.49% of the company's total share capital ...

True Events:	**Predicted Events:**
Event #1: Equity Overweight	**Event #1: Equity Overweight**
EquityHolder: Guangxin Holding Group Co., Ltd.	EquityHolder: Guangxin Holding Group Co., Ltd.
TradedShares: 1653640 shares	TradedShares: 1653640 shares
StartDate: October 23, 2018	StartDate: **null**
EndDate: November 19, 2018	EndDate: **November 20, 2018**
LaterHoldingShares: null	LaterHoldingShares: null
AveragePrice: 9.59 yuan per share	AveragePrice: 9.59 yuan per share
Event #2: Equity Overweight	**Event #2: Equity Overweight**
EquityHolder: Guangxin Holding Group Co., Ltd.	EquityHolder: Guangxin Holding Group Co., Ltd.
TradedShares: 1788000 shares	TradedShares: 1788000 shares
StartDate: November 20, 2018	StartDate: **null**
EndDate: November 20, 2018	EndDate: November 20, 2018
LaterHoldingShares: 264558774 shares	LaterHoldingShares: **null**
AveragePrice: 9.36 yuan per share	AveragePrice: 9.36 yuan per share

Figure 3: Error case study with incorrect arguments colored in red. [.] denotes the sentence numbering.

4.5 Ablation Study

Model	ChFinANN			DuEE-Fin		
	P.	R.	F1	P.	R.	F1
ProCNet (Ours)	84.1	81.9	83.0	78.8	72.8	75.6
−Hypernetwork	82.7	81.6	82.1	77.0	72.2	74.5
−Proxy node	41.3	2.3	4.4	21.1	1.0	1.7
−HDM	17.0	19.8	18.3	13.3	8.2	10.1

Table 5: Ablation study on ChFinAnn and DuEE-Fin.

Table 5 shows how different components in ProCNet contribute to performance:

−Hypernetwork Hypernetwork is removed by replacing GNN-FiLM with RGCN (Schlichtkrull et al., 2018), where all proxy nodes in RGCN share the same message-passing function. We see a drop of about 1% in F1 on both datasets, showing the importance of using different entity aggregation functions for different event proxy nodes.

−Proxy Node We replace $\{\boldsymbol{h}_{z_i}\}_{i=1}^{n}$ with $\{\boldsymbol{h}_{z_0}\}_{i=1}^{n}$, where all proxy nodes share the same embedding $\boldsymbol{h}_{z_0}$. In this way, $\boldsymbol{h}_{z_0}$ acts as a common start node as in existing baselines. It can be observed that F1 drops significantly to 4.4% and 1.7%, respectively. The model learns almost nothing, which verifies the importance of the proxy nodes for ProCNet.

−HDM Instead of minimizing the Hausdorff distance between the predicted set and the ground-truth set globally, we randomly initialize the edge set $\widehat{\mathcal{T}}$ without employing Eq. (13), where the minimization is not performed towards the global minimum. We see a drastic decrease in performance. Without HDM, it is difficult for the the model to learn the alignment between a proxy node and a ground-truth event, showing that HDM is an indispensable component of ProCNet.

4.6 Case Study

Figure 3 shows an error case of ProCNet. *Event #1* spans from sentence #9 to sentence #21, and the *StartDate* is too far from the main context of *Event #1*. Moreover, the classification of *LaterHoldingShares* in *Event #2* requires the model to relate the pronoun *above-mentioned* to the *Event #2*. These mistakes show that ProCNetstill faces a difficulty in modeling long-distance dependencies.

5 Conclusion

In this paper, we no longer focus on inter-entities relation modeling and decoding strategy as in previous methods, but directly learns all events globally through the use of event proxy nodes and the minimization of the Hausdorff distance in our proposed ProCNet. In our experiments, ProCNet outperforms state-of-the-art approaches while only requiring a fraction of time for training.

Acknowledgements

This work was supported in part by the UK Engineering and Physical Sciences Research Council (grant no. EP/T017112/2, EP/V048597/1, EP/X019063/1). YH is supported by a Turing AI Fellowship funded by the UK Research and Innovation (grant no. EP/V020579/2).

Limitations

In our proposed model, we introduce a hyper-parameter n as the number of event proxy nodes. The value of n needs to be pre-set. Setting n to a value larger than the actual event number in a document would lead to computational redundancy as more proxy nodes would be mapped to the *null* event. However, setting n to a small value may miss some events in a document. We have experimented with automatically learning the value of n based on an input document in `ProCNet`. But we did not observe improved event extraction performance. As such, we simply set it to 16. In the ChFinAnn dataset, 98% documents have less than 7 events annotated. This results in the learning of many redundant proxy nodes for such documents. It remains an open challenge on automatically learning a varying number of event proxy nodes based on an input document. Reducing the number of redundant proxy nodes can reduce training time further.

Another shortcoming is the limited capability of `ProCNet` in capturing the long-term dependencies of sentences, as have been discussed in the per-event-type results in Section 4.2 and 4.3. We observed a relatively worse performance of `ProCNet` in dealing with long documents with more than 40 sentences as it does not explicitly model the inter-relations of sentences. One possible direction is to explore the use of a heterogeneous graph which additionally models the entity-entity, entity-sentence, and sentence-sentence relations. We will leave it as the future work to study the trade-off between event extraction performance and training efficiency.

References

Dimitri P. Bertsekas. 1981. A new algorithm for the assignment problem. *Mathematical Programming*, 21:152–171.

Marc Brockschmidt. 2020. Gnn-film: Graph neural networks with feature-wise linear modulation. In *ICML*.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 167–176, Beijing, China. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Xinya Du and Claire Cardie. 2020. Event extraction by answering (almost) natural questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 671–683, Online. Association for Computational Linguistics.

Xinya Du, Alexander Rush, and Claire Cardie. 2021. GRIT: Generative role-filler transformers for document-level event entity extraction. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 634–644, Online. Association for Computational Linguistics.

David Ha, Andrew M. Dai, and Quoc V. Le. 2017. Hypernetworks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Cuiyun Han, Jinchuan Zhang, Xinyu Li, Guojin Xu, Weihua Peng, and Zengfeng Zeng. 2022. Duee-fin: A large-scale dataset for document-level event extraction. In *Natural Language Processing and Chinese Computing*, pages 172–183, Cham. Springer International Publishing.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv: Learning*.

I Hsu, Kuan-Hao Huang, Elizabeth Boschee, Scott Miller, Prem Natarajan, Kai-Wei Chang, Nanyun Peng, et al. 2021. Degree: A data-efficient generative event extraction model. *arXiv preprint arXiv:2108.12724*.

Kung-Hsiang Huang, Sam Tang, and Nanyun Peng. 2021. Document-level entity-based extraction as template generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5257–5269, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.