applied sciences

MDPI

Article

Sudden Event Prediction Based on Event Knowledge Graph

Yanhao Li and Wei Liu *

School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China
* Correspondence: liuw@shu.edu.cn

Abstract: Event prediction is a knowledge inference problem that predicts the consequences or effects of an event based on existing information. Early work on event prediction typically modeled the event context to predict what would happen next. Moreover, the predicted outcome was often singular. These studies had difficulty coping with both the problems of predicting sudden events in unknown contexts and predicting outcomes consisting of multiple events. To address these two problems better, we present the heterogeneous graph event prediction model (HGEP), which is based on an event knowledge graph. To cope with the situation of missing contexts, we propose a representation learning method based on the heterogeneous graph transformer. We generate event scenario representations from arguments of the initial event and other related concurrent events for the event prediction. We improve the prediction ability of the HGEP model through prior knowledge provided by scenario models in the event knowledge graph. To obtain multiple prediction outcomes, we design a scoring function to calculate the score of the occurrence probability of each event class. The event classes with scores higher than a priori values are adopted as prediction outcomes. In this paper, we create an event knowledge graph in the domain of transportation for an event prediction experiment. The experimental results show that the HGEP model can effectively make predictions with event scenario representations and has a more accurate matching rate and higher precision than the baseline model.

Keywords: event knowledge graph; event prediction; event scenario representation; heterogeneous graph neural network

Citation: Li, Y.; Liu, W. Sudden Event Prediction Based on Event Knowledge Graph. *Appl. Sci.* **2022**, *12*, 11195. https://doi.org/10.3390/app122111195

Academic Editor: Tobias Meisen

Received: 2 September 2022
Accepted: 2 November 2022
Published: 4 November 2022

1. Introduction

Event prediction is a knowledge inference problem that predicts the consequences or effects of an event based on existing information. The predecessor of the event prediction task was a kind of evaluation method proposed by Chambers and Jurafsky [1], which selected subsequent events from a pool of candidate events based on known event information. A series of subsequent studies on event prediction extended Chambers' work [2–6]. One drawback of the studies that follows Chambers' method, however, is that they relied on a long chain of events in order to make predictions. This chain structure performs well in predicting a single event but is not suitable for predicting multiple concurrent subsequent events. Furthermore, event prediction based on the chain structure ignores the effect of some features of the event itself on the event prediction. For example, subsequent events, including the injuries and deaths of passengers and rescue events, can be predicted based on the analysis of the features of traffic accidents.

In recent years, more and more attention has been paid to the application of event prediction. Studies have been designed to solve event prediction difficulties in a specific application area or in a real-world situation. For example, estimating medications a patient will take next time based on their medical history [7], predicting civil unrest in 10 Latin American nations [8], and other fields [9,10]. These studies, similar to Chambers', also have a clear event context. However, we have observed that some sudden events, such as chemical plant explosions, traffic accidents, and other accidents, frequently take place

without evidence. Usually, the context of these accidents is implicit. Once these events happen, people can reduce the loss by predicting subsequent events and taking some actions. In this paper, we study a subsequent events prediction method for sudden events. Subsequent events prediction for sudden events usually faces the following challenges:

1. The prediction result of a sudden event is often related to the features of the event itself, including the actions and objects involved, the condition of the event, and the scene (place and circumstances) of the event, therefore to classify a sudden event, such as natural disaster or terrorist attack, it is quite challenging to obtain effective feature information for the sudden event.
2. An effective event prediction method requires predicting all possible subsequent events of an initial event, and the features of an initial event determine whether it has one or several concurrent subsequent events. For example, the subsequent events of "drowning" may include "first aid treatment" and "call for police". If the person in the drowning event has been "dead", then the subsequent event will only be "call for police".

In this paper, we provide a heterogeneous graph event prediction model (HGEP) based on an event knowledge graph to address the aforementioned challenges. An event knowledge graph can provide a priori knowledge for events prediction. For one thing, it can provide feature information for the analysis of initial events. Furthermore, it can provide scenario knowledge for the prediction of events. In other words, based on an event knowledge graph, the event type of the initial sudden event can be identified, and then the scenario model in the event knowledge graph can be used to predict the probability of its subsequent events. In the event knowledge graph, triples are used to describe the semantic relation between events and the relation between events and event arguments (action, object, place, time, and so on). Triples can also define the type (event class) of each event instance. On this basis, an event knowledge graph in a specific domain provided event scenario models to represent occurrence patterns of different events, which are described by a set of subevents and logical relations (including cause, sequence, concurrence, and so on) between them.

In recent years, the research on reasoning based on knowledge graphs has become an important area of knowledge reasoning. It mainly revolves around relational reasoning; by understanding the characteristic information of entities, relations, and knowledge graph structures, unknown facts or relations are inferred based on the existing facts or relations of the knowledge graph. Reasoning based on the knowledge graph can primarily assist in reasoning about new facts, new relations, new rules, etc. Similarly, we use the constructed event knowledge graph, and with the help of the concept of knowledge graph reasoning, it is possible to reason about new events from known event features, that is, to predict the occurrence of new events. Since the research on event knowledge graphs has just started, reasoning and event prediction based on event knowledge graphs are still in their initial stages. Extending the reasoning research on knowledge graphs to event knowledge graphs is also the motivation of this paper. In previous studies, representation learning on knowledge graphs [11–14] mapped sparse graphs into dense vector spaces such that entities and relations with a high semantic similarity had similar embedding vectors. The corresponding tail nodes were then inferred by understanding the semantic information of the head nodes and relations in the triplet. Then, theoretically, representation learning on the event knowledge graph could infer the corresponding subsequent events by understanding the semantic information of known events and relations. Therefore, we design a novel event representation learning method. Using this approach, the features of an initial event and its concurrent events (if they exist) can be learned from the event knowledge graph, and then its subsequent events can be predicted. We store the events that have occurred and their subsequent events in the event knowledge graph. The logic of subsequent events occurring is then captured by extracting the features of the initial event and its concurrent events. When a new event occurs, the machine quickly makes a reliable prediction based on past events with similar features.

We use the method in [15] to construct a knowledge graph of events in the field of transportation for the experiments. An "event" is composed of a trigger and four types of arguments: time, place, object, and participants. Events and arguments are connected to each other by different argumentative relations. Through a concurrent relation, concurrent events are linked to one another. The concurrent events and their arguments form a heterogeneous graph. We learn the representation of each such heterogeneous graph to obtain event scenario representations. Specifically, we use a heterogeneous graph transformer (HGT) [16] to fuse the entity embedding of concurrent events in an inductive manner. It differs from the common graph attention network by adding a reverse edge to each directional edge in a graph, which facilitates the flow of information through the graph. We use a multilabel classification approach to project event scenario representations onto the result set to handle the presence of events in subsequent events. To determine whether a specific event type would show up in the results, we create scoring functions.

We summarize our main contributions as follows:

1. An event representation method based on a HGT is proposed, which efficiently obtains an embedding containing event semantic information from sparse heterogeneous graphs composed of event arguments.
2. An event prediction method based on the event knowledge graph is proposed. The method understands the semantic relation between events in the knowledge graph through representation learning and predicts subsequent events according to the logical relation of events in the event scenario model.
3. An event knowledge graph in the domain of transportation is constructed.

The rest of the paper is organized as follows. Section 2 describes the related research on event prediction. Section 3 gives some preliminary knowledge about event knowledge graphs and our modeling of the prediction task. Section 4 presents the prediction methods based on event feature representation. Experimental results are given in Section 5. Finally, the outlook of the next research is discussed in Section 6.

2. Related Work

Most existing event prediction studies were originally designed to deal with specific application domains, such as predicting adverse drug reaction events in the medical field [17]. Classical machine learning algorithms were used in the early stages of these studies. In recent years, graph-based algorithms have frequently been used to solve the problem of event prediction.

Traditional event prediction methods use machine learning methods such as classification, clustering, and frequent-pattern mining to predict whether an event will occur based on statistical data. Laxman et al. [18] designed a frequent-episode mining framework to discover a continuous sequence of events in an event stream based on a discrete hidden Markov model. Furthermore, based on this, the framework was extended to support multiple sequences in a dataset [19]. Zhou et al. [20] further ranked the prediction results by confidence and output the top k events as the final result. Ballings et al. [21] used logistic regression and a classification tree model to predict customer churn events. Caigny et al. [22] proposed a novel logit leaf model to predict customer churn events by fusing a logistic regression and decision tree models. Korkmaz et al. [23] fused data sources such as Twitter, blogs, and news, using logistic regression models with Lasso to select sparse features in order to predict the probability of occurrence of civil unrest events. Radinsky et al. [24] used the HAC hierarchical clustering algorithm to aggregate similar events in an abstract event-centric manner and extract the generalized causal relation between two events to predict events in a rule-generating manner. Granroth-Wilding et al. [2] predicted events in the form of event chains, using a composite neural network that predicted whether two events would appear in the same event chain by learning a vector representation of event predicates and parameter nouns. Pichotta et al. [25] proposed a script learning model using LSTM, where a script was a sequence of prototypical events represented by multiple predicates as a way to predict future events. Wang et al. [6] also used LSTM for

scripted event learning, modeling pairs of events based on the relation between the event sequence information and event pairs. Compared with Pichotta, Wang's method had better performance on the event-pair model. Ramakrishnan et al. [8] collected data in numerous media for their analysis and used logistic regression and LASSO to select sparse feature sets and built separate models for different countries to predict their probability of civil unrest. Damaschke et al. [26] utilized a Weibull distribution to model volcano eruption events.

With the growing popularity of graph neural network [27] in recent years, there have been many studies on constructing event graphs from events or event elements and then using graph-based algorithms and techniques to solve event prediction problems. Li et al. proposed a scaled graph neural network (SGNN) [4] to predict script events. Li used triples <subject, predicate, object> to represent events and constructed a narrative-based event evolution graph using causal chains of events to model the event prediction as choosing the correct answer among one correct event and four random events. Yang et al. proposed a narrative graph neural network [5] to predict script events. Differing from Li, Yang used a combination of information from event chains, event graphs, and event pairs, and proposed a prediction task to predict intermediate events based on previous and subsequent events. Li et al. [28] proposed a method for predicting events with an autoregressive graph generation model, which constructed event instance graphs based on a time axis, where each event instance graph contained a complex event involving all entities and predicting the event type and possible event parameters using pattern induction. Hu et al. [29] proposed a dynamic graph structure constructed from time series data, designed with temporal attention to simulate the effect of time and predict event occurrence with structural changes in the graph. Song et al. [10] used event participants as nodes and event types as edges, modeled social events in open-source media as a temporal event graph, and extracted keywords in the text to enhance semantics, and predicted multiple co-occurring events targeting the future. Liu et al. [30] constructed a sequential event graph to model the sequential event evolution process in the physical world and adopted the SGNN to predict events. Liu et al. [7] predicted the next stage of patient prescriptions on a temporal medical event graph to predict different drugs with a multilabel classification task. Luo et al. [9] used a dynamically constructed heterogeneous graphs to encode the attributes of surrounding events as well as the environment, using a multilayer graph neural network to learn the influence of historical behavior and the surrounding environment on the current event, and generating an efficient event representation that was used on the DiDi platform and significantly improved the prediction results. Our study is also a graph-based approach. The difference from the event diagram in the above method is the different event definition and composition methods.

3. Preliminaries

With "events" as the knowledge unit, the event knowledge graph defines events, event classes, and event (class) relations. Furthermore, the event knowledge graph contains a set of scenario models that express the rules and patterns of event occurrence or evolution. Our proposed event prediction method is based on the above knowledge. For the completeness of this paper, the relevant definitions of an event knowledge graph, as well as a description of the event prediction, are provided in this section.

3.1. Event Knowledge Graph Model

In the event knowledge graph, the event occurs at a specific time and in a specific environment, involving several actors, showing specific action characteristics and state changes. We use the definition of event in [15] to describe an event in the event knowledge graph.

$$E ::= \langle A, P, O, L, T \rangle \quad (1)$$

An event is defined as a unit of knowledge containing five elements. A is the set of actions contained in the event. P is the set of participants of the event. O is the set of objects or bearers of the event. L is the location where the event occurs. T is the period of time during which the event occurs or lasts. There are elemental relations (including hasAction, hasParticipant, hasObject, hasLocation, hasTime) between events and event elements.

The event knowledge graph is an event-based knowledge base built for different application domains, containing an event ontology and a large number of event instances. Its definition can be described as follows.

$$EKG ::= \langle EOs, EIs \rangle \tag{2}$$

where EOs and EIs denote the event ontology and the event instance library, respectively. The event ontology includes: (1) event class concepts, (2) event class relations, (3) assertion and inference rules for the event classes. The definition of the event ontology can be described as follows.

$$EOs ::= \langle ECs, Event\_Relations, Rules \rangle \tag{3}$$

ECs represent the set of event classes. $Event\_Relations$ represent the set of event class relations, including a taxonomy relation (is_a) and logical relations (including causal, isComposedOf, follow, and concurrence). $Rules$ represent a collection of inference rules for implementing various kinds of reasoning for event classes. The event ontology contains an event hierarchy model and a set of event scenario models. The event scenario model is formed by associating the event classes through logical relations. Furthermore, it reflects the pattern about the occurrence of an initial event and a set of subsequent events triggered by it. The event hierarchy model consists of event classes and a taxonomy relation.

The event instance library contains a large number of event instances, defined as follows.

$$EIs ::= \langle Events, Event\_Relations \rangle \tag{4}$$

where $Events$ represents a collection of event instances and $Event\_Relations$ represents the relations between event instances. An event instance is a concretization of the event class concept and represents a specific event. Event instances can commonly be obtained from text or from a database. The same logical relations exist between event instances as between event classes, expressing the internal logic and evolutionary patterns between events. The structure of the event knowledge graph model is shown in Figure 1.

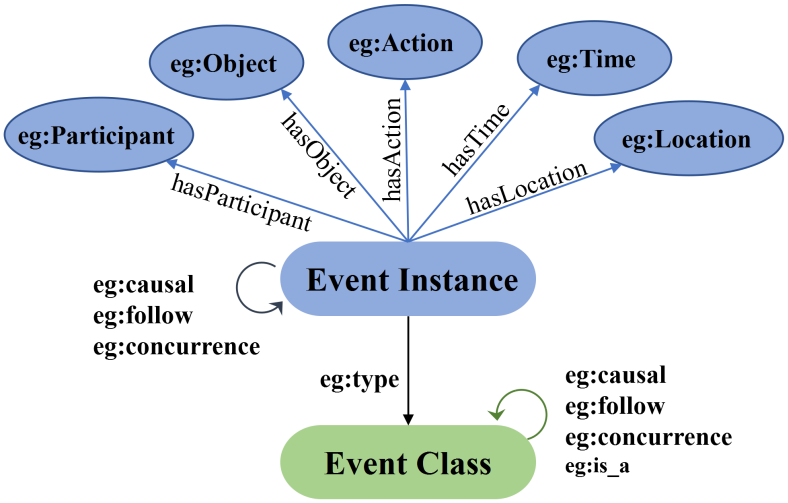

Figure 1. Event knowledge graph model structure.

Event knowledge graphs are represented and stored through extended RDF(s) to achieve a normalized description and storage of event knowledge, i.e., event knowledge

graphs are represented and stored through RDF triples. The main triple schema of relations is defined in Table 1.

Table 1. Schema of relations.

Relation Model	Ternary Mode and Description
Relation between event classes	$<ec_1, ec\_relation, ec_2>$, ec_1 and ec_2 are event classes, *ec_relation* is the event class relation
Relation between event class and event instance	$<e_1, type, ec_1>$, e_1 is the event instance, ec_1 is the event class
Relation between event instance and event element	$<e_1, element\_relation, element\_entity_1>$, e_1 is the event instance, $element\_entity_1$ is the element entity, *element_relation* is the event element relation
Relation between event instances	$<e_1, event\_relation, e_2>$, e_1 and e_2 are event instances, *event_relation* is the event relation

3.2. Event Prediction by Scenario Model

For event prediction based on event knowledge graphs, our expectation is for the machine to understand "what happened" by perceiving the current scenario, and then give "what will happen next". To make it easier to understand, let us imagine a scenario. A boy opens the door of his home, and their perceptual ability makes them instantly aware of some events that are happening. Event 1: his teacher is talking to their parents; event 2: his parents show a frustrated expression. Perhaps the boy has seen such a scenario before, and his experience lets him know that the situation is not good; maybe there will be a "storm" behind. It is this ability to understand situations from events and react to them that we want to equip machines with. As mentioned before, the features of the event scenario information are extracted from the initial event and all possible concurrent events. We formalized the event scenario information as follows:

$$ES = EI_e(A_e, O_e, L_e, T_e, P_e) + \sum_n EI_{ec}(A_{ec}, O_{ec}, L_{ec}, T_{ec}, P_{ec}) \quad (5)$$

EI_e denotes the initial event e containing the argument information, and EI_{ec} denotes the event information of concurrent event ec of e. ES is the feature vector of the scenario information. We modeled the prediction of event occurrence from the scene information as a multilabel classification task. We designed prediction functions to parse the scenario information to obtain the probability of occurrence for each event category. First, a linear function was used as the projection function to project the scenario information into different result spaces to obtain the result vector. Then, we designed an analytic function to obtain the probability of occurrence of each event class corresponding to the result vector.

$$W_R = Linear(ES_e) \quad (6)$$

$$P(e) = Sigmoid(log\_softmax(W_R)) \quad (7)$$

4. Methods

The HGEP model is mainly composed of three parts. Firstly, some events in the event knowledge graph are obtained by the event graph extraction module to build the event subgraph. Then, arguments in text form are embedded with a pretrained XLNet [23] and fed to the scenario learning model. The scenario learning model uses a heterogeneous mutual attention graph neural network to learn event scenario representations with scenario information. Finally, the event scenario representation is parsed to obtain the event prediction results. The model is shown in Figure 2. The HGEP model needs to be based on a well-constructed event knowledge graph. Before describing the prediction model, we provide the event knowledge graph construction method for the transportation field in the first section.

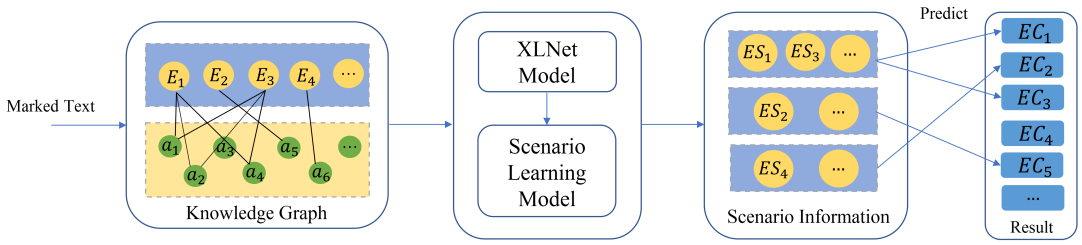

Figure 2. Event prediction model.

4.1. Event Knowledge Graph Construction—Taking the Traffic Domain as an Example

Based on the event knowledge graph structure given in Section 3.2, this section manually constructs an event knowledge graph for the transportation domain. The construction process firstly collected the traffic news corpus and obtained coarse-grained event classes from the news under the guidance of experts. Secondly, the event granularity was refined layer by layer to get the event classes applicable to event prediction. Then, several event scenario models were constructed using several different event classes, respectively. The relations between the event classes were obtained in the scenario models. Finally, arguments and event relations were labeled from the corpus according to the scenario models and stored. The labeling format is shown in Figure 3. We used the <*Event*> tag to mark event instances, and the Class attribute to indicate the event class to which the event instance belonged. Each event instance had a unique ID. In the event instance, we marked its arguments with <*Denoter*>, <*Object*>, <*Participant*>, <*Location*>, and <*Time*> tags. <*Relations*> brought together all the <*eRelation*> tags. Each <*eRelation*> marked the relation between an event pair and identified this relation with the *relType* attribute. For different event relations, we recorded the event pairs using different attributes. For a causal relation, *cause_eid* denoted the initial event and $effect\_eid$ denoted the subsequent event. For a follow relation, *aevent_eid* denoted the initial event and *bevent_eid* denoted the subsequent event. For a concurrence relation, we also used *aevent_eid* and *bevent_eid*. Due to the special symmetry of Concurrence, the meaning of these two attributes in Concurrence was equivalent.

```
<Event Class="TrafficAccident_CarCrash" Eid="E0B">
        <Denoter Did="D0B">collided</Denoter>
        <Object Oid="O0B">two cars</Object>
        <Participant Pid="P0B"></Participant>
        <Location Lid="L0B">Interstate 94, Minnesota, USA</Location>
        <Time Tid="T0B">December 25, 2021</Time>
</Event>

<Relations>
        <eRelation relType="Causal" cause_eid="E0B" effect_eid="E0C"/>
        <eRelation relType="Follow" bevent_eid="E0E" aevent_eid="E0D"/>
</Relations>
```

Figure 3. Data annotation format.

We collected hundreds of news articles about traffic from China News Network and defined the event classes under the guidance of experts while referring to other field. In this paper, the coarse-grained event categories in the transportation domain were divided into nine categories. Namely road conditions, vehicle loss of control, traffic accidents, road rescue, casualties, impeded safe driving, illegal driving, accident handling, and criminal punishment. Then, the final event class was obtained by subdividing the coarse-grained event class. For example, we subdivided the road condition to get the snow event class, rain event class, and road congestion event class, and subdivided the traffic accident event class to get the crash event class, rear-end event class, rollover event class, and collision event class, etc. Finally, we obtained a total of thirty-five fine-grained event classes. In this paper, we constructed several complex scenario models in the fine-grained event classes,

and took the example of a vehicle collision, hitting people, and tailgating; the scenario model is shown in Figure 4.

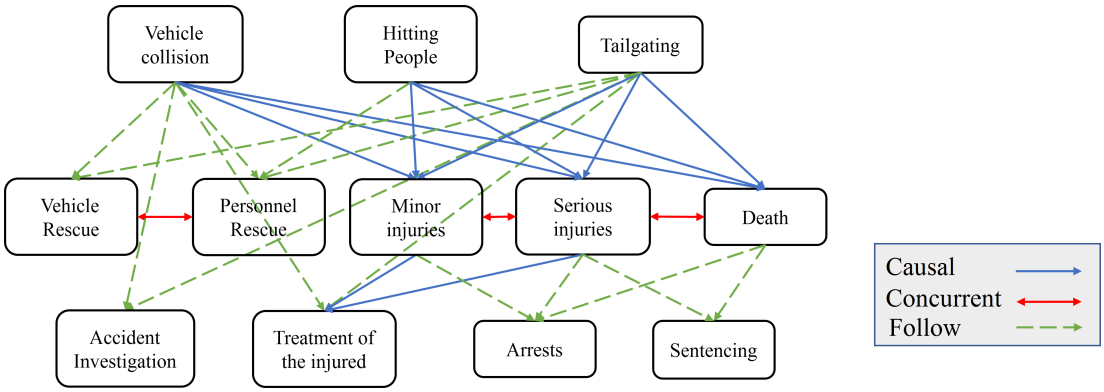

Figure 4. Part of traffic accident scenario model.

Guided by the scenario model, we annotated event instances and their relations on a corpus of more than one hundred news items to obtain a knowledge graph of events in the transportation domain, which contained 510 events and 500 event relation pairs.

4.2. Event Graph Extraction

An event knowledge graph contains many event instances. An event instance is composed of arguments, which itself is equivalent to an empty node. For each event instance, we used the Chinese character of "event" as the default text to generate the central node and the text of the arguments to generate the argument nodes. A central node connected the corresponding argument nodes, which represented an event instance. The central nodes of all event instances were connected into a subgraph according to the relations in the event knowledge graph. We used the generated subgraphs as the inputs to the graph neural network. The generation process is shown in Algorithm 1.

We first needed to extract the seed events, i.e., event instances in the training set and test set and add them to E_{seed}. Then, we extracted their neighbor nodes and corresponding relations based on the seed events and added each of them to E_{eg} and R_{eg}. We traversed R_{eg} and extracted the event instance node pairs from E_{seed} and E_{eg}. Meanwhile, we obtained the argument nodes corresponding to the event instance nodes. We connected these nodes to form a subgraph. For subsequent training, a further processing of the subgraphs was required. There were five one-way edges representing argument relations and one two-way edge representing concurrent relations in the event sub-graph. To maintain the flow of argument information in the graph, we added a reverse argument edge to each one-way argument edge in the graph. Finally, after the algorithm was finished, we masked the follow and causal edges in the subgraph. We fed the subgraph into the event scenario representation layer. The event classes of the tail events associated with the masked edges were sent to the outcome prediction layer as ground truth.

Algorithm 1 Event subgraph extraction.

Require: EKG
Ensure: EKG_Part
Initialize: $E_{seed}; R_{eg}; E_{eg}; E_{node};;$
while not end of E_{seed} **do**
  E_{temp}, R_{temp} = **Get_neighbour**(E_{seed});
  R_{eg}.**append**(R_{temp});
  E_{eg}.**append**(E_{temp});
  E_{seed}.**next**();
end while;
while not end of R_{eg} **do**
  if R_{eg} is not 'follow' or 'causal' **then**
    EKG_Part = **Append**$\langle E_{seed}, R_{eg}, E_{eg}\rangle$;
  end if;
  R_{eg}.**next**();
end while;
E_{node} = **Append** E_{seed} & E_{eg};
while not end of E_{node} **do**
  elm, type = **Get_argument**(E_{node});
  EKG_Part = **Append**$\langle E_{node}, type, elm\rangle$;
  E_{node}.**next**();
end while;
return EKG_Part;

4.3. Event Scenario Representation

The event graph also needed to be preprocessed before learning the event scenario representation. We embedded each event argument into a vector using a pretrained XLNet, and the "event" node was initially null. When initializing the "event" nodes, any vector of the same dimension as the event argument could be used, as long as each "event" was the same to avoid unnecessary interference. After preprocessing, the event subgraph and embedding results were fed into the scenario learning model.

The scenario learning model referred to the HGT approach. In the event graph, for a given arbitrary node, its neighboring nodes might all belong to different distributions, and we calculated the mutual attention of neighboring nodes based on different edges. By giving different weights to argument relations through the mutual attention, the information of each argument of the event was finally obtained from the event representation after several times of weighted message passing and message aggregation in the graph neural network. Figure 5 depicts the internal structure of the scenario learning model.

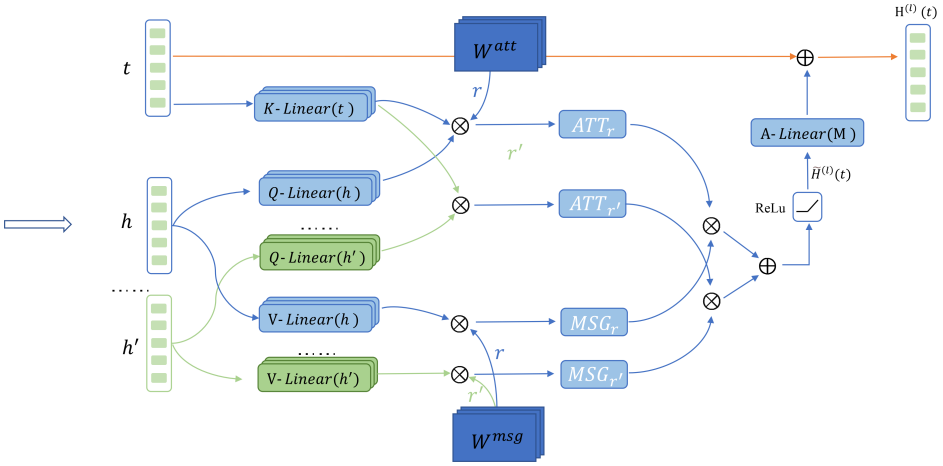

Figure 5. Event characterization layer.

Message passing and aggregation are important steps in graph neural networks. In an HGT, message passing needs to be accompanied by a calculation of the attention of each relational edge as passing weights. For different edges $\varphi_r = (h, t)$, the attention is obtained by multihead attention splicing, which is calculated as follows:

$$Attention(h, r, t) = Softmax(\Big\|_{i\in[1,h]} ATT - head^i(h, r, t)) \tag{8}$$

Each attention head is calculated as follows:

$$ATT - head^i(h, r, t) = (K^i W^r_{att} Q^i(t)^T) \cdot \frac{1}{\sqrt{d}} \tag{9}$$

The different types of head and tail nodes are mapped into Key vectors and Query vectors by different linear mapping functions K-Linear(h) and Q-Linear(t), which can maximize the simulation of the distribution differences of different nodes. Furthermore, unlike the common attention that directly computes the dot product of Key and Query, a heterogeneous mutual attention builds different matrices W^r_{att} for each relational edge, so that the different semantic associations between node pairs due to different relations can be captured. In the information transfer phase from the head node to the tail node, this step is parallel to the computation of mutual attention and has the main role of transferring information about the arguments and neighboring events to the current event node. The transfer of information from the head node to the tail node through different relational edges can be computed as:

$$Message(h, r, t) =\Big\|_{i\in[1,h]} MSG - head^i(h, r, t) \tag{10}$$

$$ATT - head^i(h, r, t) = Linear^i(H^{(l-1)}[s])W^{msg}_r \tag{11}$$

The multiheaded message is similar to the multiheaded attention calculation method, where the incoming upper-level message is projected through different linear layers to get different message headers, and multiple message headers are stitched together to get *Message(h, r, t)* passed from each head node to the tail node.

Finally, to aggregate messages, graph neural networks are generally aggregated by Add, Mean and Max, and we used Add in our approach. For the messages from the head nodes, they were averaged using the attention vector as the weight, and then the messages from each head node were summed to obtain the updated vector.

$$\tilde{H}^{[t]} = \oplus(Attention(h, r, t) \cdot Message(h, r, t)) \tag{12}$$

Finally, after the activation with ReLu, we used a linear layer after $\tilde{H}^{[t]}$ mapping back to the embedding dimension to fuse the neighbor information into the original nodes. The model took one-hop nodes at a time, and since the initial event nodes did not contain information, the event nodes aggregated information about the arguments after one representation layer, and at least one additional representation layer was needed to obtain information about concurrent events. In fact, we used a total of four characterization layers to obtain our desired scenario representation.

4.4. Model Training and Prediction Results

We formulated the training task as a multilabel classification task, with the event class to which all tail events belonged as the true labels G for comparison with our prediction results P. The model was optimized using MultiLabelSoftMarginLoss as the loss function to reduce the prediction of mislabeling, and the loss is calculated as follows:

$$loss(P, G) = -\sum_i G[i] * log(1 + exp(-P[i])^{-1}) + (1 - G[i]) * log(\frac{exp(-P[i])}{1 + exp(-P[i])}) \tag{13}$$

The prediction results were obtained as shown in Equation (2), using sigmoid and log_softmax functions to resolve the probability of the event occurrence. To enhance the prediction ability, the prior value λ was added to adjust the prediction results in the experiment. Using $sig()$ to denote the $Sigmoid()$ function and $l\_s()$ to denote the $log_softmax()$ function, the final prediction formula is shown below:

$$P(H(t)) = \begin{cases} 1, & \text{if } sig(l\_s(Linear(H(t)))) \geq \lambda; \\ 0, & \text{if } sig(l\_s(Linear(H(t)))) < \lambda; \end{cases} \quad (14)$$

where $Linear(H[t]) : R^d \rightarrow R^n$ and n is the prediction category.

5. Results

In this section, we used a real dataset to evaluate our model. We used different graph neural networks as representation layers and different language models as embedding modules for comparison and compared the impact of different event graphs on the prediction results.

5.1. Dataset and Experimental Setup

Dataset: Our dataset came from hundreds of related news articles crawled by the China News Network, and we filtered more than 100 high-quality data for chapter-level event and event relation labeling. Finally, we obtained a dataset containing about 510 events and 500 event relation pairs in 35 event classes. In order to ensure the data quality and retain as many events as possible, we removed the event classes containing too few event instances and divided the training and testing sets in a 7:3 ratio according to the event classes to which the event instances belonged. We tried to keep as many event classes as possible while keeping the results valid. After some trials, we removed the event classes with less than eight event instances. The final dataset contained a total of 510 event instances in 15 event classes. Our final prediction results contained a total of 13 event classes. The final event classes and the number of event instances they contained are shown in Table 2. The delineated dataset was used to generate subgraphs for training and testing according to the method given in Section 4.2. As we mentioned earlier, when generating the subgraphs, a central node with "event" as the default text was created for each event instance.

Table 2. Event classes and data volume.

Event Class	Event Instance Number
Death	101
CarCrash	68
TreatTheInjured	57
AccidentInvestigation	50
SeriousInjury	43
MinorInjury	35
Sideslip	28
PersonnelRescue	27
Rollover	21
Arrest	17
ModerateInjury	15
Speeding	14
PeopleCrash	13
Tailgate	13
FatigueDriving	8

Experimental setup: We use Adam [31] as the optimizer, all models had a learning rate of 0.0006, the number of multiheaded attention layers were all set to four, the hidden layer dimension was set to 400, and the input–output dimension was 768. We set the number of

training epochs to 100. To measure our model performance, we used accurate matching as a measure of the model's predictive power. Accurate matching is the ratio of the number of samples whose prediction result is exactly the same as the ground truth to the total number of samples. For the data that were not perfectly matched, we calculated precision, recall, and F1 values following the method of Godbole et al. [32]. Precision can be viewed as the probability of each event occurring in the prediction results given by the model, and recall represents how many events the model successfully predicted out of all upcoming events. Precision, recall, and F1 values were calculated as follows.

Let T be the true label set and S be the predicted label set. The standard IR measures of precision(P), recall(R) and F1 are defined in the multilabeled classification setting as follows

$$P = |T \cap S| / |S| \tag{15}$$

$$R = |T \cap S| / |T| \tag{16}$$

$$F1 = 2PR/(P + R) \tag{17}$$

5.2. Experiments

We experimentally validated the effectiveness of the HGEP model. In addition to the HGEP model, we also implemented models using a graph attention neural network [33] and graph convolution neural network [34] to obtain event feature representations as a baseline, hereafter referred to as GAT and GCN. We also used RoBERTa [35] and BERT [36] as embedding layers, respectively, to compare with the HGEP model using XLNet. We used pretrained basic versions of BERT, RoBERTa, and XLNet, which were retrained by the HFL lab based on a large Chinese corpus. People can get these open-source models at github.com/ymcui.

The difference between graph attention networks and graph convolutional networks is how they aggregate information from one hop: graph convolutional operations produce normalized sums of neighboring node features and graph neural networks introduce an attention mechanism instead of static normalized convolutional operations. Both are similar in that they are designed for isomorphic graphs and only consider different types of nodes, not different types of edges. In this paper, we extended the graph attention to a heterogeneous graph attention by assigning different weights to each type of edge in the graph and applying this weight to the message delivery and aggregation.

We give the experimental results in Table 3, from which we can see that the model in this paper outperformed the prediction models using GCN and GAT in terms of accurate matching values and accuracy. The reason why the model in this paper was lower than GCN and GAT in the recall value, we believe is due to the fact that the model became more conservative in making predictions with higher accurate matching value, which resulted in fewer correct and incorrect predictions. For this reason, we adjusted the prior value λ to change the accurate matching value and observed the changes in other evaluation metrics, as shown in Figure 6. Initially λ was one while the accuracy was only around 0.15, the false positive rate was extremely high and the model made a large number of predictions to cover the true labels. The recall value kept decreasing as λ increased. The precision value kept increasing when λ was lower than 0.58. The model predicted some of the positive cases as negative cases while correcting some of the false positive cases, and repredicted these false positive cases as negative cases. This means that for some uncertain events, the model preferred to predict that they would not happen.

Table 3. Benchmarks and ablation experiments.

Model	Accurate Matching	Precision	Recall	F1-Score
GCN [34]	34.72	63.95	39.86	49.11
GAT [33]	31.94	57.00	41.30	47.90
HGAEP	36.11	64.56	36.96	47.00
HGEP_BERT	30.56	58.82	36.23	44.84
HGEP_RoBERTa	33.33	61.73	36.23	45.66
w/o relation	36.11	64.56	36.96	47.00
w/o argument	34.72	64.00	34.78	45.07

As far as the experimental results are concerned, GCN outperformed GAT, while the model in this paper had a significant improvement over GAT and outperformed GCN, reflecting the role of the heterogeneous attention. This indicates that different arguments and neighboring events have different effects on events, and fusing them with different weights can improve the machine's ability to perceive situational information.

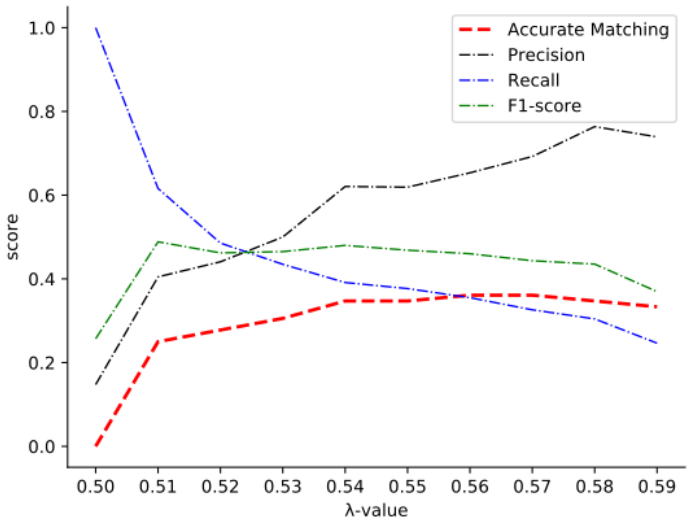

Figure 6. Variation of λ-indicator.

We also analyzed the effect of the size of the training data on the performance of the models when conducting the experiments of the embedding models. We randomly selected 50% and 70% of the training data, respectively, and conducted comparison experiments using each of the three embedding models. When using XLNet embedding, the model exact matching rate was reduced by about 1.4% and 2.8% when the data volume was reduced to 70% and 50%, respectively. XLNet outperformed Roberta and Bert by 1.5–3% and 4–7% for different data sizes. Furthermore, using 70% of the data volume, Bert overfitted our dataset, which resulted in a reduction in its performance. The results show that using XLNet had different degrees of improvement compared to using the other models. The experimental results are shown in Figure 7.

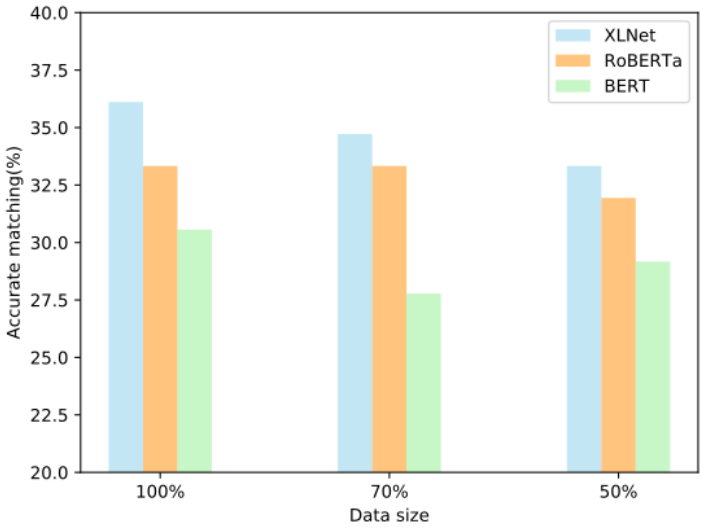

Figure 7. Data sampling rate.

5.3. Ablation Study

In this section, we set up ablation experiments to investigate the effect of different event representations on the prediction results. This included ablating relations and arguments from the event graph. The relation-level ablation experiments removed concurrent event relation edges from the graph and kept only the arguments. In the argument-level ablation experiments, the temporal and spatial parameters of events were removed, while other parameters and the graph structure were retained. The results of the ablation experiments are shown in Table 3.

It can be seen from the experimental results that the performance of the model decreased to some extent in different ablation experiments. Notably, there was a larger increase in recall values in the relation-level ablation experiments, which indicated that although the inclusion of concurrent events effectively improved the prediction accuracy, it was also an important factor in making the model predictions more conservative. In the argument-level ablation experiments, the decrease in metrics reflected the positive effect of temporal and spatial information on the event prediction task.

5.4. Case Study

In order to show the prediction results of our model more intuitively, we selected some representative cases. Our model had the ability to accurately predict multiple outcomes of events, and we analyzed the reasons why some events were not accurately predicted.

Case 1. 16 December 2019 afternoon A bus carrying 42 passengers from the capital Khartoum to Maidani was involved in a major traffic accident near the town of Abu Osher.

Case 2. Xinhua News Agency, Rio de Janeiro, September 20 (Reporter Zhao Yan Chen Weihua) Minas Gerais state fire department in southeastern Brazil said on September 20, the state early that morning, a small bus and truck collision.

Case 3. A collision between an off-road vehicle and a container truck occurred in the early morning of 11 in Agra district, Uttar Pradesh, India.

Our model in Case 1 predicted "fatalities" and "large injuries (serious injury and many injured)" based on the arguments: 16 December 2019 afternoon, Abu Osher town, bus, occurrence, traffic accident. The follow-up result "2 people died on the spot, 12 people were seriously injured, and other passengers were injured to varying degrees" also included "fatal" and "major injuries" events. Our forecast was consistent with this.

Case 2 follow-up events were "death", "minor injuries (minor injury and few injured)", and "accident investigation". Case 3 follow-up events were "death", "minor injury", and

"injury treatment". Our model predicted only "death" and "minor injury", which was only partially correct. We believe that this was due to the event relation, where the event outcomes "death" and "minor injuries" were causally related to the event and were directly caused by the event. The "treatment of the injured" and "accident investigation" as follow-up events followed the event, and although they were also event outcomes, they were generated by human decisions after the event. We believe that the lack of decision-related information was one of the reasons why it was difficult for the model to make accurate predictions for the subsequent events connected by the following relation.

6. Discussion

In this paper, we proposed an event prediction model based on event knowledge graphs and manually constructed an event knowledge graph in the transportation domain to verify the model's effectiveness. To better incorporate the information contained in multiple arguments and concurrent events, we explored the application of a graph attention neural network based on the model to verify the usefulness of a heterogeneous graph transformer for event scenario characterization. Furthermore, our experiments validated the importance of temporal and spatial factors in event representation.

In the future, we will explore more prediction methods and consider the construction of more complex event relations. The arguments of events have different entity types. Considering the role of different entity types on the model may help to further improve the prediction. In the future, we will consider incorporating multimodal information into the event knowledge graph to monitor sudden events in real time and provide rapid decision-making. To improve the decision-making capability of the model, we consider incorporating the Internet of things to obtain more information for decision-making.

Author Contributions: Conceptualization, W.L.; methodology, Y.L. and W.L.; validation, Y.L.; writing—original draft, Y.L.; writing—review and editing, W.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Natural Science Foundation of China Major Projects grant number 61991410, and Research on Marine Environment Vector Construction for Intelligent Evolution of Marine Unmanned System grant number P22KN00391.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available due to the need for subsequent scientific work.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Chambers, N.; Jurafsky, D. Unsupervised learning of narrative event chains. In *Proceedings of the ACL-08: HLT*; Association for Computational Linguistics: Columbus, OH, USA, 2008; pp. 789–797.
2. Granroth-Wilding, M.; Clark, S. What happens next? event prediction using a compositional neural network model. In Proceedings of the AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; Volume 30.
3. Zheng, J.; Cai, F.; Ling, Y.; Chen, H. Heterogeneous graph neural networks to predict what happen next. In Proceedings of the 28th International Conference on Computational Linguistics, Barcelona, Spain, 8–13 December 2020; pp. 328–338.
4. Li, Z.; Ding, X.; Liu, T. Constructing narrative event evolutionary graph for script event prediction. *arXiv* **2018**, arXiv:1805.05081.
5. Yang, S.; Wang, F.; Zha, D.; Xue, C.; Tang, Z. NarGNN: Narrative Graph Neural Networks for New Script Event Prediction Problem. In Proceedings of the 2020 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom), Exeter, UK, 17–19 December 2020; pp. 481–488.
6. Wang, Z.; Zhang, Y.; Chang, C.Y. Integrating order information and event relation for script event prediction. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Copenhagen, Denmark, 7–11 September 2017; pp. 57–67.
7. Liu, S.; Li, T.; Ding, H.; Tang, B.; Wang, X.; Chen, Q.; Yan, J.; Zhou, Y. A hybrid method of recurrent neural network and graph neural network for next-period prescription prediction. *Int. J. Mach. Learn. Cybern.* **2020**, *11*, 2849–2856.

8. Ramakrishnan, N.; Butler, P.; Muthiah, S.; Self, N.; Khandpur, R.; Saraf, P.; Wang, W.; Cadena, J.; Vullikanti, A.; Korkmaz, G.; et al. 'Beating the news' with EMBERS: Forecasting civil unrest using open source indicators. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data mining, New York, NY, USA, 24–27 August 2014; pp. 1799–1808.
9. Luo, W.; Zhang, H.; Yang, X.; Bo, L.; Yang, X.; Li, Z.; Qie, X.; Ye, J. Dynamic heterogeneous graph neural network for real-time event prediction. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, San Diego, CA, USA, 23–27 August 2020; pp. 3213–3223.
10. Song, X.; Wang, H.; Zeng, K.; Liu, Y.; Zhou, B. *KatGCN: Knowledge-Aware Attention Based Temporal Graph Convolutional Network for Multi-Event Prediction*; In Proceedings of the 33rd International Conference on Software Engineering & Knowledge Engineering, Pittsburgh, PA, USA, 1–10 July 2021.
11. Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; Yakhnenko, O. Translating embeddings for modeling multi-relational data. *Adv. Neural Inf. Process. Syst.* **2013**, *26*, 2787–2795.
12. Wang, Z.; Zhang, J.; Feng, J.; Chen, Z. Knowledge graph embedding by translating on hyperplanes. In Proceedings of the AAAI Conference on Artificial Intelligence, Quebec, QC, Canada, 27–31 July 2014; Volume 28.
13. Lin, Y.; Liu, Z.; Sun, M.; Liu, Y.; Zhu, X. Learning entity and relation embeddings for knowledge graph completion. In Proceedings of the AAAI Conference on Artificial Intelligence, Austin, TX, USA, 25–30 January 2015.
14. Ji, G.; He, S.; Xu, L.; Liu, K.; Zhao, J. Knowledge graph embedding via dynamic mapping matrix. 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), Beijing, China, 26–31 July 2015; pp. 687–696.
15. Tang, T.; Liu, W.; Li, W.; Wu, J.; Ren, H. Event Relation Reasoning Based on Event Knowledge Graph. In Proceedings of the 14th International Conference on Knowledge Science, Engineering and Management; Tokyo, Japan, 14–16 August 2021; pp. 491–503.
16. Hu, Z.; Dong, Y.; Wang, K.; Sun, Y. Heterogeneous graph transformer. In Proceedings of the 29th International World Wide Web Conference, Taipei, China, 20–24 April 2020; pp. 2704–2710.
17. Santiso, S.; Casillas, A.; Pérez, A.; Oronoz, M.; Gojenola, K. Adverse drug event prediction combining shallow analysis and machine learning. In Proceedings of the 5th International Workshop on Health Text Mining and Information Analysis (Louhi), Gothenburg, Sweden, 26–30 April 2014; pp. 85–89.
18. Laxman, S.; Sastry, P.; Unnikrishnan, K. Discovering frequent episodes and learning hidden markov models: A formal connection. *IEEE Trans. Knowl. Data Eng.* **2005**, *17*, 1505–1517.
19. Laxman, S.; Tankasali, V.; White, R.W. Stream prediction using a generative model based on frequent episodes in event sequences. In Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, NE, USA, 24–27 August 2008; pp. 453–461.
20. Zhou, C.; Cule, B.; Goethals, B. A pattern based predictor for event streams. *Expert Syst. Appl.* **2015**, *42*, 9294–9306.
21. Ballings, M.; Van den Poel, D. Customer event history for churn prediction: How long is long enough? *Expert Syst. Appl.* **2012**, *39*, 13517–13522.
22. De Caigny, A.; Coussement, K.; De Bock, K.W. A new hybrid classification algorithm for customer churn prediction based on logistic regression and decision trees. *Eur. J. Oper. Res.* **2018**, *269*, 760–772.
23. Korkmaz, G.; Cadena, J.; Kuhlman, C.J.; Marathe, A.; Vullikanti, A.; Ramakrishnan, N. Combining heterogeneous data sources for civil unrest forecasting. In Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015, Paris, France, 25–28 August 2015; pp. 258–265.
24. Radinsky, K.; Davidovich, S.; Markovitch, S. Learning causality for news events prediction. In Proceedings of the 21st international Conference on World Wide Web, Lyon, France, 16–20 April 2012; pp. 909–918.
25. Pichotta, K.; Mooney, R. Learning statistical scripts with LSTM recurrent neural networks. In Proceedings of the AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; Volume 30.
26. Damaschke, M.; Cronin, S.J.; Bebbington, M.S. A volcanic event forecasting model for multiple tephra records, demonstrated on Mt. Taranaki, New Zealand. *Bull. Volcanol.* **2018**, *80*, 1–14.
27. Scarselli, F.; Gori, M.; Tsoi, A.C.; Hagenbuchner, M.; Monfardini, G. The graph neural network model. *IEEE Trans. Neural Netw.* **2008**, *20*, 61–80.
28. Li, M.; Li, S.; Wang, Z.; Huang, L.; Cho, K.; Ji, H.; Han, J.; Voss, C. The Future is not One-dimensional: Complex Event Schema Induction by Graph Modeling for Event Prediction. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Punta Cana, Dominican Republic, 7–11 November 2021; pp. 5203–5215.
29. Hu, W.; Yang, Y.; Cheng, Z.; Yang, C.; Ren, X. Time-series event prediction with evolutionary state graph. In Proceedings of 14th ACM International Conference on Web Search and Data Mining, Jerusalem, Israel, 8–12 March 2021; pp. 580–588.
30. Liu, Z.; Cai, H.; Yu, H.; Shen, B.; Jiang, L. Constructing the Sequential Event Graph for Event Prediction towards Cyber-Physical Systems. In Proceedings of the 2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD), Dalian, China, 5–7 May 2021; pp. 1292–1297.
31. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
32. Godbole, S.; Sarawagi, S. Discriminative methods for multi-labeled classification. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 22–30.
33. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; Bengio, Y. Graph attention networks. *arXiv* **2017**, arXiv:1710.10903.
34. Kipf, T.N.; Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv* **2016**, arXiv:1609.02907.

35. Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv* **2019**, arXiv:1907.11692.
36. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv* **2018**, arXiv:1810.04805.