Nested Event Extraction upon Pivot Element Recognition

Weicheng Ren[1,2], **Zixuan Li**[2*], **Xiaolong Jin**[1,2*], **Long Bai**[2], **Miao Su**[1,2], **Yantao Liu**[1,2], **Saiping Guan**[2], **Jiafeng Guo**[1,2], **Xueqi Cheng**[1,2]
[1]School of Computer Science and Technology, University of Chinese Academy of Sciences;
[2]Key Lab of Network Data Science and Technology,
Institute of Computing Technology, Chinese Academy of Sciences.
{renweicheng21b, lizixuan, jinxiaolong, bailong18b, sumiao22z}@ict.ac.cn
{liuyantao22s, guansaiping, guojiafeng, cxq}@ict.ac.cn

Abstract

Nested Event Extraction (NEE) aims to extract complex event structures where an event contains other events as its arguments recursively. Nested events involve a kind of Pivot Elements (PEs) that simultaneously act as arguments of outer-nest events and as triggers of inner-nest events, and thus connect them into nested structures. This special characteristic of PEs brings challenges to existing NEE methods, as they cannot well cope with the dual identities of PEs. Therefore, this paper proposes a new model, called PerNee, which extracts nested events mainly based on recognizing PEs. Specifically, PerNee first recognizes the triggers of both inner-nest and outer-nest events and further recognizes the PEs via classifying the relation type between trigger pairs. The model uses prompt learning to incorporate information from both event types and argument roles for better trigger and argument representations to improve NEE performance. Since existing NEE datasets (e.g., Genia11) are limited to specific domains and contain a narrow range of event types with nested structures, we systematically categorize nested events in the generic domain and construct a new NEE dataset, called ACE2005-Nest. Experimental results demonstrate that PerNee consistently achieves state-of-the-art performance on ACE2005-Nest, Genia11, and Genia13. The ACE2005-Nest dataset and the code of the PerNee model are available at `https://github.com/waysonren/PerNee`.

Keywords: Information Extraction, Corpus, Text Mining, Nested Event Extraction

1. Introduction

Event Extraction (EE), as an important task in information extraction, aims to extract event triggers and their corresponding arguments from sentences. Traditional EE implicitly assumes that all events in the same sentence have flat structure, thus called Flat Event Extraction (FEE). However, there also exists a kind of nested structures where an event contains other events as its arguments recursively. Therefore, Nested Event Extraction (NEE) as a new information extraction task has recently attracted attention (Trieu et al., 2020; Cao et al., 2022). Figure 1 (a) and (b) illustrate two examples of both flat and nested events, correspondingly. NEE holds immense importance in attaining a profound semantic understanding and acquiring a comprehensive perspective of the event structure.

In the case of nested events, events are connected via a kind of special elements that simultaneously act as arguments of outer-nest events and as triggers of inner-nest events. This kind of elements play as pivots in the nested event structures, thus called Pivot Elements (PEs) in this paper. As shown in Figure 1(b), "pay" is a PE, which serves as the trigger of the inner-nest event `Transfer-Ownership` and as an argument of the outer-nest event `Intention`. Through the PE "pay", these two events are connected to form a nested event structure. Therefore, the key for the NEE task is to recognize this kind of PEs.

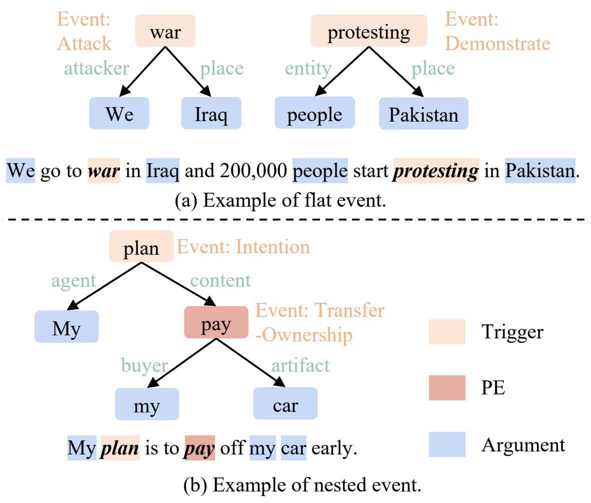

Event: Attack
war
attacker
place
We
Iraq
protesting
Event: Demonstrate
entity
place
people
Pakistan
We go to *war* in Iraq and 200,000 people start *protesting* in Pakistan.
(a) Example of flat event.
plan
Event: Intention
agent
content
My
pay
Event: Transfer -Ownership
buyer
artifact
my
car
My *plan* is to *pay* off my car early.
(b) Example of nested event.
Trigger
PE
Argument

Figure 1: Examples of flat (a) and nested (b) events.

However, the dual identities of PEs present challenges to existing NEE methods (Lin et al., 2020; Cao et al., 2022). These methods typically employ two separate modules to extract triggers and arguments, and recognize those overlapping ones as PEs. However, due to their more trigger-like

* Corresponding authors

characteristics of PEs, it is difficult for the argument extraction module to recognize them as the arguments of outer-nest events, which affects the performance of those existing methods on NEE.

To address this challenge, we propose PerNee, a novel model for the NEE task via better recognizing PEs. Unlike existing methods, PerNee transfers the identification problem of the argument identities of PEs to a classification problem of relations between trigger pairs within the same sentences. Specifically, PerNee utilizes the label names of event types and argument roles as prompts, which are prepended to the sentences. It then employs a BERT-based network to encode the sentences along with these prompts, generating contextual representations enriched with the information of event types and argument roles. Next, PerNee recognizes triggers and regular arguments (i.e., entities that are definitely not PEs) by employing two separate Feedforward Neural Networks (FNNs) combined with a Conditional Random Field (CRF) layer. Finally, PerNee identifies every regular argument corresponding to its trigger and further determines its role by generating and classifying pairs between triggers and regular arguments using an FNN. Simultaneously, by generating and classifying the pairs of triggers based on another FNN, it recognizes from the set of triggers, if any, every PE as well as the trigger of its corresponding outer-nest event and its role therein. By so doing, the nested event structure contained in the input sentence is identified.

There are several event extraction datasets containing nested events (e.g., Genia11 (Kim et al., 2011), Genia13 (Kim et al., 2013)). However, these existing datasets primarily focus on the medical domain and have a narrow range of event types that can introduce nested structures. For instance, in Genia11, only some of the `Regulation` events exhibit nested structures. In contrast, the generic domain contains a diverse array of event types that can introduce nested events, such as `Intention`, `Belief`, and `Statement`. To address these limitations, we systematically categorize nested events in the generic domain into different types and create a new NEE dataset, ACE2005-Nest, based on the widely used benchmark dataset ACE2005 for FEE. ACE2005-Nest contains 14 event types that can introduce nested structures in the generic domain.

Our contributions can be summarized as follows:

- We propose PerNee for the NEE task, which extracts nested events mainly based on recognizing PEs. By classifying the relations between trigger pairs, PerNee significantly enhances the accuracy of PE extraction.

- We systematically categorize nested events in the generic domain and construct a new NEE dataset, ACE2005-Nest, which can serve as a valuable resource to advance the NEE task in the generic domain.

- Experimental results demonstrate that the PerNee model consistently outperforms existing baselines on ACE2005-Nest, Genia11, and Genia13, demonstrating its effectiveness in both FEE and NEE tasks.

2. Related Work

2.1. Nested Event Extraction

Some existing studies tackle NEE using methods actually for overlapping events (Yang et al., 2019; Li et al., 2020; Sheng et al., 2021; Cao et al., 2022), as NEE can be seen as a specific type of overlapping events, where triggers and arguments overlap. For example, Cao et al. (2022) proposed OneEE to address both overlapping and nested events. PEs are recognized in both the trigger recognition module and the argument recognition module, which handles the overlapping issue between triggers and arguments, thereby addressing NEE. In a similar manner, some existing FEE methods (Nguyen and Nguyen, 2019; Raffel et al., 2020; Wadden et al., 2019; Lin et al., 2020; Lu et al., 2022; Shi et al., 2023) can be adapted to address NEE by treating PEs as both triggers and regular arguments and recognizing the overlapping ones as PEs.

However, these methods face difficulties in coping with the dual identities of PEs. They simply treat PEs as regular arguments and extract them within the argument extraction module, neglecting their trigger-like characteristics, which brings challenges to argument extraction.

2.2. NEE Datasets

In the medical domain, there are several NEE datasets available. Genia11 (Kim et al., 2011) is a medical domain event extraction dataset, containing a total of 9 event types. Among these event types, `Regulation`, `Positive Regulation`, and `Negative Regulation` are 3 event types that can involve other events as arguments. Based on Genia11, Genia13 (Kim et al., 2013) introduces additional event types such as `Phosphorylation` that can introduce nested events. Besides, in the Cancer Genetics dataset (Pyysalo et al., 2013) and Pathway Curation dataset (Ohta et al., 2013), the `Regulation` event type is prominent for introducing nested event structures.

Above all, in existing NEE datasets, nested events are mainly concentrated in limited event types like `Regulation`, with a predominant focus on the medical domain. However, in the generic domain, nested events are widespread with a diverse range of types, indicating a need for generic domain NEE datasets.

3. Problem Formulation

Given a sentence X, the NEE task aims to extract the events therein, including their triggers and arguments, and further identify the specific roles of all extracted arguments and, if any, the nested structures between events. Let $E = \{e_1, e_2, ..., e_k\}$ be the set of events contained in X. Each event e_i ($1 \leq i \leq k$) is represented as a 4-tuple (τ_i, t_i, A_i, R_i), where τ_i is its type and t_i is its trigger associated with τ_i, indicating its occurrence; A_i and R_i are the sets of its arguments and their corresponding roles, respectively. For each e_i, the l^{th} argument $a_i^l \in A_i$ is associated with a corresponding role in R_i.

The nested event structures, if any, in E that can essentially be characterized by a PE set $P = \{t_i | \exists j, t_i \in A_j\}$. In this view, the NEE task involves the following subtasks:

Trigger Recognition: Given the sentence X, it is to recognize all triggers $T = \{t_1, t_2, ..., t_k\}$ therein and further determine their respective event types.

Regular Argument Extraction: Given the sentence X and a trigger $t_i \in T$, this subtask is to extract the set of its arguments excluding PEs, $A_i = \{a_i^1, a_i^2, ..., a_i^l, ...\}$ and further determine their respective roles in R_i.

Pivot Element Recognition: Given the sentence X and a trigger $t_i \in T$, the goal is to identify whether or not there exists another trigger $t_j \in T$, t_i is one of its argument and, if so, further determine its role.

4. The PerNee Model

In this section, we will introduce the framework of PerNee. As shown in Figure 2, it mainly contains five modules. The text encoder encodes the sentence with prompts to obtain the representations of all words therein. Based on these representations, the trigger recognizer and the regular argument recognizer recognize triggers and regular arguments, respectively. Next, the pivot element recognizer is adopted to recognize, if any, all PEs. Based on the extracted elements, the structure decoder explores possible event structures using beam search to generate events with the highest global score.

4.1. The Text Encoder

This module aims to obtain the representation for each word within a given sentence X. In order to acquire the word representation enriched with a contextual understanding of the event types and argument roles, we prepend the label names of all event types and argument roles as prompts to X. This, in turn, enhances the model's perception of event schema. Some related papers (Lu et al., 2022; Wang et al., 2022; Lou et al., 2023) have demonstrated that introducing label information of event types and argument roles can improve the ability of the model to perceive the information to be extracted.

Following Brown et al. (2020); Schick and Schütze (2020), we use [EVENT] and [ROLE] as placeholder separators (abbreviated as [$\mathcal{T}$] and [R] hereafter) to concatenate the label names of event types τ_i and argument roles r_i. Finally, the input of the text encoder is:

$$[\mathcal{T}]\tau_1[\mathcal{T}]\tau_2...[\mathcal{T}]\tau_n[R]r_1[R]r_2...[R]r_n[SEP]X$$

Next, the input is encoded through a pre-trained BERT model (Devlin et al., 2018). As BERT tokenizes each word into several subword pieces (e.g., “blowdryers” → “blow”, “##dr”, “##yers”), we obtain its representation by computing the average of the representations of those corresponding subword pieces.

Finally, this module generates the representations of all words in X, denoted as $\mathbf{H} = \{\mathbf{h_1}, \mathbf{h_2}, ..., \mathbf{h_n}\}$.

4.2. The Trigger Recognizer

This module aims to recognize triggers, which contains two steps: an identification step to identify triggers and a classification step to obtain, for each trigger, the label scores of corresponding event types.

The trigger identification can be formulated as a sequence labeling problem. Specifically, the module takes word representations in each sentence as its input and calculates a score vector for each word using an FNN. Each value in the vector represents the score of a specific tag corresponding to the BIO tag schema. To capture the dependencies among predicted tags, a CRF layer is utilized to ensure the validity of certain tag sequences. For instance, an *I-Intention* tag should not follow a *B-Attack* tag. The trigger tag sequence corresponding to the sentence is obtained as $\hat{\mathbf{z}}^{\mathbf{t}}$. Inspired by Lample et al. (2016), the objective is to maximize the log-likelihood of the gold-standard tag sequence. Thus, the loss of trigger identification is defined as:

$$\mathcal{L}_1^t = \log \sum_{\hat{\mathbf{z}}^t \in Z^t} e^{s(\mathbf{H}, \hat{\mathbf{z}}^t)} - s(\mathbf{H}, \mathbf{z}^t), \qquad (1)$$

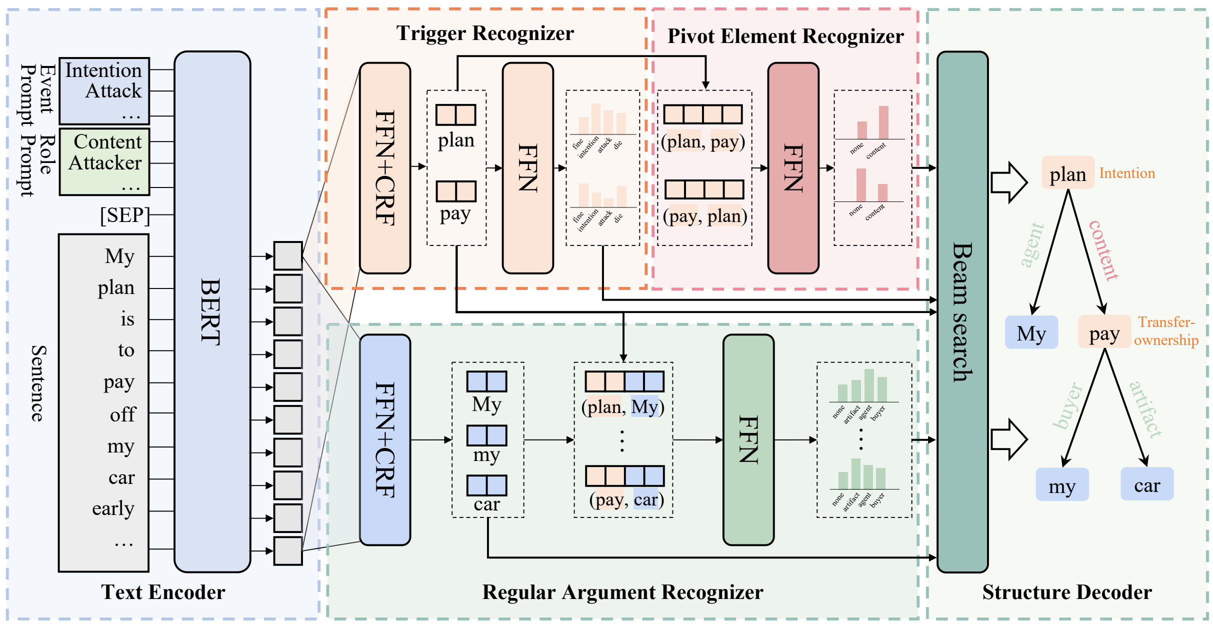


Figure 2: The overall framework of the PerNee model.

where s denotes the tag sequence scoring function, $\mathbf{z}^t$ represents the golden trigger tag sequence, and Z^t represents the set of all possible trigger tag sequences for a given sentence.

In the classification step, since the identified triggers may contain several words, the representation of the i^{th} identified trigger is obtained by averaging its word representations, denoted as $\mathbf{t_i}$. To obtain its corresponding event type, another FNN is employed to calculate type label scores as $\hat{\mathbf{y}}_i^t = FNN(\mathbf{t_i})$.

For trigger classification, the objective is to minimize the following cross-entropy loss:

$$\mathcal{L}_2^t = -\frac{1}{N^t}\sum_{i=1}^{N^t} \mathbf{y}_i^t \log \hat{\mathbf{y}}_i^t, \qquad (2)$$

where N^t and $\mathbf{y}_i^t$ represent the number of triggers and the true label vector, respectively. Therefore, the training loss of the trigger recognizer is defined as:

$$\mathcal{L}^t = \mathcal{L}_1^t + \mathcal{L}_2^t. \qquad (3)$$

4.3. The Regular Argument Recognizer

Considering the trigger-like characteristics of PEs and their notable differences from regular arguments (i.e., entities), jointly recognizing PEs and regular arguments may affect the performance of argument recognition. Therefore, this regular argument recognizer focuses only on extracting regular arguments. It involves two steps: an identification step to extract regular arguments and a classification step to obtain the label scores of role types.

In the identification step, this module employs an FNN followed by a CRF layer to generate tag sequences for regular arguments. Similar to Equation 1, the loss of regular argument identification is defined as:

$$\mathcal{L}_1^a = \log \sum_{\hat{\mathbf{z}}^a \in Z^a} e^{s(\mathbf{H},\hat{\mathbf{z}}^a)} - s(\mathbf{H},\mathbf{z}^a), \qquad (4)$$

where $\hat{\mathbf{z}}^\mathbf{a}$, $\mathbf{z}^a$, and Z^a represent the predicted regular argument tag sequence, the golden regular argument tag sequence, and the set of all possible regular argument tag sequences for a given sentence, respectively.

In the classification step, role types of arguments are determined by establishing relations between triggers and regular arguments. Given a trigger and a regular argument, the representation of the trigger-argument pair is calculated by concatenating the representations of the identified trigger and regular argument, denoted as $[\mathbf{t_i};\mathbf{a_j}]$. Then, another FNN is employed to calculate the score vector of the trigger-argument pair, denoted as $\hat{\mathbf{y}}_{i,j}^a = FNN([\mathbf{t_i};\mathbf{a_j}])$, which represents the role type scores for the identified regular argument.

For trigger-argument pair classification, the objective is to minimize the following cross-entropy loss:

$$\mathcal{L}_2^a = -\frac{1}{N^a}\sum_{i=1}^{N^a} \mathbf{y}_{i,j}^a \log \hat{\mathbf{y}}_{i,j}^a, \qquad (5)$$

where N^a and $\mathbf{y}_{i,j}^a$ represent the number of trigger-argument pairs and the true label vector, respectively. Therefore, the training loss of the regular argument recognizer is defined as:

$$\mathcal{L}^a = \mathcal{L}_1^a + \mathcal{L}_2^a. \tag{6}$$

4.4. The Pivot Element Recognizer

The nested events arise when one event serves as an argument of another event. To recognize nested events, it is crucial to recognize PEs. However, PEs bring challenges to existing methods due to their dual identities. Considering the trigger-like characteristics of PEs, PerNee first identifies the trigger identities of PEs via the trigger recognizer as mentioned in Section 4.2. Then, in the pivot element recognizer, PerNee further identifies the argument identities of PEs by transferring the identification problem to a classification problem of the relations between trigger pairs within the same sentence. By doing so, recognizing PEs can be transferred to discovering the argument relations between the trigger pairs, hereby helping the model avoid confusion arising from the dual identities of PEs.

Specifically, given the set T of all the extracted triggers in a sentence, PerNee first generates the candidate trigger pairs $\{(t_i, t_j) | t_i, t_j \in T\}$. Note that if the trigger pair (t_i, t_j) is added to candidate trigger pairs, the trigger pair (t_j, t_i) is also included. Then, the representations of the triggers are concatenated to form the representation of the trigger pair, represented as $[\mathbf{t_i}; \mathbf{t_j}]$. An FNN is employed to calculate the score vector of the trigger pair, denoted as $\hat{\mathbf{y}}_{i,j}^p = FNN([\mathbf{t_i}; \mathbf{t_j}])$, which represents the role type scores for the PE in the corresponding outer-nest event.

For trigger-trigger pair classification, the objective is to minimize the following cross-entropy loss:

$$\mathcal{L}^p = -\frac{1}{N^p} \sum_{i=1}^{N^p} \mathbf{y}_{i,j}^p \log \hat{\mathbf{y}}_{i,j}^p, \tag{7}$$

where N^p and $\mathbf{y}_{i,j}^p$ represent the number of trigger-trigger pairs and the true label vector, respectively.

Finally, the joint objective function during training is optimized by minimizing the following loss function:

$$\mathcal{L} = \mathcal{L}^t + \mathcal{L}^a + \mathcal{L}^p. \tag{8}$$

4.5. The Structure Decoder

In the prediction stage, we first extract the elements and their corresponding score vectors based on the above modules and subsequently employ a beam search-based strategy to decode the globally optimal event structure, following (Lin et al., 2020). This approach aims to achieve global best extraction results instead of local ones. In this context, event structures are represented as graphs in which triggers and regular arguments serve as nodes, connected by edges denoting their relations. The score for a given graph g is computed as:

$$score(g) = \sum_{i=0}^{N^v} s(v_i) + \sum_{i=0}^{N^\ell} s(\ell_i), \tag{9}$$

where $s(v_i)$, $s(\ell_i)$ represent the scores of node types and edge types, and N^v, N^ℓ denote the number of nodes and edges. Note that all scores are normalized within the nodes or edges.

Beam search is used to iteratively extend nodes and edges with a beam set of size θ. The extension process involves selecting the top k most likely labels for both nodes and edges. After extending nodes and edges, a set of candidate graphs is obtained, denoted as $G = \{g_1, g_2, ..., g_n\}$. The graph with the highest score is then selected from this set:

$$g_{best} = \arg\max_{g_k \in G}(score(g_k)), k = 1, 2, ..., n. \tag{10}$$

5. The ACE2005-Nest Dataset

To address the limitations of existing NEE datasets, such as Genia11, which are domain-specific and have a limited range of event types that can introduce nested structures, we construct a new NEE dataset in the generic domain, building upon the ACE2005 dataset[1] (a widely used source for FEE). It contains 8 event categories, 33 sub-categories, and 35 argument roles, derived from news, broadcasts, and conversations. Based on ACE2005, we discover extra event types that can introduce nested structures and their associated argument roles. We then annotate instances of these new event types based on the original events.

5.1. Nested Event Schema Discovery

Building upon the existing event annotations in the ACE2005 dataset, we discover the nested event schema as follows: First, triggers that may cause nested structures are identified; Then, the inner-nest events (i.e., PEs) are identified as well as their relevant arguments, such as agent and time. After that, we build up the connections between the triggers and their respective arguments.

To categorize event types and determine the frame semantics descriptions, some established resources are referred to, including WordNet (Fellbaum, 2010), FrameNet (Baker et al., 1998), and FactBank (Saurí and Pustejovsky, 2009). These resources provide valuable insights into verb classification and frame semantics. With this knowledge, we systematically define various types of triggers that have the potential to introduce nested structures. Based on our analysis, these triggers can be

[1] https://catalog.ldc.upenn.edu/LDC2006T06

classified into 7 categories and 14 sub-categories, as shown in Table 1.

Event Types	Subtypes	Trigger Examples
Statement	Oral	say, speak
	Written	write, report
Idea	Belief	believe, think
	Attitude	oppose, agree
	Doubt	wonder, doubt
Knowledge	Aware	know, aware
	Perception	see, hear
	Inference	mean, indicate
Sentiment	Preference	like, hate
	Emotion	worry, fear
Instruction	Command	order, instruct
	Demand	require, ask
Judgement	-	accuse, blame
Intention	-	plan, want

Table 1: Event types in the generic domain that can introduce nested events and their corresponding trigger examples.

5.2. Data Analysis

ACE2005-Nest is divided into the train, dev, and test sets following pre-processing of Wadden et al. (2019). We conduct an analysis of ACE2005-Nest, along with the other two NEE datasets, Genia11 and Genia13, as shown in Table 2. It reveals that in ACE2005-Nest, approximately 25% of the sentences with events contain nested events, while in Genia11 and Genia13, the account is 39% and 49%. Besides, ACE2005-Nest significantly surpasses Genia11 and Genia13 in terms of the number of event types capable of introducing nested events. While Genia11 and Genia13 only have 3 and 5 such event types, ACE2005-Nest has 14, indicating that ACE2005-Nest exhibits greater diversity in event types capable of introducing nested events.

Additionally, we conduct a detailed analysis of the proportions of event types that may introduce nested events, as shown in Figure 3. The results show that `Statement:Oral`, `Idea:Belief` and `Intention` are the top three event types that may introduce nested structures with the highest number of occurrences, accounting for 45.54%, 13.64%, and 13.64%, respectively.

Besides, the ACE2005-Nest dataset also has some shortcomings: (1) The coverage breadth of event types capable of introducing nested events

		#S.	#S.E.	#S.N.E.	#E.T.	#E.T.N.
ACE2005-Nest	Train	19,204	3,342	778	47	14
	Dev	901	327	103		
	Test	676	293	112		
Genia11	Train	8,722	3,707	1,464	9	3
	Dev	1,090	474	167		
	Test	1,091	456	173		
Genia13	Train	4,000	1,574	795	13	5
	Dev	500	189	90		
	Test	500	201	85		

Table 2: Statistics of the datasets. “#S.”, “#S.E.”, “#S.N.E.”, “#E.T.”, and “#E.T.N.” denote the numbers of sentences, sentences with events, sentences with nested events, event types, and event types that can introduce nested events, respectively.

is insufficient. Nested events are a common phenomenon in natural language, and our current classification is based on statistical analysis during the annotation process and referencing some resources such as WordNet (Fellbaum, 2010), FrameNet (Baker et al., 1998), and FactBank (Saurí and Pustejovsky, 2009). However, this is still a preliminary exploration, and the relevant definitions need further refinement and supplementation. (2) ACE2005-Nest is annotated based on ACE2005. Due to inherent noise in ACE2005 and variations in the standards among annotators during the labeling process, additional noise may be introduced.

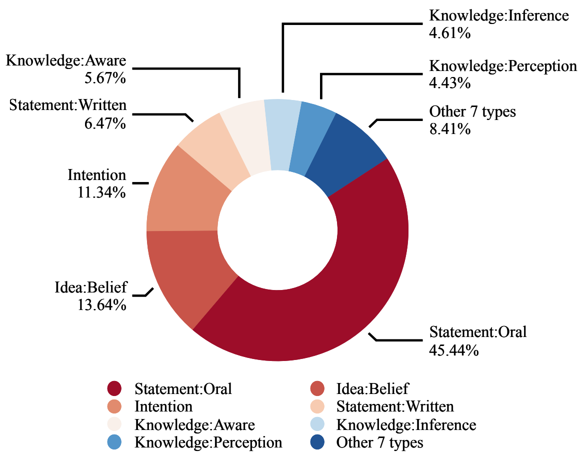


Figure 3: Analysis of the proportions of event types capable of introducing nested events.

6. Experiments

6.1. Experimental Setup

Datasets. We conduct experiments on ACE2005-Nest, Genia11 and Genia13. The statistics of datasets are shown in Table 2. Since the golden annotations for the test set of Genia11[2] and Genia13[3] are not released, we follow Cao et al. (2022) to split the train and dev sets into train/dev/test sets with a proportion of 8:1:1 randomly.

Evaluation Metrics. For evaluation, we use the same criteria of previous work (Cao et al., 2022) and two new criteria. The traditional criteria are as follows: 1) Trigger Identification (**TI**): A trigger is correctly identified when its span matches the golden label; 2) Trigger Classification (**TC**): A trigger is correctly classified when it is identified correctly and its event type matches the golden label; 3) Argument Identification (**AI**): An argument is correctly identified when its event type matches the golden label and its predicted span matches the golden label; 4) Argument Classification (**AC**): An argument is correctly classified when it is identified correctly, and its role type matches the golden label. Besides, to evaluate the performance of PE recognition, two new criteria are added: 5) Pivot Element Identification (**PEI**): A PE is correctly identified when its outer-nest event type is recognized correctly, and its predicted span matches the golden label; 6) Pivot Element Classification (**PEC**): A PE is correctly classified when it is identified correctly, and its role in outer-nest events matches the golden label. We report the average F-measure (F1) scores across five runs for each of these metrics.

Baselines. For evaluation, we choose OneIE (Lin et al., 2020), UIE (Lu et al., 2022), PLMEE (Yang et al., 2019), CasEE (Sheng et al., 2021), HDGSE (Shi et al., 2023), and OneEE (Cao et al., 2022) as the baseline models. OneIE, UIE, and HDGSE were initially designed for FEE and have exhibited robust performance. We adapt these models to NEE by considering spans that are simultaneously identified as triggers and regular arguments as PEs. Furthermore, aligning with previous work (Cao et al., 2022), PLMEE, CasEE, and OneEE are selected as our baselines that have demonstrated strong performance in overlapping and nested event extraction tasks.

6.2. Results on Extracting All Events

Table 3 presents the F1 scores for extracting all events on ACE2005-Nest, Genia11, and Genia13. It can be observed that PerNee outperforms all the baselines. The existence of PEs in the dataset confuses the baselines and affects their performance on all the subtasks. The PerNee model effectively mitigates the impact of dual identities associated with PEs through a two-step process: 1) Firstly, it identifies the candidate PEs within triggers using the trigger recognizer. 2) It subsequently determines the argument identities of the PEs via classifying the relations between trigger pairs with the pivot element recognizer. By disentangling the two identities of PEs through the above two modules, the model achieves enhanced precision in recognizing PEs. Besides, the utilization of label information of event types and argument roles as prompts can help the model get better representations. Thereby, PerNee leads to better performance in most NEE tasks.

In Table 3, the improvement in AI and AC on ACE2005-Nest is significant while the enhancement on Genia11 and Genia13 is marginal. This may be attributed to the fewer outer-nest event types and a lower proportion of PEs in Genia11 and Genia13. (1) In the Genia11 and Genia13 datasets, there are only 3 and 5 outer-nest event types, with the majority falling into the “Regulation” type, while ACE2005-Nest contains 14 outer-nest event types. This implies that the extraction of nested events in Genia11 and Genia13 is relatively easier compared to ACE2005-Nest. (2) the number of PEs accounts for only 19% and 20% of all arguments in Genia11 and Genia13, respectively. The AI and AC metrics in Table 3 reflect the extraction performance of all arguments. As our model is primarily optimized for PE recognition, the overall improvement in AI and AC is not significant.

6.3. Results on Extracting Nested Events

To further verify that the improvement on ACE2005-Nest, Genia11, and Genia13 are indeed attributed to the enhancement of NEE, we evaluate the performance of PerNee only on the sentences that contain at least one nested event structure in ACE2005-Nest, Genia11, and Genia13. The results are presented in Table 4. It can be observed that PerNee significantly surpasses other baselines on all the subtasks on these three datasets, which convincingly verifies its effectiveness. To further analyze the reason, the F1 scores on PEI and PEC are also reported in Table 4. It can be noticed that PerNee achieves significant improvement on the PEI and PEC because the designed trigger recognizer and pivot element recognizer can better process the dual identities of PEs.

6.4. Ablation Studies

To verify the effectiveness of PE recognition for NEE, we treat PEs as regular arguments and recognize them using the same module as the regular argument recognizer rather than the designed PER module in PerNee. The results are denoted as “w/o PER” in Table 5. It shows that removing PER leads to a decrease, particularly in AI, AC, PEI, and PEC metrics. It implies that emphasizing PE recognition and designing a specialized PER module can improve the performance of NEE.

[2]https://2011.bionlp-st.org/home/genia-event-extraction-genia

[3]https://bionlp.dbcls.jp/projects/bionlp-st-ge-2013/wiki/Overview

	ACE2005-Nest				Genia11				Genia13			
	TI	TC	AI	AC	TI	TC	AI	AC	TI	TC	AI	AC
PLMEE (Yang et al., 2019)	60.8	58.5	45.2	44.3	65.4	63.5	57.1	56.8	72.6	69.9	65.7	64.9
CasEE (Sheng et al., 2021)	71.9	67.2	47.8	46.3	70.8	68.2	60.2	59.8	78.9	75.1	67.8	67.1
UIE (Lu et al., 2022)	70.0	66.0	50.7	48.2	70.4	68.5	60.7	59.2	75.5	72.2	66.7	66.3
HDGSE (Shi et al., 2023)	72.3	69.5	49.9	46.9	71.5	69.2	61.3	59.9	77.1	76.2	69.4	67.7
OneIE (Lin et al., 2020)	70.9	68.3	53.1	51.3	71.0	68.8	58.7	57.4	78.7	75.2	66.4	64.6
OneEE (Cao et al., 2022)	72.7	69.9	51.3	48.4	71.3	69.2	61.9	60.8	78.6	76.4	70.8	68.1
PerNee	**72.8**	**70.0**	**55.5**	**53.8**	**71.8**	**69.5**	**62.3**	**61.2**	**79.6**	**76.7**	**70.9**	**68.9**

Table 3: Experimental results of extracting all events on ACE2005-Nest, Genia11, and Genia13, respectively.

	TI	TC	AI	AC	PEI	PEC
• **ACE2005-Nest**						
PLMEE	66.3	64.2	46.2	45.4	28.8	28.8
CasEE	78.3	75.7	53.5	50.9	46.1	46.1
UIE	78.0	75.0	54.3	51.0	46.3	46.3
HDGSE	78.2	75.9	53.2	50.2	43.2	43.2
OneIE	74.0	71.6	49.1	48.1	24.9	24.9
OneEE	78.5	76.9	54.4	50.2	46.8	46.8
PerNee	**79.0**	**77.5**	**55.7**	**54.6**	**49.2**	**49.2**
• **Genia11**						
PLMEE	70.4	68.1	65.4	64.8	54.7	53.7
CasEE	75.0	72.2	64.4	63.8	54.0	53.2
UIE	73.5	69.5	63.5	62.9	55.1	54.4
HDGSE	75.1	72.5	64.2	63.3	53.6	52.9
OneIE	73.7	69.8	64.0	63.4	53.7	52.2
OneEE	75.4	73.0	65.0	63.9	57.3	55.7
PerNee	**75.7**	**73.2**	**66.6**	**65.9**	**58.8**	**56.9**
• **Genia13**						
PLMEE	72.8	70.4	69.0	67.5	65.7	65.2
CasEE	78.8	75.4	68.3	67.4	66.2	65.9
UIE	78.4	75.2	71.4	66.8	65.8	65.8
HDGSE	79.1	76.5	70.1	67.6	65.9	65.6
OneIE	79.0	76.0	69.2	66.1	63.9	63.5
OneEE	78.6	**76.9**	70.4	70.3	66.7	66.7
PerNee	**79.6**	**76.9**	**72.2**	**70.9**	**68.1**	**68.1**

Table 4: Experimental results of extracting nested events on ACE2005-Nest, Genia11, and Genia13, respectively.

To further verify the effectiveness of identifying PEs via trigger-trigger relation classification, we treat PE recognition as a sequence labeling problem by utilizing a specific FNN with CRF, denoted as "repl. PER". It illustrates a significant decline of 27.2% in both PEI and PEC metrics. This indicates the design of the PER module in PerNee is more effective in recognizing PEs.

Besides, to verify the effectiveness of prompts, we remove them (denoted as "w/o. pmt"), resulting in performance decreases across all subtasks. It suggests that label information related to event types and argument roles proves beneficial for NEE.

	TI	TC	AI	AC	PEI	PEC
PerNee	72.8	70.0	55.5	53.8	41.5	41.5
w/o PER	72.1	69.6	54.0	52.4	35.5	35.5
repl. PER	64.8	62.2	42.5	41.2	14.3	14.3
w/o pmt.	71.4	69.4	53.4	52.0	40.1	40.1

Table 5: Ablation studies on ACE2005-Nest.

6.5. Detailed Analysis

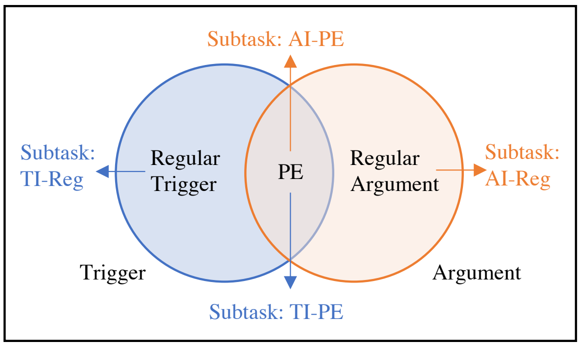
To verify the motivation that identifying the dual identities of PEs is crucial for NEE, we compare the F1 scores of PerNee with OneIE[4] on the following four subtasks, as shown in Figure 4 (a): 1) PE-TI: Identifying the trigger identities of PEs; 2) Identifying regular triggers; 3) AI-PE: Identifying the argument identities of PEs; 4) AI-Reg: Identifying regular arguments. From results shown in Figure 4 (b), we can observe that:

1) Identifying the argument identities of PEs is more challenging than the trigger identities of PEs. Both OneIE and PerNee achieve high F1 scores in PE-TI (81.3% and 84.8%) but low F1 scores in PE-AI (40.1% and 66.2%). It suggests that PEs possess more trigger-like characteristics, making the argument identities of PEs more difficult to be recognized;

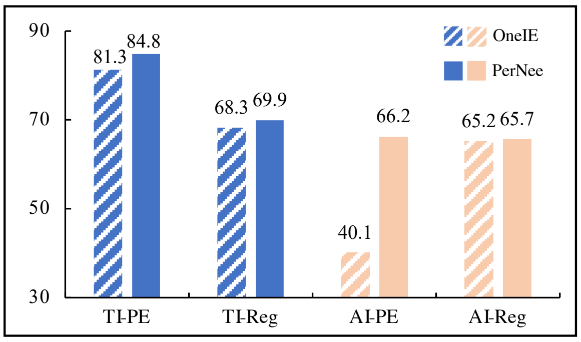
2) For OneIE, identifying the argument identities of PEs is more difficult than identifying regular arguments. The F1 score on PE-AI (40.1%) is significantly lower than that on Reg-AI (65.2%). It indicates deploying the same module to identify the argument identities of PEs and regular arguments is ineffective;

3) PerNee significantly improves the argument identification of PEs. PerNee outperforms OneIE

[4]Here, we adopt OneIE for comparison, as it achieves the second best performance on ACE2005-Nest and allows for the separation of the TI and AI subtasks.



(a) Four subtasks for detail analysis.



(b) Comparison between OneIE and PerNee on the four subtasks.

Figure 4: Detailed analysis on ACE2005-Nest.

on all four subtasks. Especially on PE-AI, it increases from 40.1% to 66.2%, indicating the effectiveness of the PER module;

4) PerNee exhibits a much smaller performance gap in identifying regular arguments and the argument identities of PEs compared to OneIE. The gap between AI-PE and AI-Reg is 0.5% for PerNee, while the gap for OneIE is 26.1%. It indicates that, with the PER module, the AI-PE subtask is not that difficult and can achieve comparable performance with the AI-Reg subtask.

6.6. Case Study

To demonstrate the importance of Pivot Element (PE) recognition in NEE, two cases are selected from the ACE2005-Nest dataset. As shown in Table 5, PerNee correctly extracted the nested events while OneIE, the second-best model in the ACE2005-Nest dataset, failed.

In these cases, PerNee correctly recognized the PEs ("destroyed" in Case 1, "met" in Case 2) and consequently extracted nested events accurately. On the other hand, OneIE successfully recognized most arguments but failed to recognize the argument identities of PEs, resulting in the inability to extract nested events. This highlights the effectiveness of our specifically designed PE recognition module, which enhances the performance of nested event extraction.

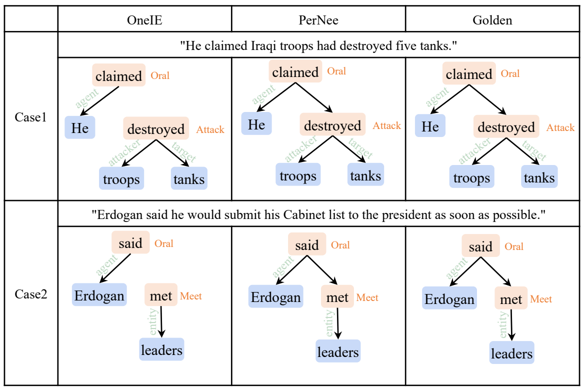

OneIE
PerNee
Golden
Case1
"He claimed Iraqi troops had destroyed five tanks."
claimed Oral
agent
He
destroyed Attack
attacker
target
troops
tanks
Case2
"Erdogan said he would submit his Cabinet list to the president as soon as possible."
said Oral
agent
Erdogan
met Meet
entity
leaders

Figure 5: Case study on the importance of PE recognition for NEE on ACE2005-Nest.

7. Conclusions

In this paper, we presented PerNee, a new model designed to tackle the Nested Event Extraction (NEE) task by focusing on recognizing Pivot Elements (PEs). PEs are crucial in connecting outer-nest and inner-nest events within a nested structure. Since PEs have dual identities (i.e., trigger identities and argument identities), we employ a trigger recognizer to recognize the trigger identities and utilize a PE recognizer to recognize the argument identities, thus connecting events into nested structures. To enhance NEE performance, the semantic information of event types and argument roles is leveraged through prompt learning. Additionally, we introduced ACE2005-Nest, a new NEE dataset in the generic domain, which systematically categorizes nested events therein. Experimental results demonstrate that the proposed model consistently achieves state-of-the-art performance on ACE2005-Nest, Genia11, and Genia13. Moreover, ablation studies validate the effectiveness of the PE recognizer module and prompts. In the future, we will try to optimize the PE recognizer module and extend our method to handle events with more complex structures, such as multi-level nested events. This will allow us to further advance NEE capabilities and address the challenges posed by intricate event hierarchies.

8. Acknowledgments

The work is supported by the GFKJ Innovation Project, the National Natural Science Foundation of China under grant 62306299, the National Key Research and Development Project of China, the Beijing Academy of Artificial Intelligence under grant BAAI2019ZD0306, the KGJ Project under grant JCKY2022130C039, and the Lenovo-CAS Joint Lab Youth Scientist Project. We appreciate anonymous reviewers for their insightful comments and suggestions.

9. Bibliographical References

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *COLING 1998 Volume 1: The 17th International Conference on Computational Linguistics*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Hu Cao, Jingye Li, Fangfang Su, Fei Li, Hao Fei, Shengqiong Wu, Bobo Li, Liang Zhao, and Donghong Ji. 2022. Oneee: A one-stage framework for fast overlapping and nested event extraction. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1953–1964.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Christiane Fellbaum. 2010. Wordnet. In *Theory and applications of ontology: computer applications*, pages 231–243. Springer.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Fayuan Li, Weihua Peng, Yuguang Chen, Quan Wang, Lu Pan, Yajuan Lyu, and Yong Zhu. 2020. Event extraction as multi-turn question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 829–838.

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

Jie Lou, Yaojie Lu, Dai Dai, Wei Jia, Hongyu Lin, Xianpei Han, Le Sun, and Hua Wu. 2023. Universal information extraction as unified semantic matching. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(11):13318–13326.

Yaojie Lu, Qing Liu, Dai Dai, Xinyan Xiao, Hongyu Lin, Xianpei Han, Le Sun, and Hua Wu. 2022. Unified structure generation for universal information extraction. *arXiv preprint arXiv:2203.12277*.

Trung Minh Nguyen and Thien Huu Nguyen. 2019. One for all: Neural joint modeling of entities and events. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 6851–6858.

Tomoko Ohta, Sampo Pyysalo, Rafal Rak, Andrew Rowley, Hong-Woo Chun, Sung-Jae Jung, Sung-Pil Choi, Sophia Ananiadou, and Jun'ichi Tsujii. 2013. Overview of the pathway curation (pc) task of bionlp shared task 2013. In *Proceedings of the BioNLP shared task 2013 workshop*, pages 67–75.

Sampo Pyysalo, Tomoko Ohta, and Sophia Ananiadou. 2013. Overview of the cancer genetics (cg) task of bionlp shared task 2013. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 58–66.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Roser Saurí and James Pustejovsky. 2009. Factbank: a corpus annotated with event factuality. *Language resources and evaluation*, 43:227–268.

Timo Schick and Hinrich Schütze. 2020. Exploiting cloze questions for few shot text classification and natural language inference. *arXiv preprint arXiv:2001.07676*.

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. *arXiv preprint arXiv:2107.01583*.

Ge Shi, Yunyue Su, Yongliang Ma, and Ming Zhou. 2023. A hybrid detection and generation framework with separate encoders for event extraction. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3163–3180, Dubrovnik, Croatia. Association for Computational Linguistics.

Hai-Long Trieu, Thy Thy Tran, Khoa NA Duong, Anh Nguyen, Makoto Miwa, and Sophia Ananiadou. 2020. Deepeventmine: end-to-end neural nested event extraction from biomedical texts. *Bioinformatics*, 36(19):4910–4917.

David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 2019. Entity, relation, and event extraction with contextualized span representations. *arXiv preprint arXiv:1909.03546*.

Sijia Wang, Mo Yu, Shiyu Chang, Lichao Sun, and Lifu Huang. 2022. Query and extract: Refining event extraction as type-oriented binary decoding. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 169–182, Dublin, Ireland. Association for Computational Linguistics.

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 the 57th annual meeting of the association for computational linguistics*, pages 5284–5294.

10. Language Resource References

Jin-Dong Kim, Yue Wang, Toshihisa Takagi, and Akinori Yonezawa. 2011. Overview of genia event task in bionlp shared task 2011. In *Proceedings of BioNLP shared task 2011 workshop*, pages 7–15.

Jin-Dong Kim, Yue Wang, and Yamamoto Yasunori. 2013. The genia event extraction shared task, 2013 edition-overview. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 8–15.