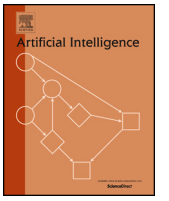
Artificial Intelligence 332 (2024) 104144

Contents lists available at ScienceDirect

Artificial Intelligence

journal homepage: www.elsevier.com/locate/artint



A multi-graph representation for event extraction

Hui Huang [a], Yanping Chen [a,*], Chuan Lin [a], Ruizhang Huang [a], Qinghua Zheng [b], Yongbin Qin [a,*]

[a] Text Computing and Cognitive Intelligence Engineering Research Center of National Education Ministry, State Key Laboratory of Public Big Data, College of Computer Science and Technology, Guizhou University, Guiyang, 550025, China
[b] Department of Computer Science and Technology, Xi'an Jiaotong University, Xi'an 710049, China

ARTICLE INFO

Keywords:
Event extraction
Multigraph
Argument multiplexing
Event representation

ABSTRACT

Event extraction has a trend in identifying event triggers and arguments in a unified framework, which has the advantage of avoiding the cascading failure in pipeline methods. The main problem is that joint models usually assume a one-to-one relationship between event triggers and arguments. It leads to the argument multiplexing problem, in which an argument mention can serve different roles in an event or shared by different events. To address this problem, we propose a multigraph-based event extraction framework. It allows parallel edges between any nodes, which is effective to represent semantic structures of an event. The framework enables the neural network to map a sentence(s) into a structurized semantic representation, which encodes multi-overlapped events. After evaluated on four public datasets, our method achieves the state-of-the-art performance, outperforming all compared models. Analytical experiments show that the multigraph representation is effective to address the argument multiplexing problem and helpful to advance the discriminability of the neural network for event extraction.

1. Introduction

An event is typically defined as a semantic structure composed of a trigger and relevant arguments [1]. The trigger is defined as a word (or a phrase), which indicates the occurrence of an event. The arguments describe *who*, *when*, *where*, *what*, *why* and *how* an event occurred. In information extraction, the task to extract events is usually divided into two subtasks: event detection (extracting event triggers), and argument extraction (recognizing relevant arguments) [2]. Because extracting events is the key to automatically discovery narrative knowledge from plain texts [3], it is an important task in natural language processing, and has received great attention in this field. As a fundamental task, it is widely adopted to support tasks such as information retrieval [4], text summarization [5], question answering [6], and news recommendation [7].

Extracting events from sentences is a challenging task, because events usually have complex semantic structures. The task should identify triggers, arguments, and estimate the semantic relation between them. In the early stages, event extraction is implemented in a pipeline framework [8–11], in which the two subtasks are conducted separately. The problem is that they suffer from the cascading failure problem, where errors of event detection can propagate to the argument extraction, which heavily influences the

* Corresponding authors.
E-mail addresses: gs.hhuang21@gzu.edu.cn (H. Huang), ypench@gzu.edu.cn (Y. Chen), clin@gzu.edu.cn (C. Lin), rzhuang@gzu.edu.cn (R. Huang), qhzheng@mail.xjtu.edu.cn (Q. Zheng), ybqin@gzu.edu.cn (Y. Qin).

https://doi.org/10.1016/j.artint.2024.104144
Received 27 March 2023; Received in revised form 25 April 2024; Accepted 27 April 2024

Available online 3 May 2024

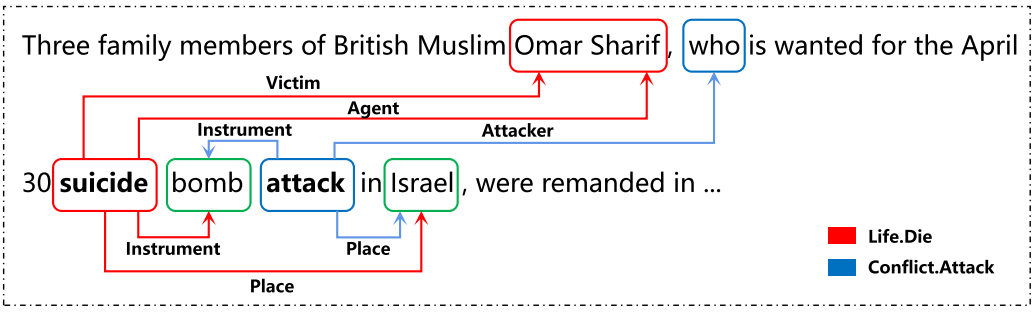


Fig. 1. Examples of argument multiplexing. (For interpretation of the colors in the figure(s) and table(s), the reader is referred to the web version of this article.)

final performance. Furthermore, the two subtasks are optimized independently, which cannot build the interaction between them and easily lead to the local optimization problem.

An important phenomenon of the event extraction is that several events may simultaneously occur in a sentence. In this condition, an argument mention (e.g., a word or a phrase) may act as different arguments shared by different events. We refer this problem as the argument multiplexing. An example of the argument multiplexing is shown in Fig. 1.

The sentence in Fig. 1 contains two events triggered by "suicide" and "attack". They are highlighted by the red color and the blue color, respectively. The two events are annotated with the types $Life.Die$ and $Conflict.Attack$. The $Life.Die$ event has four event arguments indicated by the red arcs. The example shows that the "Omar Sharif" is used as two arguments of the event indicated by the argument role types $Victim$ and $Agent$. On the other hand, the $Conflict.Attack$ event has three arguments "who", "bomb" and "Israel". They are shown in blue arcs annotated with role types $Attacker$, $Instrument$ and $Place$. In this example, the "bomb" is an argument simultaneously shared by the $Life.Die$ event and the $Conflict.Attack$ event.

The example in Fig. 1 shows that there are two types of the argument multiplexing in event extraction: intro-multiplexing and extro-multiplexing. Intro-multiplexing refers to the phenomenon that an argument mention is used as different arguments in the same event. In extro-multiplexing, an argument mention is shared by different events at the same time. The main problem with previous joint models is that they usually assume an one-to-one relationship between each mention and argument in a sentence. In this setting, the argument multiplexing problem is ignored, which leads to three problems in event extraction. First, triggers and arguments in an event usually have a complex semantic structure. Ignoring the argument multiplexing can not encode the semantic structure between them, which distorts the decision plane of a classifier and worsens the discriminability of a neural network. Second, multiplexed arguments have the same semantic representation, but annotated with different argument types, which weakens the semantic expression of their representations. Third, in an end-to-end model, an argument can be annotated with only an type tag. The one-to-one labeling strategy is inevitable to lose multiplexed arguments, which leads to lower recall ratio.

In this paper, we propose a multigraph-based event extraction framework to address the argument multiplexing problem in event extraction. In this framework, semantic structures of events are represented as a multigrapha, where nodes denote tokens in a sentence. The edges between nodes represent semantic roles between tokens, e.g., entity types or argument roles. Because the multigraph allows multi-parallel edges between any two nodes, they can represent multiplexed arguments in event extraction. Furthermore, the multigraph possesses powerful expressiveness. It has the ability to represent events with any semantic structures, and distinguish overlapped events in a sentence. Based on this framework, we design a multigraph-based end-to-end (denoted as MGREE) deep architecture for event extraction. This model identifies all overlapped events and multiplexed trigger mentions and argument mentions in a sentence in a unified framework. It has the advantage to simultaneously avoid the cascading failure and argument multiplexing problems. After evaluated on four public datasets, MGREE achieves the state-of-the-art performance, outperforming related works about 4% in F1-score. The main contributions of this study are listed as follows.

- A multigraph-based event extraction framework is proposed in this paper. It has the ability to represent the semantic structure in overlapped events and avoid the argument multiplexing problem in event extraction.
- An end-to-end event extraction model is designed for event extraction. It achieved state-of-the-art performance, outperforming existing models on public benchmarks.

The remainder of this paper is organized as follows. Section 2 presents related studies on event extraction. Section 3 presents the multigraph representation and introduces the architecture of the MGREE model. Section 4 gives the performance of our model. Section 5 analyzes the effects of multigraph representations. Finally, the conclusions are presented in Section 6.

2. Related work

Extracting events from sentences is a widely studied topic in information extraction. Existing methods can be roughly divided into pipeline-based methods and joint-based methods [12]. Joint methods can be further divided into classification-based methods and generation-based methods. They are briefly introduced as follows.

2.1. Pipeline based methods

Early pipeline-based methods [8] for event extraction separate the task into event detection and argument extraction. Sequence labeling models have been widely used to identify triggers and event types in sentences [13–18]. After event triggers have been identified, the argument extraction task is also implemented as a sequence labeling task [19–21]. With the development of pre-trained language models [22], Yang et al. [23] introduced a BERT-based event extraction model that utilizes two independent models with multiple sets of binary classifiers to extract triggers and arguments. To incorporate prior knowledge, many studies [9–11] transformed event extraction into a multi-round question-answering task. Among them, Du et al. [11] is the first model to use predetermined question templates to extract triggers, followed by parameter extraction and filling with another question template. Pipeline-based methods have show great competitiveness for event extraction. They have the advantage to facilitate information transferring between the two tasks. However, pipeline-based event extraction methods have the drawback to propagate errors from event detection to argument extraction. Inaccurate or missed event triggers set the upper limit for the extraction task. Furthermore, the two subtasks are optimized independently, which cannot share semantic features between them and easily lead to the local optimization problem.

2.2. Joint based methods

To alleviate the influence of error propagation, many studies focused on joint learning methods for event extraction, which extract events in a unified framework. Early joint methods mainly relied on handcrafted features. Following the development of deep neural network, Nguyen et al. [24] proposed a recursive neural network that assumes entity information is known in advance and utilizes entity types to assist the prediction. Liu et al. [2] jointly extracted multiple event triggers and arguments in a single passing model. They introduced syntactic shortcut arcs to enhance information flow and adopted an attention-based graph convolutional network to model graph information. All these studies assume that entities in the sentence are pre-annotated and given in advance. They extract event arguments by classifying the relationships between triggers and arguments. However, in practical applications, they heavily depend on external toolkits to identify named entities. It can lead to the error propagation and performance degradation problems. Therefore, many joint extraction methods integrate entity recognition, trigger extraction, and argument extraction into a unified framework. For example, Nguyen et al. [25] extracted entities, triggers, and arguments from an end-to-end model. DyGIE++ [26] is a joint framework based on dynamic span graphs that can capture intra-sentence and cross-sentence context. OneIE [27] is a joint model based on global features that decodes the globally optimal graph through beam search and achieves the best results in event extraction. Joint extraction methods have the advantage of sharing model parameters across different tasks, but related works ignore the argument-multiplexing problem.

2.3. Generation based methods

In recent years, researchers have focused on generation-based methods that use sequence-to-structure generation networks to achieve unified event extraction. TANL [28] treats event extraction as a translation task in natural language processing. However, due to the gap between natural language and event structures, these models usually have poor performance. Lu et al. [29] proposed a controllable sequence-to-structure model that improves extraction performance by transforming a natural language into an event-structure-oriented language. Lu et al. [30] further proposed an UIE model, which extends the sequence-to-structure pattern to all information extraction tasks. Hsu et al. [31] first focused on using prompt templates to inject additional weak supervision information into the model, which improves low-resource event extraction performance. Liu et al. [32] proposed a dynamic event extraction method based on generative templates. It learns specific prefixes for each context by combining context information and specific type prefixes. Compared to classification-based joint extraction models, generative-based models require less annotated data. They have the ability to extract new event types by modifying the decoding strategy. However, despite the use of structured language to represent events, there is still a significant gap between natural language and event structures, which hinders the effective transfer of knowledge from pre-trained models. Moreover, due to fixed event templates in generation-based methods, they cannot correctly identify events with the argument-multiplexing problem.

3. Approach

In this section, before presenting our model to support event extraction, we first give a formalized discussion of the multigraph based event structure representation.

3.1. Event multigraph

A graph $\mathbf{G}$ is represented as a two-tuple $\langle \mathbf{V}, \mathbf{E} \rangle$, where $\mathbf{V} = \{v_1, v_2, ..., v_N\}$ is a node set and $\mathbf{E} = \{e_{ij} | 1 \leq i, j \leq N\}$ is an edge set. An element $e_{ij} \in \mathbf{E}$ indicates that there is an edge between nodes v_i and v_j in $\mathbf{V}$, represented as $e_{ij} = \langle v_i, v_j \rangle$. In a traditional directed graph, two vertices can only be connected by a directed edge, also known as "arc". This indicates that, in the edge set $\mathbf{E}$, $e_{ij} = e_{i'j'}$, if and only if $i = i'$ and $j = j'$. In this condition, a graph can also be denoted by an $N \times N$ matrix $\mathbf{M}$. An element $\mathbf{M}_{ij} = 1$ indicates that there is an edge between v_i and v_j; otherwise, $\mathbf{M}_{ij} = 0$.

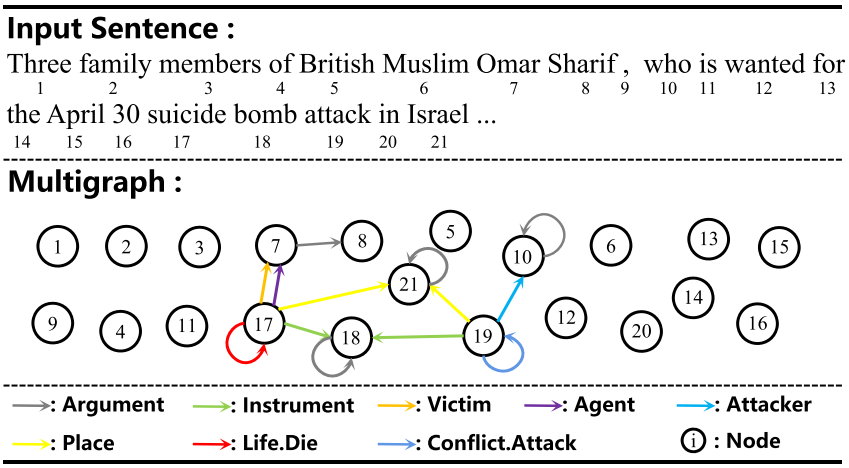


Fig. 2. An Example of Event Multigraph.

A multigraph $\mathbb{G} = \langle \mathbb{V}, \mathbb{E} \rangle$ is a generalization of a traditional graph $\mathbf{G}$. The difference is that in a multigraph, the edge set $\mathbb{E}$ is defined as a multiset, which allows multiple instances for each element. In multiset, the edge set can be further defined as a two-tuple $\mathbf{E} = (A, m)$. A is the underlying set of $\mathbf{E}$, formed from its distinct elements in $\mathbf{E}$. m is a function from A to positive integers. It is used to indicate the number of occurrences of repeated elements. For example, $m(e_{ij}) = k$ means that there are k edges between nodes v_i and v_j. To simplify our problem, we define the order of a multigraph as the maximum multiplicity of all the elements in the edge set $\mathbb{E}$. The order of $\mathbb{G}$ is denoted as K. In event extraction, it is used to indicate the largest number of labels per argument mention.

Let $\mathbf{S} = [v_1, v_2, \cdots, v_N]$ be a sentence composed of N tokens referred from v_1 to v_N. A trigger or argument of an event is defined as a mention (a subsequence) in $\mathbf{S}$, represented as $[v_i, v_{i+1}, \cdots, v_j]$ or $v_{i:j}$. Let $\mathbf{T} = \{t_1, \cdots, t_{K_t}\}$ be the trigger type set. If a mention $v_{i:j}$ is a trigger, an edge is connected from v_i to v_j. It is assigned a label in $\mathbf{T}$ to indicate the event type of the trigger. Let $\mathbf{R} = \{r_1, \cdots, r_{K_r}\}$ be the argument type set. Labels belonging to $\mathbf{R}$ are used to determine whether a mention is an argument or not. It means that $\mathbf{R} = \{true, false\}$. In event extraction, every argument (e.g., $v_{s:t}$) should be assigned with a label (e.g., "*Agent*" and "*Attacker*") to indicate its argument role in an event. The argument role type set is denoted as $\mathbf{L} = \{l_1, \cdots, l_{K_l}\}$. Then, if an argument mention $v_{s:t}$ is an argument role of an event trigger $v_{i:j}$, there is an edge from v_i to v_s. It is assigned with a label in $\mathbf{L}$ to indicate its argument role. Note that, an edge with label in $\mathbf{L}$ connect the head token of a trigger with the head token of an argument. On the other hand, an edge with label in $\mathbf{R}$ connect the head token and tail token of an argument. It is used to indicate the range of an argument in a sentence.

In summary, in event extraction, the node set $\mathbb{V}$ of $\mathbb{G}$ is composed of all tokens in $\mathbf{S}$. All edges from v_i to v_j can be assigned labels in $\mathbf{Y} = \mathbf{T} \cup \mathbf{L} \cup \mathbf{R}$. The cardinality of $\mathbf{Y}$, denoted $|\mathbf{Y}|$, is the same as the order of $\mathbb{G}$, denoted as $K = K_t + K_l + K_r$. To distinguish different edges from v_i to v_j, all edges between them are indexed from 1 to K. Therefore, a multigraph can be represented by an $N \times N \times K$ tensor $\mathbb{M}$. If $\mathbb{M}_{ijk} = 1$, then the edge of the k-th type of $\mathbf{Y}$ is connected from v_i to v_j.

Based on the above formalization, in the following, a multigraph example is given to show the semantic structure of events in a sentence. It is illustrated in Fig. 2.

A sentence is shown at the top of Fig. 2, where each token has an index below it, which also used referring to nodes of the multigraph in the middle of Fig. 2. In this multigraph, each node represents a token. Each directed edge from v_i to v_j represents a mention $v_{i:j}$. The edge types is represented by different colors shown at the bottom of Fig. 2. The types of edges indicate that a mention is a trigger, an argument, or a semantic role in an event. These edges are divided into three categories: a trigger type set $\mathbf{T}$, an argument type set $\mathbf{R}$ and a argument role type set $\mathbf{L}$. The triggers $Life.Die$ and $Conflict.Attack$ belong to the $\mathbf{T}$ set. *Argument* belongs to $\mathbf{R}$ set. The rest of them belong to the $\mathbf{L}$ set. In the multigraph, the red loop $\langle v_{17}, v_{17} \rangle$ indicates that the mention "suicide" triggered an $Life.Die$ event. Likewise, the four gray edges $\langle v_7, v_8 \rangle$, $\langle v_{10}, v_{10} \rangle$, $\langle v_{18}, v_{18} \rangle$ and $\langle v_{21}, v_{21} \rangle$ indicate the range of three argument mentions in a sentence ($v_{7:8}$, $v_{10:10}$, $v_{18:18}$, $v_{21:21}$). The directed edges $\langle v_{17}, v_7 \rangle$, $\langle v_{17}, v_{21} \rangle$, and $\langle v_{17}, v_{18} \rangle$ indicate their argument role types in the $Life.Die$ event. It can be seen that there are two parallel edges between v_{17} and v_7. This indicates that "Omar Sharif" has two argument role types ($Victim$ and $Agent$) in the same event triggered by the mention "suicide".

3.2. Model architecture

Based on the multigraph event representation, we propose a multigraph-based end-to-end (MGREE) model for event extraction. The architecture of MGREE is illustrated in Fig. 3. It is composed of three modules: a node encoding module, a multigraph building module, and an event decoding module.

The node encoding module first transforms tokens of a sentence into abstract token representations. These token representations serve as nodes in the multigraph. Then, the relationship between tokens are initialized into an $N \times N \times K$ matrix $\mathbb{M}$, which denotes to a formalized multigraph representation. Finally, events are extracted from the event decoding module, which jointly predicts the triggers, arguments, and roles of events. Each module is discussed as follows.

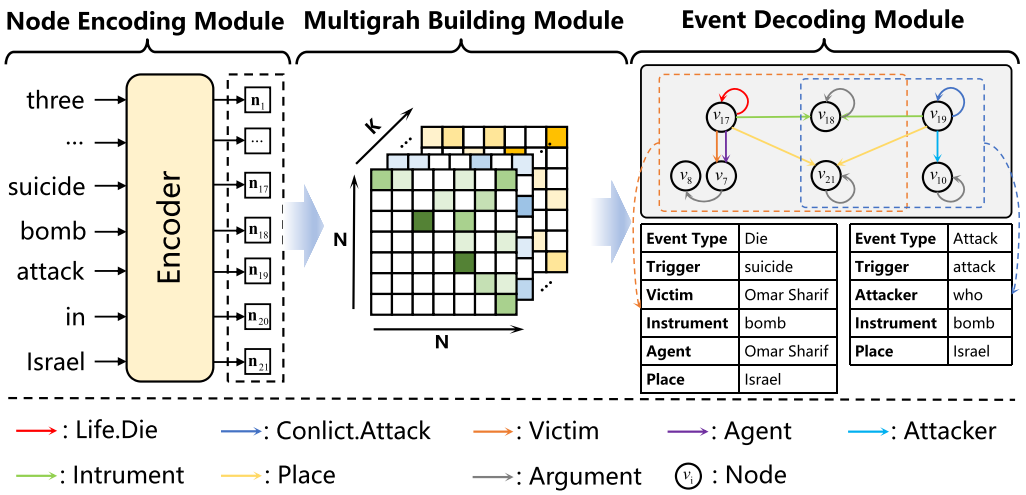


Fig. 3. Mode Architecture of MGREE. Firstly, the node encoding is used to convert tokens into node representations. Secondly, a 3D matrix, denoted to a multigraph, is constructed to represent the semantic structures of all events in a sentence. Finally, the event decoding module outputs events composed of triggers and relevant arguments.

3.2.1. *Node encoding module*

Each node of the multigraph is an abstract token representation of a sentence. This module is used to learn node representations of the multigraph from a raw input sentence. Because these node representations are directly adopted to support the multigraph building and event decoding, it is important to make use of semantic information of tokens and encode contextual features of a sentence.

In natural language processing, tokens are usually indexed by integers denoting to entries of a dictionary [33,27]. The integer token representation can not be directly processed by a deep model, because neural networks treat the input as the intensity of signals. Therefore, before feeding tokens into a deep neural network, they should be transformed into signalized representation. There are three types of transformations: one-hot representation, word embedding and pre-trained language models (PLMs).

One-hot representations are orthogonal vectors with a large dimensional size. Every vector only contains a non-zero element, which cannot encode semantic information of tokens. In word embedding, every token is transformed into a dense vector by a lookup table initialized from external resources, which has the advantage to encode semantic information of tokens. Because every token owns a unique vector learned from a whole corpus, it is not effective to encode contextual features of a sentence. PLMs are Transformer based models trained with self-supervised algorithms, which predict masked tokens depending on contextual features of a sentence. Therefore, token representations of PLMs have the advantage to encode the contextual features in a sentence. In the node encoding module, both word embedding and PLMs are used to generate node representations.

Let $\mathbf{S} = [v_1, v_2, \cdots, v_N]$ be the input sentence consisting of N tokens. It is first fed into a PLM encoder for generating token representations. In order to make full use of potential knowledge of PLMs, following Lin et al. [27], an average pooling operation is adopted to generate token representations from the last three layers of PLM, which has the advantage to encode multi-granularity semantic features of tokens to enrich the token representations. The output of PLM is represented as: $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_N]$.

The word embedding is generated by a pre-trained skip-gram word embedding.[1] Let $\mathbf{H}' = [\mathbf{h}'_1, \mathbf{h}'_2, \cdots, \mathbf{h}'_N]$ be the output of skip-gram word embedding. Then, a token representation is generated by concatenating $\mathbf{H}$ and $\mathbf{H}'$ to enhance the semantic information of the tokens. The final token representation is:

$$\hat{\mathbf{H}} = [\hat{\mathbf{h}}_1, \hat{\mathbf{h}}_2, \cdots, \hat{\mathbf{h}}_N] \tag{1}$$

where $\hat{\mathbf{h}}_i = \mathbf{h}_i \oplus \mathbf{h}'_i$ is an abstract representation of the token v_i.

The skip-gram word embedding is initialized on news data. As discussed before, every token has a unique vector in the lookup table. It cannot encode semantic dependencies between tokens in a sentence. Furthermore, we generate multi-granularity semantic representations from the last three layers of PLM. To learning the interaction between token representations in $\hat{\mathbf{H}}$, a BiLSTM layer is set to learn the semantic dependencies between token representations. This process is expressed as follows:

$$\overrightarrow{\mathbf{n}}_i = \overrightarrow{\mathrm{LSTM}}(\hat{\mathbf{h}}_i, \overrightarrow{\mathbf{n}}_{i-1}) \tag{2}$$

$$\overleftarrow{\mathbf{n}}_i = \overleftarrow{\mathrm{LSTM}}(\hat{\mathbf{h}}_i, \overleftarrow{\mathbf{n}}_{i+1}) \tag{3}$$

$$\mathbf{n}_i = [\overrightarrow{\mathbf{n}}_i; \overleftarrow{\mathbf{n}}_i] \tag{4}$$

[1] The word vector file we are utilizing originates from the following repository: https://github.com/yubochen/NBTNGMA4ED.

where [;] denotes to the concatenation.

Finally, the node encoding module outputs a sequence of vectors $\mathbf{N} = \{\mathbf{n}_1, \mathbf{n}_2, \cdots, \mathbf{n}_N\} \in \mathbb{R}^{N \times D'}$ to represent $\mathbf{S}$, where D' denotes the dimensions of the BiLSTM output. $\mathbf{N}$ are composed of node representations, encoding semantic information of token and contextual features of a sentence.

Note that, in this paper, v_i is a token of a sentence. $\mathbf{n}_i$ is an abstract representation of v_i. They can also be used to refer to a node or node representation in the multigraph representation. In the training process, all parameters of the employed PLM and skip-gram are updated under the task relevant objective. For tokens that are unseen during the testing process, we replace them with a specific 'unknown' token.

3.2.2. Multigraph building module

As introduced in Section 3.1, in an event multigraph, three types of label sets are defined to support the event extraction: a trigger type set $\mathbf{T}$, an argument type set $\mathbf{R}$ and an argument role type set $\mathbf{L}$. Unlike token labeling methods or span classification methods which assign a label on a token or a span to indicate its semantic role in an event, in multigraph, all labels in $Y \in \mathbf{Y}$ ($\mathbf{Y} = \mathbf{T} \cup \mathbf{L} \cup \mathbf{R}$) are assigned to edges between two nodes. For example, if a mention $v_{i:j}$ is a trigger, an edge in $\mathbf{T}$ is connected from v_i to v_j. Labels in $\mathbf{R}$ indicates whether the edge from v_i to v_j is an argument or not. If an argument mention $v_{s:t}$ is an argument role of an event trigger $v_{i:j}$, there is an edge from v_i to v_s. It can be assigned with a label in $\mathbf{L}$.

If the length of sentence is N, there are N nodes in the multigraph representation. Because a label $y_k \in \mathbf{Y}$ can be assigned to any edge, it means that we should construct $N \times N$ node pairs (or edges) for each type of label in $\mathbf{Y}$. The order of the multigraph in our experiment is K. It is the same as the cardinality of $\mathbf{Y}$ ($K = |\mathbf{Y}|$). Therefore, to address the argument multiplexing problem in event extraction, the multigraph representation should contain $N \times N \times K$ edges (node pairs). They can be organized into a $N \times N \times K$ three-dimensional matrix, denoted as $\mathbb{E}$. The element of $\mathbb{E}$, denoted as $\mathbb{E}_{ijk}$, is the kth edge between nodes v_i and v_j. Because a node can act as a head node or a tail node, it has two abstract representation in a multigraph. Our method to generate node representations is discussed as follows.

Given the node representations $\mathbf{N} = \{\mathbf{n}_1, \mathbf{n}_2, \cdots, \mathbf{n}_N\}$, we used two separate MLP layers to perform the nonlinear transformation of the node representations. The outputs are two specific node representations, named as head node $\mathbf{n}_i^h$ and tail node $\mathbf{n}_i^t$. They are computed as:

$$\mathbf{n}_i^h = \sigma(\mathbf{W}_h \mathbf{n}_i + \mathbf{b}_h) \tag{5}$$

$$\mathbf{n}_i^t = \sigma(\mathbf{W}_t \mathbf{n}_i + \mathbf{b}_t) \tag{6}$$

where $\mathbf{n}_i^h \in \mathbb{R}^{d_b}$ and $\mathbf{n}_i^t \in \mathbb{R}^{d_b}$ can be seen as node representations acting as a head node or a tail node in different edges. Therefore, an edge is represented as a two-tuple $\langle n_i^h, n_j^t \rangle$. Because two nodes can be connected by several edges, $\mathbb{E}_{ijk} = \langle n_i^h, n_j^t, k \rangle$ denotes to the kth edge pointing from v_j to v_i. There are at most K edges between two nodes. They can also be referred as $\mathbb{E}_{ij:} = \langle n_i^h, n_j^t, : \rangle$. σ denotes the GELU activation function [34], $\mathbf{W}_h$ and $\mathbf{W}_t$ are trainable parameters. $\mathbf{b}_h$ and $\mathbf{b}_t$ are biases.

Then, given two node representations n_i^h, n_j^t, a biaffine attention mechanism is adopted to calculate the confidence score $\mathbb{M}_{ijk}$ relevant to the edge type of y_k:

$$\mathbb{M}_{ijk} = (\mathbf{n}_i^h)^T \mathbf{U}_k^{(1)} \mathbf{n}_j^t + (\mathbf{n}_i^h \oplus \mathbf{n}_j^t)^T \mathbf{U}_k^{(2)} + \mathbf{b}_\mathbf{k} \tag{7}$$

where $\mathbf{U}_k^{(1)} \in \mathbb{R}^{d_b \times d_b}$ and $\mathbf{U}_k^{(2)} \in \mathbb{R}^{2d_b}$ are trainable weight parameters and $\mathbf{b}_\mathbf{k}$ is the bias. $\mathbb{M}_{ijk}$ denotes to the confidence score of edge $\langle v_i, v_j \rangle$ relevant the label $y_k \in \mathbf{Y}$. Then, $\mathbb{M}$ can be seen as the confidence score matrix, which encodes the interconnection relationships between nodes in a multigraph.

In the training process, the circle loss proposed by Sun et al. [35] is adopted to minimize the disparity between the matrix $\mathbb{M}$ and the ground truth matrix initialized with annotated information. The total loss is computed as follows:

$$Loss = \frac{1}{K} \sum_k \left[log\left(e^{\epsilon} + \sum_{i,j \in \mathbf{E}_{neg}} e^{\mathbb{M}_{ijk}}\right) + log\left(e^{-\epsilon} + \sum_{i,j \in \mathbf{E}_{pos}} e^{-\mathbb{M}_{ijk}}\right) \right] \tag{8}$$

where $\mathbf{E}_{pos}$ and $\mathbf{E}_{neg}$ are the positive edge set and negative edge set in the multigraph. Positive or negative edge indicate whether there is an edge between two nodes. Positive and negative edges take 1 or 0 in the ground truth matrix. $\mathbb{M}_{ijk}$ is the confidence score between two nodes. In the Circle loss, the confidence scores take values from $-\infty$ to $+\infty$. $\epsilon > 0$ means that there is an edge between two nodes, and vice versa.

The interconnection confidence scores of $\mathbb{M}$ can be normalized to show the distribution of all edges in label set $\mathbf{Y}$. The process is represented as:

$$\mathcal{P} = Sigmoid(\mathbb{M}) \tag{9}$$

The $Sigmoid$ is a sigmoid function. $\mathcal{P} \in \mathbb{R}^{N \times N \times K}$ is a matrix. The element $\mathcal{P}_{ijk} \in \mathcal{P}$ denotes to the probability of edge $\langle n_i^h, n_j^t \rangle$ with label $y_k \in \mathbf{Y}$.

3.2.3. Event decoding module

The matrix $\mathcal{P}$ contains all interconnection information of a multigraph. Therefore, in the inference stage, the objective is to decode possible events from $\mathcal{P}$. The main problem of decoding events is that, for each possible event, we should find a dense

semantic structure which includes all arguments of the event. For example, if we find an argument mention with label in **R** but without relevant to any trigger, it is filtered as noise. In this module, an event decoding strategy is proposed to induce events from $\mathcal{P}$. The strategy is shown in Algorithm 1.

The input of Algorithm 1 are three matrices $\mathcal{P}_T$, $\mathcal{P}_R$, $\mathcal{P}_L$ disassembled from $\mathcal{P}$ according to label sets **T**, **R**, and **L**. In the multigraph representation, there are K edges between any two nodes, which leads to a large number of edges being predicted in the output. Therefore, we set three predefined thresholds $\theta^t, \theta^r, \theta^l$ to filter those that are unlikely to be edges. In real applications, they are all set to be 0.5 for balancing the precision and the recall. The output of Algorithm 1 are two sets contain triggers and relevant arguments, respectively.

Algorithm 1: Event Decoding Algorithm.

Input: matrices $\mathcal{P}_T, \mathcal{P}_R, \mathcal{P}_L$ and thresholds $\theta^t, \theta^r, \theta^l$
Output: trigger set $\mathcal{T}$, argument set $\mathcal{A}$

1 Initialize $\mathcal{T} \leftarrow \emptyset$, $\mathcal{R} \leftarrow \emptyset$, $\mathcal{L} \leftarrow \emptyset$, $\mathcal{A} \leftarrow \emptyset$;
2 **for** $i \leftarrow 1$ *to* N **do**
3 **for** $j \leftarrow 1$ *to* N **do**
4 **if** $i \leq j$ **then**
5 $edges \leftarrow filter(\mathcal{P}_T[i, j, :], \theta^t)$ // an edge is $\langle v_i, v_j, y_k \rangle$;
6 $\mathcal{T}.add(edges)$ // edges with labels in **T**;
7 $edges \leftarrow filter(\mathcal{P}_R[i, j, :], \theta^r)$;
8 $\mathcal{R}.add(edges)$ // edges with labels in **R**;
9 **end**
10 $edges \leftarrow filter(\mathcal{P}_L[i, j, :], \theta^l)$;
11 $\mathcal{L}.add(edges)$ // edges with labels in **L**;
12 **end**
13 **end**
14 **for** $e \in \mathcal{L}$ **do**
15 **if** $\exists t \in \mathcal{T} \wedge e[0] = t[0]$ ***and*** $\exists r \in \mathcal{R} \wedge e[1] = r[0]$ **then**
16 $\mathcal{A}.add((t[0], t[1], t[2], r[0], r[1], t[2]))$;
17 **end**
18 **end**

In Algorithm 1, we Firstly, we formalize the event structure as a trigger set E and an argument set R. According to the definition in Section 3.1, $\mathcal{P} \in \mathbb{R}^{N \times N \times K}$ can be decomposed into three probability tensors: $\mathcal{P}_T \in \mathbb{R}^{N \times N \times K_t}$, $\mathcal{P}_R \in \mathbb{R}^{N \times N \times K_r}$, and $\mathcal{P}_L \in \mathbb{R}^{N \times N \times K_l}$, corresponding to the sets **T**, **R**, and **L**, respectively.

In Line 1, four sets $\mathcal{T}$, $\mathcal{R}$, $\mathcal{L}$ and $\mathcal{A}$ are initialized for holding trigger mentions, argument candidates, argument roles and event arguments, respectively

From Line 2 to Line 13, two nested loops are used to collect all edge with confidence scores larger than predefined thresholds θ^t, θ^r, and θ^l. The outputs are three edge sets $\mathcal{T}$, $\mathcal{R}$ and $\mathcal{L}$. Every edge denotes to a three tuple $\langle v_i, v_j, y_k \rangle$. For example, each edge in $\mathcal{T}$ is a trigger composed of tokens from v_i to v_j with type label y_k. An edge in $\mathcal{L}$ denotes to an argument role relation connecting the head token of a trigger v_i to the head token of an argument v_j with label in **L**.

From Line 14 to Line 18, all triggers of $\mathcal{T}$ are enumerated for constructing structuralized events. For every trigger (or event), we traveled all argument role relations and argument mentions from the argument role set $\mathcal{L}$ and the argument candidate set $\mathcal{R}$. They are combined to a structuralized form and added into the event argument set $\mathcal{A}$. An element of $\mathcal{A}$ is composed of trigger head, trigger tail, event type, argument head, argument tail, and role type.

After event decoding, we get the trigger set $\mathcal{T}$ and the event argument set $\mathcal{A}$. The decoded arguments may not match to predefined event schema. For example, a $Marry$ event may contain only two types of arguments, $Person$ and $Place$, and must not contain an $Attacker$. Under this condition, following related works [36,37], we also use the event pattern knowledge to constrain the decoding for filtering out unmatched arguments.

4. Experiments

In this section, the employed evaluation datasets and experimental settings are first introduced in Section 4.1. Then, our model is compared with related works in Section 4.2. Finally, in Section 4.3, ablation studies are conducted to evaluate the effectiveness of multigraph-based event extraction.

4.1. Dataset and settings

Our method is evaluated on the Automatic Content Extraction (ACE) 2005[2] event extraction dataset [1], collected from newswires, broadcasts, and weblogs. This dataset comprises approximately 1,800 files in English, Arabic, and Chinese, annotated with 33 event types ($K_t = 33$) and 22 argument roles ($K_l = 22$). Because we only consider whether a mention is an argument or not, we set $K_r = 1$.

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

Table 1
Dataset Statistics. **Sents**, **Events** and **Roles** denote the numbers of sentences, entity mentions, relations, and events in the datasets, respectively.

Dataset	Split	Sents	Events	Roles	Dataset	Split	Sents	Events	Roles
ACE05-EN	Train	17172	4202	4859	ACE05-EN+	Train	19204	4419	6607
	Dev	923	450	605		Dev	901	468	759
	Test	832	403	576		Test	676	424	689
ACE05-CN	Train	6,841	2,926	5463	ERE-EN	Train	14376	6208	8924
	Dev	526	217	403		Dev	1209	525	730
	Test	547	190	332		Test	1163	551	822

For comparing with related works, we follow the data split and data preprocessing in previous works [26,27,29,38]. We adopt the same strategy to generate two evaluation datasets from the ACE English corpus: ACE05-EN and ACE05-EN+. The difference between them is that the ACE05-EN+ dataset further considers pronoun roles and multi-token event triggers. To show the extendibility of the multigraph on different languages, we followed Lin et al. [27] to construct a Chinese dataset derived from ACE2005, named as ACE05-CN. In addition, following related works [27,29], we combine data from three English datasets (namely LDC2015E29, LDC2015E68, and LDC2015E78) to create the ERE-EN dataset, which was developed under the Deep Exploration and Filtering of Test (DEFT) program [39]. The statistical information for the datasets is listed in Table 1.

To keep in line with previous works [27,9,29], we use the Precision, Recall and F1 scores to measure the performance of trigger classification (**Trg-C**) and argument classification (**Arg-C**). A predicted trigger or argument is correct only if its location offset and type exactly match annotated labels.

Our experiments are implemented on the transformers library [40]. For English datasets, we use the pre-trained DeBERTa-large model[3] and skip-gram word embeddings as node encoders. As for the Chinese dataset, the bert-base-chinese model[4] is adopted. To obtain skip-gram word embeddings, we use the "unknown" marker for out-of-vocabulary words. The dimension size of skip-gram word vector is 100. A bidirectional LSTM is adopted to aggregate the node representations. The dimension of LSTM hidden layer is set to 512. The hidden-layer dimension of the MLP layer is 768. Our model is trained with an Adam weight decay optimizer [41]. All parameters of the model are involved in optimization and updating, including the PTM and the skip-gram word embedding module. The initial learning rate is set to 5e-5 for the pre-trained parameters and 1e-3 for other parameters. The warmup ratio, training epoch, batch size are set to 0.1, 10 and 32, respectively. The code to implement our model is available online at: https://github.com/huanghuidmml/mgree.

4.2. Main results

Our model is compared with previous works, which can be divided into three groups: pipeline-based, joint-based, and generation-based models. In addition, a large language model (LLM) is fine-tuned for comparison. They are discussed as follows.

Pipeline-based methods: EEQA [11] and MQAEE [9] are both QA-based event extraction models. They used a multiple cascaded machine-reading comprehension model to extract triggers and arguments of events. Wang et al. [42] used event types and argument roles as queries to drive event triggers and argument extraction. It was combined with an attention mechanism to enhance the association between the queries and tokens. It achieved the state-of-the-art performance.

Joint-based methods: Joint3EE [25] extracts entities, triggers, and arguments in a unified model. DyGIE++ [26] is a dynamic span graph-based joint information extraction framework that captures both within-sentence and cross-sentence context. OneIE [27] is a joint framework based on global features, which decodes the global optimal graph by beam search. UniEX [43] is a unified extraction framework that decomposes information extraction into span detection, classification, and association.

Generation-based methods: Text2Event [29] is a joint event-extraction method based on a generative language model. UIE [30] adopts an unified structure generation to learn universal information extraction capabilities. DEGREE [31] uses a conditional generative model to support end-to-end event extraction.

LLM-based methods: We transformed the event extraction task into an instructional format and fine-tuned the Llama2-7b model for comparison [44], which contains 7 billion parameters. In this experiment, we employed the LoRA method proposed by Hu et al. [45] to train model parameters.

The results are shown in Table 2, where "Tri-C" and "Arg-C" represent the performance on trigger classification and argument classification, respectively. The performance is given in F1 score. "Ent" indicates whether or not the manually annotated entity information is used. "*" denotes experimental results we replicated by using officially published codes.

Compared to pipeline-based models, we achieve a significantly improvement on the event extraction task. Our MGREE model uses a multigraph to represent event semantic structures, which uniformly extract triggers and arguments. It has the ability to avoid the performance degradation problem caused by error propagation. Furthermore, MGREE is a joint model, it is effective to learn the interaction between the two subtasks. Compared with joint event extraction models, MGREE has the advantage to avoid the argument multiplexing problem. Generation-based models encode sentences into dense representations and directly generate a structurized

[3] https://huggingface.co/microsoft/deberta-v3-large.
[4] https://huggingface.co/bert-base-chinese.

Table 2
Comparing with Related Works.

Models	Type	Ent	ACE05-EN		ACE05-EN+		ACE05-CN		ERE-EN	
			Tri-C	Arg-C	Tri-C	Arg-C	Tri-C	Arg-C	Tri-C	Arg-C
EEQA [11]	Pipe	-	72.4	53.3	-	-	-	-	-	-
MQAEE [9]	Pipe	-	71.7	53.4	-	-	-	-	-	-
Wang et al. [42]	Pipe	✓	-	-	73.6	55.1	-	-	-	-
Joint3EE [25]	Joint	✓	69.8	52.1	-	-	-	-	-	-
DYGIE++ * [26]	Joint	✓	69.7	48.8	-	-	72.3	59.3	-	-
OneIE * [27]	Joint	✓	74.7	56.8	72.8	54.8	73.3	60.5	57.0	46.5
UniEX [43]	Joint	-	-	-	74.1	53.9	-	-	-	-
Text2Event [29]	Gen	-	71.9	53.8	71.8	54.4	-	-	59.4	48.3
UIE [30]	Gen	-	-	-	73.4	54.8	-	-	-	-
DEGREE [31]	Gen	-	73.3	55.8	70.9	56.3	-	-	57.1	49.6
Llama2-7b [44]	LLM	-	72.7	52.6	69.9	55.5	51.1	38.4	61.1	50.5
Ours	Joint	-	**75.9**	**57.4**	**76.1**	**58.8**	73.1	**62.1**	**62.8**	**51.7**

output, which may lead to potential information loss and result in poor performance. Compared with the generation-based models, our model also show significant improvement.

Compared with traditional models, MGREE achieves significant improvement over previous state-of-the-art models on both the ACE05-EN and ACE05-EN+ datasets. Specifically, compared to the OneIE [27] which had achieved the best performance on the ACE05-EN dataset, MGREE improves the performance of "Tri-C" and "Arg-C" about 1.2% and 0.6% in F1-score. Compared to the best model on the ACE05-EN+ dataset (DEGREE [31]), MGREE shows an increase of 5.2% in "Tri-C" and 2.5% increase in "Arg-C". Compared to the best model on the ACE05-CN dataset (OneIE [27]), our model achieves a comparable performance in "Tri-C", while showcasing a 1.6% improvement in "Arg-C". MGREE also achieves a 5.7% improvement in "Tri-C" and a 2.1% improvement in "Arg-C", outperforming the best model on the ERE-EN dataset (DEGREE [32]). The results demonstrate that our model is effective for joint event extraction.

Compared to the Llama2-7b model which contains 7 billion parameters, MGREE only has 0.47 billion parameters. Despite the Llama2-7b model contains a larger number of parameters, our model still achieves better performance than the Llama2-7b model. We think that there may be four reasons leading to the result. First, the LLM is a generative model, which is optimized with different learning objective. Second, a large number of parameters and pre-training datasets make the LLM easily overfitting to the training data. Third, a sentence usually contains overlapped events which share the same contextual features. They may lead to complex sentential semantic structures. It is more challenging to be disassembled in a generative model. Fourth, event extraction should deal with numerous labels [46]. It may weaken the performance in the LLM model.

Another interesting phenomenon showing in Table 2 is that, our model also outperforms models that use manually annotation entity information, e.g., Wang et al. [42], Joint3EE [25], DyGIE++ [26] and OneIE [27]. The result indicates that our model has stabler performance. It has the ability to represent events with complex semantic structures, and distinguish overlapped events in a sentence. It is effective to avoid the cascading failure and argument multiplexing problems, which are helpful to advance the discriminability of the neural network for event extraction.

4.3. Ablation study

In this section, we give an ablation study to analyze the influence of each component of the MGREE on the performance. Four models are conducted for comparison. These models are introduced as follows:

- "MGREE" is our complete model listed for comparison. In this model, the last three layers of the DeBERTa are averaged to generate token representations.
- "w/o avg" removes the average pooling layer. It directly uses the output of the last DeBERTa layer to generate token representations.
- "w/o skip-gram" removes the Skip-Gram module. It only uses the PLM to generate node representations.
- "w/o search" denotes to the model that the three thresholds in section 3.2.3 are fixed to 0.5. The threshold search operation is removed.
- "w/o lstm" removes the LSTM layer. It directly uses the output of the PLM and concatenates it with skip-gram word vectors as node representations.
- "w/o lstm & skip-gram" simultaneously removes the LSTM layer and skip-gram embeddings layer. It solely utilizes the PLM output as node representations.

In this experiment, the ACE05-EN+ and ERE-EN datasets are used to evaluate the influence of different models. The result is shown in Table 3.

Based on the results in Table 3, the following three conclusions can be drawn:

Table 3
Results of ablation studies on ACE05-EN+ and ERE-EN.

Models	ACE05-EN+						ERE-EN					
	Tri-C			Arg-C			Tri-C			Arg-C		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
MGREE	73.9	78.3	**76.1**	62.4	55.6	**58.8**	61.1	64.6	**62.8**	54.8	48.9	51.7
w/o avg	75.6	75.9	75.8	58.0	56.1	57.0	61.9	63	62.4	53.4	50.3	**51.8**
w/o skip-gram	70.59	76.4	73.4	56.6	58.2	57.4	62.1	61.5	61.8	53.1	49.3	51.1
w/o search	75.2	76.4	75.8	63.8	53.3	58.0	62.8	61.1	61.7	52.5	50.5	51.2
w/o lstm	68.7	78.3	73.2	58.6	51.3	55.4	61.6	61.7	61.6	54.4	46.4	50.1
w/o lstm & skip-gram	70.7	77.1	73.6	61.7	53.3	57.2	61.2	63.1	62.1	53.0	50.2	51.4

1) Pre-trained language models contain rich semantic information automatically learned with self-supervised methods. The last three layers of PLMs encode semantic features at different granularities. Fusing them is effective to enhance the semantic information of node features. Therefore, removing the average pooling operation results in degraded performance of "Tri-C" and "Arg-C" on both datasets.

2) As shown in "w/o skip-gram", obtaining word embeddings from different pre-trained models can further improve the performance of event extraction. The reason is that local contextual features are effective to support the trigger identification task. Comparing with the output of PLMs which is heavily influenced by semantic dependencies of a sentence, skip-gram word vectors encode informative local features of each word because they are learned from neighboring words. Therefore, skip-gram word vectors are also helpful to improve the performance, especially for trigger extraction. Removing the skip-gram module has greater impact on the "Tri-C" performance.

3) Comparing the MGREE with the "w/o search" model, the threshold search strategy is also helpful to improve the event extraction task. Because the argument extraction task requires not only predicting the argument head-tail relationship but also the affiliation relationship between arguments and triggers. The threshold search filters incorrect edges, which makes the output more compatible with event structures, thereby improving the performance of event extraction.

4) Comparing the performance between "w/o lstm" and "w/o lstm & skip-gram", the former is more influential on the performance. This result is interesting, because it indicates that, without the LSTM layer, directly adding skip-gram word embeddings may worsen the performance. Implementing the LSTM layer on the output of skip-gram can construct the semantic dependencies between word embeddings. Under this condition, in the training process, word embeddings can be tuned to learn task relevant word representations.

4.4. Results on argument multiplexing samples

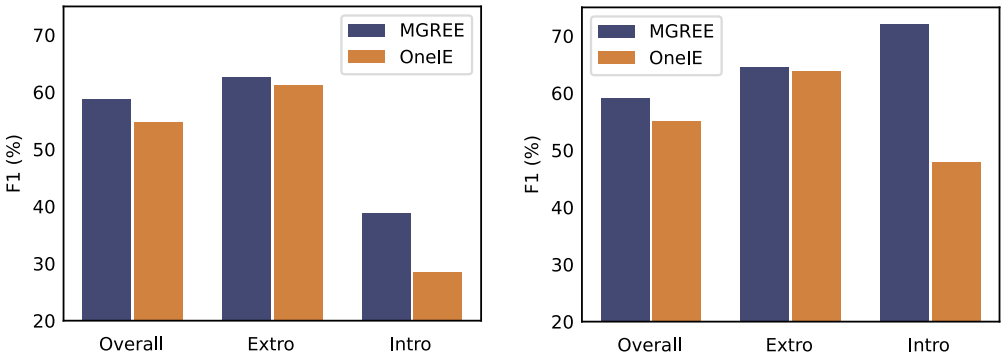
In event extraction, an argument mention can be used as different arguments in the same event or shared by different events at the same time. It leads to the intro-multiplexing and extro-multiplexing problems. As discussed before, in the intro-multiplexing, an argument mention is used as different arguments in an event. In the extro-multiplexing, an argument mention can be shared by different events. The multigraph based event representation allows parallel edges between two nodes. It is effective to resolve the multiplexing problem in event semantic structures.

To investigate the ability of MGREE to extract events with the argument multiplexing problem, in this section, we conducted experiments to compare our model with the previous SOTA model OneIE [27] to show the ability to resolve the multiplexing problem. The OneIE [27] model is a joint model based on global features that decodes the globally optimal graph through beam search and achieves the best results in event extraction.

Traditional methods to partition the training/ developing/ testing datasets may lead to the problem that no argument multiplexing samples in the testing dataset. In this condition, it is not effective to evaluate the ability to extract argument multiplexing samples. Therefore, in this experiment, we manually collect all argument multiplexing samples and randomly partition them into the ratio 6:2:2 for training, developing and testing, containing 11,904, 4,706, and 4,171 sentences respectively. The evaluation dataset is referred to as the ACE05-Multiplexing dataset. Then, we conduct the OneIE [27] model and our MGREE on the ACE05-EN+ dataset and the ACE05-Multiplexing dataset, respectively. The result is shown in Fig. 4.

Fig. 4(a) compares the performance of MGREE and OneIE on the traditional ACE05-EN+ dataset. They have similar performance on the extro-multiplexing samples. However, our model has higher performance on the intro-multiplexing samples. In Fig. 4(b), when evaluated on the ACE05-Multiplexing dataset, compared with the OneIE model, our model has more significant improvement, outperforming the F1 value 20+% on the intro-multiplexing samples.

The OneIE model uses a pairwise classifier for event extraction. It assumes a one-to-one relationship between triggers and arguments in an event. Because every event is extracted independently in the OneIE model, extro-multiplexing samples can also be recognized by this model. However, the OneIE cannot resolve the intro-multiplexing problem, where a mention can be used as a different argument for the same event. In contrast, our MGREE model allows parallel edges between triggers and arguments. It has the ability to represent events with any semantic structures, and distinguish overlapped events in a sentence. Therefore, our model is more effective than the OneIE model in supporting argument multiplexing for event extraction.



(a) The ACE05-EN+ dataset. (b) The ACE05-Multiplexing dataset

Fig. 4. Evaluation on Argument Multiplexing Samples. **Overall**, **Intro** and **Extro** denote the overall samples, intro-multiplexing samples, and extro-multiplexing samples, respectively.

Table 4
Comparison of different graph representations.

Data/Model	Tri-C			Arg-C		
	P	R	F1	P	R	F1
ACE05-EN+/HetGraph	74.1	76.4	75.3	59.8	49.4	54.1
ACE05-EN+/MultiGraph	75.2	76.4	**75.8**	63.8	53.3	**58.0**
ERE-EN/HetGraph	58.5	60.0	59.3	55.9	42.4	48.3
ERE-EN/MultiGraph	62.8	61.1	**61.7**	52.5	50.5	**51.2**

5. Analytical experiments

In this section, three issues regarding the multigraph event representation are discussed. The first issue concerns the ability of a multigraph to represent event structures. In the second issue, the computational complexity of multigraph-based event extraction was analyzed. Finally, a case study is given to show the effectiveness of the MGREE model.

5.1. Analysis of representation method

To demonstrate the ability of the multigraph to represent an event structure, this experiment compares our multigraph event representation with another event representation known as a heterogeneous graph [47,48].

The heterogeneous graph also supports different types of edges in the event-extraction graph. The main problem for the heterogeneous graph representation is that only one categorized directed edge is allowed between nodes, which cannot resolve the intra-multiplexing problem. In the heterogeneous graph representation, no threshold was used to filter edges between nodes. For making a fair comparison, our model is implemented without the threshold search. Table 4 shows the experimental results, where the heterogeneous graph event extraction is denoted to "HetGraph".

The results in Table 4 show that, in the trigger identification task, our model has similar performance as the 'HetGraph" model. The reason is that there is no multiplexing problem in triggers. However, in the argument extraction task, our multigraph representation achieves a significant improvement over the heterogeneous graph, where the F1 value is improved by 4.0% on the ACE05-EN dataset and 3.9% on the ACE05-EN+ dataset. The reason is that there is one edge between two nodes in the heterogeneous graph. Because the reason of the argument multiplexing problem, the heterogeneous graph is not effective to represent the event structures, which leads to lower performance.

5.2. Analysis of computational complexity

As discussed in Section 3.1, the semantic structure of an event is represented as multigraph, where nodes denote to tokens in a sentence. The edges between nodes represent semantic roles between tokens, e.g., argument types or argument roles. The multigraph allows multi-parallel edges between any two nodes. It can be represented by an $N \times N \times K$ matrix $\mathbb{E}$, where $\mathbb{E}_{ijk}$ denotes to an edge (represented as $\langle n_i^h, n_j^t \rangle$) with label $y_k \in \mathbf{Y}$. In order to show the computational complexity of our model, in this section, we analyzed the computational complexity of the multigraph in event extraction.

In this experiment, three event extraction models are conducted for comparison: Text2Event [29], OneIE [27] and our MGREE model. In this experiment, we use the ACE05-EN+ dataset for evaluation. The length of inputs is set to 128. All parameters use the default values in the official implementation of Text2Event [29] and OneIE [27]. The results are listed in Table 5.

Table 5
Comparison on Computational Complexity.

Model	PTM	Par.	Time	Tri-C	Arg-C
Text2Event	T5-large	836M	149.3/417.3	71.8	54.4
OneIE	BERT-large	338M	142.0/168.7	72.8	54.8
MGREE	BERT-large	377M	17.1/34.8	75.7	57.1
MGREE	DeBERTa-large	477M	26.3/53.7	76.1	58.8

Table 6
Case studies of intro-multiplexing and extro-multiplexing examples. Red and blue words represent triggers and arguments, respectively.

1. Extro-multiplexing case: **He** will then stay on for a regional **summit** before **heading** to **Saint Petersburg** for celebrations marking the 300th anniversary of the city's founding.		
Gold	**Trigger**	Contact-Meet: summit; Movement-Transport: heading.
	Arguments	Entity(Contact-Meet): He; Artifact(Movement-Transport): He; Destination(Movement-Transport): Saint Petersburg.
OneIE	**Trigger**	Contact-Meet: summit; Movement-Transport: heading.
	Arguments	Entity(Contact-Meet): He; Artifact(Movement-Transport): He; Place(Contact-Meet): Saint Petersburg; Destination(Movement-Transport): Saint Petersburg.
MGREE	**Trigger**	Contact-Meet: summit; Movement-Transport: heading.
	Arguments	Entity(Contact-Meet): He; Artifact(Movement-Transport): He; Destination(Movement-Transport): Saint Petersburg.
2. Intro-multiplexing case: for a week now, **protesters** have been **clashing** with **police** and **backers** of the clerical regime in **tehran**.		
Gold	**Trigger**	Conflict.Attack: clashing.
	Arguments	Attacker: protesters; Target: protesters; Attacker: police; Target: police; Attacker: backers; Target: backers; Place: tehran.
OneIE	**Trigger**	Conflict.Attack: clashing.
	Arguments	Attacker: protesters; Attacker: police; Attacker: backers; Place: tehran.
MGREE	**Trigger**	Conflict.Attack: clashing.
	Arguments	Attacker: protesters; Target: protesters; Attacker: police; Target: police; Attacker: backers; Target: backers; Place: tehran.

In the above table, "PTM" represents the pretrained language models used by the models. "Par" denotes to the total number of model parameters. "Time" refers to the inference time. The average inference time (ms) on a single sentence is used. We compare the speed on batch sizes of 8 and 1, denoted as the form "*/#", which refers to the time cost for each sentence in batch sizes of 8 (*) or 1 (#). For a fair comparison, we also show experimental results on BERT-large weights in the "Tri-C" and "Arg-C" columns. They denote to the performance on trigger detection and argument identification.

As shown in Table 5, the inference speed of MGREE is almost seven times faster than that of Text2Event and OneIE. The result indicates that the multigraph event representation is more effective, because the three-dimensional tensor representation is conducive to GPU parallel computing. Furthermore, MGREE uses only simple filtering and combining operations to decode the event structure, which is faster than the beam search decoding used by other two models. When the batch size is set to 1, the inference time is increased significantly because Text2Event needs to decode every sentence word by word. However, MGREE is still much faster than the two models. It further proves the effectiveness of the MGREE model.

5.3. Case study

In this section, we present a case study to demonstrate the behavior of MGREE in resolving semantic structures of events. For each argument multiplexing event type (intro-multiplexing or extro-multiplexing), we selected one sentence from the ACE05-Multiplexing dataset for analysis. The results are shown in Table 6, where words in red and blue colors represent triggers and arguments, respectively. When a sentence contains more than one event, they are parenthesized after the argument role type.

For the extro-multiplexing problem, both OneIE and MGREE can identify arguments simultaneously shared by different events. However, OneIE uses a pairwise classifier. It is susceptible to local semantic interference. The model incorrectly predicts "Saint

Petersburg" as the location-type argument for the *Meet* event. On the other hand, MGREE uses a global scoring mechanism to predict the association between nodes, which can take global semantic information to support the decision-making process.

The OneIE model infers argument roles by predicting a unique relationship between each entity and trigger pair. However, because the role of arguments is not unique in the event extraction task, it is shown in the intro-multiplexing case that the OneIE model can predict only one of the roles of multiplexing arguments. This problem results in a seriously incomplete event structure. The MGREE model allows parallel edges. Therefore, it has the ability to simultaneously identify intro-multiplexing arguments. Therefore, MGREE guarantees the integrity of the event structure.

6. Conclusion

Event extraction is a hot research topic and has received significant attention in the information extraction field. However, events usually have complex semantic structures, which creates challenges to represent them. This study proposes a multigraph representation for encoding the semantic structure of events. Experiments show that our method can directly extract all overlapped events in a sentence, including events involving in the argument multiplexing problem. It is effective to handle the argument multiplexing problem. The experimental results also show that the multigraph representation is robust in terms of both event representation and computational efficiency. We achieved a state-of-the-art event extraction performance on the ACE05 dataset. In future work, the multigraph representation can be developed to support tasks in other fields, such as entity-relation joint extraction.

CRediT authorship contribution statement

Hui Huang: Writing – original draft, Methodology, Conceptualization. **Yanping Chen:** Writing – review & editing, Supervision, Funding acquisition. **Chuan Lin:** Supervision, Project administration, Investigation. **Ruizhang Huang:** Supervision, Project administration, Funding acquisition, Data curation. **Qinghua Zheng:** Supervision, Project administration, Investigation. **Yongbin Qin:** Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The authors do not have permission to share data.

Acknowledgements

This work is supported by the National Natural Science Foundation of China under Grant No. 62066008 and No. 62066007, the National Key R&D Program of China under Grant No. 2023YFC3304500 and the Major Science and Technology Projects of Guizhou Province under Grant [2024]003.

References

[1] G.R. Doddington, A. Mitchell, M.A. Przybocki, L.A. Ramshaw, S.M. Strassel, R.M. Weischedel, The automatic content extraction (ace) program-tasks, data, and evaluation, in: Lrec, vol. 2, Lisbon, 2004, pp. 837–840.
[2] X. Liu, Z. Luo, H. Huang, Jointly multiple events extraction via attention-based graph information aggregation, arXiv preprint arXiv:1809.09078, 2018.
[3] Y. Chen, Z. Ding, Q. Zheng, Y. Qin, R. Huang, N. Shah, A history and theory of textual event detection and recognition, IEEE Access 8 (2020) 201371–201392.
[4] G. Glavaš, J. Šnajder, Event graphs for information retrieval and multi-document summarization, Expert Syst. Appl. 41 (15) (2014) 6904–6916.
[5] W. Li, D. Cheng, L. He, Y. Wang, X. Jin, Joint event extraction based on hierarchical event schemas from framenet, IEEE Access 7 (2019) 25001–25015.
[6] J. Wang, A. Jatowt, M. Färber, M. Yoshikawa, Improving question answering for event-focused questions in temporal collections of news articles, Inf. Retr. J. 24 (1) (2021) 29–54.
[7] C.-Y. Liu, C. Zhou, J. Wu, H. Xie, Y. Hu, L. Guo, Cpmf: a collective pairwise matrix factorization model for upcoming event recommendation, in: 2017 International Joint Conference on Neural Networks (IJCNN), IEEE, 2017, pp. 1532–1539.
[8] Y. Chen, L. Xu, K. Liu, D. Zeng, J. Zhao, 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), 2015, pp. 167–176.
[9] F. Li, W. Peng, Y. Chen, Q. Wang, L. Pan, Y. Lyu, Y. Zhu, Event extraction as multi-turn question answering, in: Findings of the Association for Computational Linguistics: EMNLP 2020, 2020, pp. 829–838.
[10] J. Liu, Y. Chen, K. Liu, W. Bi, X. Liu, Event extraction as machine reading comprehension, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2020, pp. 1641–1651.
[11] X. Du, C. Cardie, Event extraction by answering (almost) natural questions, arXiv preprint arXiv:2004.13625, 2020.
[12] Q. Li, J. Li, J. Sheng, S. Cui, J. Wu, Y. Hei, H. Peng, S. Guo, L. Wang, A. Beheshti, et al., A compact survey on event extraction: Approaches and applications, arXiv e-prints (2021) arXiv–2107.
[13] T. Nguyen, R. Grishman, Graph convolutional networks with argument-aware pooling for event detection, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32, 2018.
[14] S. Liu, R. Cheng, X. Yu, X. Cheng, Exploiting contextual information via dynamic memory network for event detection, arXiv preprint arXiv:1810.03449, 2018.

[15] H. Lin, Y. Lu, X. Han, L. Sun, Cost-sensitive regularization for label confusion-aware event detection, arXiv preprint arXiv:1906.06003, 2019.
[16] N. Ding, Z. Li, Z. Liu, H. Zheng, Z. Lin, Event detection with trigger-aware lattice neural network, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 2019, pp. 347–356.
[17] M. Tong, B. Xu, S. Wang, Y. Cao, L. Hou, J. Li, J. Xie, Improving event detection via open-domain trigger knowledge, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020, pp. 5887–5897.
[18] A.P.B. Veyseh, V. Lai, F. Dernoncourt, T.H. Nguyen, Unleash gpt-2 power for event detection, in: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), 2021, pp. 6271–6282.
[19] X. Wang, Z. Wang, X. Han, Z. Liu, J. Li, P. Li, M. Sun, J. Zhou, X. Ren, Hmeae: hierarchical modular event argument extraction, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 2019, pp. 5777–5783.
[20] J. Ma, S. Wang, R. Anubhai, M. Ballesteros, Y. Al-Onaizan, Resource-enhanced neural model for event argument extraction, arXiv preprint arXiv:2010.03022, 2020.
[21] J. Liu, Y. Chen, J. Xu, Machine reading comprehension as data augmentation: a case study on implicit event argument extraction, in: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, 2021, pp. 2716–2725.
[22] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: pre-training of deep bidirectional transformers for language understanding, arXiv preprint arXiv:1810.04805, 2018.
[23] S. Yang, D. Feng, L. Qiao, Z. Kan, D. Li, Exploring pre-trained language models for event extraction and generation, in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 2019, pp. 5284–5294.
[24] T.H. Nguyen, K. Cho, R. Grishman, Joint event extraction via recurrent neural networks, in: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2016, pp. 300–309.
[25] T.M. Nguyen, T.H. Nguyen, One for all: neural joint modeling of entities and events, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 6851–6858.
[26] D. Wadden, U. Wennberg, Y. Luan, H. Hajishirzi, Entity, relation, and event extraction with contextualized span representations, arXiv preprint arXiv:1909.03546, 2019.
[27] Y. Lin, H. Ji, F. Huang, L. Wu, A joint neural model for information extraction with global features, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020, pp. 7999–8009.
[28] G. Paolini, B. Athiwaratkun, J. Krone, J. Ma, A. Achille, R. Anubhai, C.N.d. Santos, B. Xiang, S. Soatto, Structured prediction as translation between augmented natural languages, arXiv preprint, arXiv:2101.05779, 2021.
[29] Y. Lu, H. Lin, J. Xu, X. Han, J. Tang, A. Li, L. Sun, M. Liao, S. Chen, Text2event: controllable sequence-to-structure generation for end-to-end event extraction, arXiv preprint arXiv:2106.09232, 2021.
[30] Y. Lu, Q. Liu, D. Dai, X. Xiao, H. Lin, X. Han, L. Sun, H. Wu, Unified structure generation for universal information extraction, arXiv preprint arXiv:2203.12277, 2022.
[31] I.-H. Hsu, K.-H. Huang, E. Boschee, S. Miller, P. Natarajan, K.-W. Chang, N. Peng, Degree: a data-efficient generation-based event extraction model, in: Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2022, pp. 1890–1908.
[32] X. Liu, H. Huang, G. Shi, B. Wang, Dynamic prefix-tuning for generative template-based event extraction, in: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Dublin, Ireland, 2022, pp. 5216–5228, https://aclanthology.org/2022.acl-long.358.
[33] Y. Luan, D. Wadden, L. He, A. Shah, M. Ostendorf, H. Hajishirzi, A general framework for information extraction using dynamic span graphs, 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), Association for Computational Linguistics, Minneapolis, Minnesota, 2019, pp. 3036–3046, https://aclanthology.org/N19-1308.
[34] D. Hendrycks, K. Gimpel, Gaussian error linear units (gelus), arXiv preprint arXiv:1606.08415, 2016.
[35] Y. Sun, C. Cheng, Y. Zhang, C. Zhang, L. Zheng, Z. Wang, Y. Wei, Circle loss: a unified perspective of pair similarity optimization, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 6398–6407.
[36] P. Chen, N. Bogoychev, K. Heafield, F. Kirefu, Parallel sentence mining by constrained decoding, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, 2020, pp. 1672–1678, https://aclanthology.org/2020.acl-main.152.
[37] N. De Cao, G. Izacard, S. Riedel, F. Petroni, Autoregressive entity retrieval, arXiv preprint arXiv:2010.00904, 2020.
[38] T. Zhang, H. Ji, A. Sil, Joint entity and event extraction with generative adversarial imitation learning, Data Intell. 1 (2) (2019).
[39] Z. Song, A. Bies, S. Strassel, T. Riese, J. Mott, J. Ellis, J. Wright, S. Kulick, N. Ryant, X. Ma, From light to rich ere: annotation of entities, relations, and events, in: Proceedings of the 3rd Workshop on EVENTS: Definition, Detection, Coreference, and Representation, 2015, pp. 89–98.
[40] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, et al., Transformers: state-of-the-art natural language processing, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, 2020, pp. 38–45.
[41] I. Loshchilov, F. Hutter, Decoupled weight decay regularization, arXiv preprint arXiv:1711.05101, 2017.
[42] S. Wang, M. Yu, S. Chang, L. Sun, L. Huang, Query and extract: refining event extraction as type-oriented binary decoding, arXiv preprint arXiv:2110.07476, 2021.
[43] J. Lu, P. Yang, R. Gan, J. Wang, Y. Zhang, J. Zhang, P. Zhang, Uniex: an effective and efficient framework for unified information extraction via a span-extractive perspective, arXiv preprint arXiv:2305.10306, 2023.
[44] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al., Llama 2: open foundation and fine-tuned chat models, arXiv preprint, arXiv:2307.09288, 2023.
[45] E.J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, Lora: low-rank adaptation of large language models, arXiv preprint arXiv:2106.09685, 2021.
[46] Y. Ma, Y. Cao, Y. Hong, A. Sun, Large language model is not a good few-shot information extractor, but a good reranker for hard samples!, arXiv preprint arXiv:2303.08559, 2023.
[47] C. Zhang, D. Song, C. Huang, A. Swami, N.V. Chawla, Heterogeneous graph neural network, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 793–803.
[48] H. Peng, J. Li, Y. Song, R. Yang, R. Ranjan, P.S. Yu, L. He, Streaming social event detection and evolution discovery in heterogeneous information networks, ACM Trans. Knowl. Discov. Data 15 (5) (2021) 1–33.