Dynamic Knowledge Graph based Multi-Event Forecasting

Songgaojun Deng  
Stevens Institute of Technology  
Hoboken, New Jersey  
sdeng4@stevens.edu

Huzefa Rangwala  
George Mason University  
Fairfax, Virginia  
rangwala@cs.gmu.edu

Yue Ning  
Stevens Institute of Technology  
Hoboken, New Jersey  
yue.ning@stevens.edu

ABSTRACT

Modeling concurrent events of multiple types and their involved actors from open-source social sensors is an important task for many domains such as health care, disaster relief, and financial analysis. Forecasting events in the future can help human analysts better understand global social dynamics and make quick and accurate decisions. Anticipating participants or actors who may be involved in these activities can also help stakeholders to better respond to unexpected events. However, achieving these goals is challenging due to several factors: (i) it is hard to filter relevant information from large-scale input, (ii) the input data is usually high dimensional, unstructured, and Non-IID (Non-independent and identically distributed) and (iii) associated text features are dynamic and vary over time. Recently, graph neural networks have demonstrated strengths in learning complex and relational data. In this paper, we study a temporal graph learning method with heterogeneous data fusion for predicting concurrent events of multiple types and inferring multiple candidate actors simultaneously. In order to capture temporal information from historical data, we propose Glean, a graph learning framework based on event knowledge graphs to incorporate both relational and word contexts. We present a context-aware embedding fusion module to enrich hidden features for event actors. We conducted extensive experiments on multiple real-world datasets and show that the proposed method is competitive against various state-of-the-art methods for social event prediction and also provides much-needed interpretation capabilities.

CCS CONCEPTS

- Information systems → Data mining; Computing methodologies → Knowledge representation and reasoning; Temporal reasoning.

KEYWORDS

Multi-Event Forecasting; Knowledge Graphs; Word Graphs

ACM Reference Format:


Figure 1: A motivating example of event graphs for multi-event and multi-actor forecasting. Nodes denote entities and edges denote event types and timestamps.

1 INTRODUCTION

Events such as organized crime, civil unrest movements, and disease outbreaks have a major impact on society. Previous event forecasting approaches mainly focus on predicting the occurrence of a given event type, e.g., protest [8, 24] or event subtypes, e.g. air pollution of CO vs. PM2.5 [11]. These methods are unable to identify concurrent events of multiple types. Meanwhile, existing methods neglect to predict the potential actors in an event, which is crucial for decision making. As shown in Figure 1, the entities/actors involved in social events are usually those who initiated an event or were targeted in an event (e.g. “farm worker” plays multiple roles in two events). Multi-event forecasting involves inferring multiple concurrent events of different types and participants in the future using historical input data.

Currently, events are often extracted from formal reports or news articles and structured as temporal knowledge graphs with additional textual features. An event graph, as shown in Figure 1, consists of entities (as nodes) and event types (as edges), where each edge has a timestamp indicating when the event occurred. Each event has a synopsis describing the content of the event. For instance, an event that occurred on Feb. 26, 2015, was Citizen Criticizes Government. The summary of the event is “A Politician attacked the state government on various fronts such as fertilizer crunch and land acquisition act”. These previous events among farm workers, the government, the agriculture ministry, and citizens provide indicators for the prediction of “protest” events and “farm worker” participants in the future.

Identifying related actors and past events in the prediction models provides backgrounds and clues for understanding the development of events and their hidden factors. Prior work [8, 24, 35] in this area has sought to identify supporting evidence in the form of precursor documents or context graphs while forecasting future events. However, the identified evidence in the existing work is often difficult to understand or requires further manual inspection.
when predicting multiple concurrent events of different types. Recognizing the correct set of actors and clues for each event type is difficult given the complex connections under past events. The task of multi-event and multi-actor forecasting presents several challenges:

- **Structured and unstructured features** Event data is usually a mixture of structured records (time, actors, types, etc.) and unstructured textual information (e.g., the event summary shown on an edge in Figure 1). Utilizing both forms of data for event forecasting is important given that relational records provide key elements of events while textual features provide enriched background information. However, few studies have performed the fusion of heterogeneous data for concurrent event forecasting.

- **Beyond link prediction**: Knowledge graph completion models relational information by predicting links between entities. However, it is difficult in practice to apply this line of research to predict both relations (event types) and entities (actors). Most knowledge graph completion methods only model the inherent structure of relational data and fail to exploit global historical data for future event predictions.

- **Beyond event forecasting**: Prior research on event modeling mainly focuses on forecasting event occurrences or counts in the future using either pre-defined features [39] or pre-trained embeddings [24]. Graph features constructed from text data have proven to be advantageous in deep learning methods for event prediction [8]. However, it is difficult to automatically extract event actors from graph-based text features.

We address the above challenges by considering two interconnected sub-problems: forecasting multiple co-occurring events of different types and inferring multiple actors in each event type. We propose a new methodology to fuse relational (event graphs) and semantic (text) information from heterogeneous historical data to predict concurrent events of multiple types and multiple actors in the future. Given this enriched information, we also provide interpretations for the prediction results. The model is able to automatically extract event actors from graph-based text features.

We evaluate the proposed method with other state-of-the-art models on real-world datasets with large-scale entity sets and event type sets. With quantitative and qualitative experiments, we demonstrate the strengths of the proposed method in multi-type and multi-actor event forecasting.

2 RELATED WORK

Our study is closely related to a large body of literature on event forecasting and knowledge graph completion.

**Event Forecasting.** There has been extensive research on event forecasting with spatio-temporal correlations, including many real-world applications such as stock market movements [2], disease outbreaks [29], and criminal activities [34]. Most existing machine learning methods only work on euclidean or grid-like data. For instance, linear regression models use tweet frequency (and quantity) to predict the occurrence time of future events [2]. More sophisticated features (e.g., topic-related keywords [34]) and models (e.g. multi-task learning[25, 39]) have also been investigated. Predictive methods have begun to confront the complex structure of social events and their underlying relations by considering the temporal evolution of event-related indicators or utilizing both temporal and geographical graph information [25]. Recently, Graph Convolutional Networks (GCN) have been proposed to address non-euclidean data in many domains such as social networks[26], natural language processing [20] and bio-medicine [10]. A dynamic graph convolutional network [8] was proposed to encode temporal text features into graphs for forecasting societal events and identifying their context graphs.

Most of these aforementioned models focus on the prediction of a single type of events or do not recognize the sub-types of events. Thus they are not flexible enough given that different models are trained for different types of events, which is computationally expensive and cannot guarantee good performance.

Event subtypes have also been studied. A multi-task multi-class deep learning model has been proposed for predicting the future occurrence of subtype events [11]. Different types of events occur simultaneously and it is necessary to model concurrent events.

However, many of the existing approaches have neglected hidden relational knowledge among entities. In this case, they cannot predict the actors of events who have significant impacts on social movements. Our proposed approach aims to predict multiple events where each event includes elements such as actors and event type. Events are augmented with textual descriptive summaries. This gives us the benefit of discovering the impact of hidden connections among entities for predicting future events and actors/entities.

**Knowledge Graph Completion** Although knowledge Graphs (KGs) have been recognized in many domains, most KGs are far from complete and are growing rapidly. Thus, KG completion (or link prediction) has been proposed to improve KGs by filling the missing connections.

Extensive studies have been done on modeling static, multi-relation graph data for link prediction. These methods mainly embed entities and relations into low-dimensional spaces [3, 9, 32, 36]. Among these methods, Relational Graph Convolutional Networks (RGCN) [28] is developed to generalize the GCN model [16] by dealing with direct, multi-relational graphs such as knowledge graphs.
Recently, CompGCN [33] has been proposed to jointly embed both nodes and relations in a relational graph. These methods achieve high accuracy on modeling static knowledge graphs.

There are recent attempts to incorporate temporal information in modeling dynamic knowledge graphs where facts may evolve over time. Know-Evolve [31] models the occurrence of a fact (edge) as a temporal point process. Embedding-based methods have been proposed to model temporal information, such as relation embeddings [12], time embedding [17], and temporal hyperplane [7]. These methods cannot generalize to unseen timestamps. Beyond link prediction, Jin et al. studied an autoregressive architecture for modeling temporal sequences of multi-relational graphs [14]. We employ temporal knowledge graphs in multi-type event forecasting. The difference between our work and temporal KG completion is that we predict events (and then actors) at future times. We formulate the problem into a multi-label classification problem. We consider both unstructured text data (event summaries) and graph data (event graphs) with temporal and relational information.

3 PROBLEM FORMULATION

The objective of this study is to forecast multiple co-occurring future events along with their event types (e.g., consult, provide aid, etc) and identification of involved key actors (e.g., politicians, organizations, etc). We formulate the knowledge-graph-based event forecasting task as two multi-label classification problems. We first introduce the formulation of the problems and then describe the necessary definitions used in our proposed method. The important mathematical notations are in Table 1.

Problem 1. Multi-Event Forecasting. We model the probabilities of a set of event types occurring at a future timestamp \(t\) based on historical input data \(X_{1:t-1}\):

\[
P(y_t|X_{1:t-1}).
\]

Here \(y_t \in \mathbb{R}^{|R|}\) is a vector of event types and \(X_{1:t-1}\) can be encoded from a set of graphs, documents, or word frequencies.

Problem 2. Multi-ACTOR Forecasting. In this problem, given the relevant historical data of an event, we model the probabilities of a set of actors (subject or object entities) being involved in a given type of event \(y_t\):

\[
P(a_t|y_t,X_{1:t-1}),
\]

where \(a_t \in \mathbb{R}^{|C|}\) is a vector of entities. In this work, we consider both relational information and textual event summaries as enhanced information in history to address these two problems. Next, we introduce the necessary definitions used in this paper.

Definition 3.1 Temporal Event Graph. A temporal event graph is a multi-relational, directed graph with time-stamped edges (event types) between nodes (entities). Let \(E\) be a finite set of event types and \(R\) be a finite set of event types. An event is defined as a time-stamped edge (i.e., subject entity \(s\), event type \(r\), object entity \(o\) and time \(t\)) succinctly represented by a quadruple \((s, r, o, t)\) or \((s_t, r_t, o_t)\), where \(s, o \in E\) and \(r \in R\). We denote a set of events at time \(t\) as \(G_t = \{(s, r, o, t)\}\). Temporal event graphs are built upon a sequence of event sets in ascending time order \(\{G_1, ..., G_T\}\), where each time-stamped edge has a direction pointing from the subject entity to the object entity. The same triple \((s, r, o)\) may occur multiple times in different timestamps, yielding different event quadruples. Each unique entity \(u\) (s or o) and unique event type \(r\) will be initialized with an embedding vector as \(h_u \in \mathbb{R}^d, o_r \in \mathbb{R}^d\).

Definition 3.2 Temporal Word Graph. Each event is attached with a paragraph of text or summary. We collect all the event summaries at each timestamp. Formally, we denote a set of summaries at time \(t\) as \(D_t = \{\sigma_t\}\). For each timestamp \(t\), we construct an undirected word graph \(D_t\) where each node denotes a unique word from \(D_t\). The edges are formed based on word co-occurrence in the summary set \(D_t\). Specifically, we employ point-wise mutual information (PMI) [6] to measure the semantic relevance of two words, that is, the edge weight. In the document set \(D_t\), we define an edge between word \(\omega\) and word \(\phi\) if \(\text{PMI}(\omega, \phi) > 0\), noted as \((\omega, \phi) \in D_t\), that is \(D_t = \{(\omega, \phi)\}\). \(V_t\) contains all words at time \(t\) and \(V\) is the vocabulary set. Each word \(\omega\) is initialized with a pre-trained embedding vector \(b_\omega \in \mathbb{R}^d\). Temporal word graphs are built from a series of summary sets, sorted in ascending time order.

In the following section, we use ← to over a letter to represent relational embeddings in event graphs, − to denote semantic embeddings in word graphs, and ♠ to denote fused embeddings.

4 METHODOLOGY

Figure 2 provides an overview of the multi-event forecasting framework that consists of three parts: (1) graph aggregation module to obtain the relational and semantic embeddings from event and word graphs at each historical timestamp, respectively; (2) context-aware embedding fusion module to enhance representations of entities and event types by blending contextual features from words; and (3) recurrent encoder module to learn temporal global embeddings across all historical times. For the multi-actor forecasting problem, we model node-level and edge-level representations instead of global graph features. We propose a temporal inference module to model dynamic temporal features. An additional intrinsic inference module is designed to model the inherent association of entities and event types. The pseudocode of multi-event forecasting and multi-actor forecasting is presented in the supplemental material.

4.1 Multi-Event Forecasting

Assuming that the occurrences of events at time \(t\) depend on the historical events that happened in the previous time window of \(m\) time steps, we model the probabilities of events of multiple types \(P(y_t)\) at time \(t\) using the temporal graphs of events \(\{G_{t-m:t-1}\}\) and graph data (event graphs) with temporal and relational information.

<table>
<thead>
<tr>
<th>Notations</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>(E, R, V)</td>
<td>entity, event type, and vocabulary sets</td>
</tr>
<tr>
<td>(G_t, D_t)</td>
<td>event and word graphs at time (t)</td>
</tr>
<tr>
<td>(G_r, D_r)</td>
<td>event and word subgraphs of event type (r) at time (t)</td>
</tr>
<tr>
<td>(h_u, o_r)</td>
<td>pre-trained word/vec embedding of word (\omega)</td>
</tr>
<tr>
<td>(H_t)</td>
<td>semantic embedding matrix of words at (t)</td>
</tr>
<tr>
<td>(H_t, O_t)</td>
<td>relational embedding matrices of entities at (t)</td>
</tr>
<tr>
<td>(t)</td>
<td>feature dimension</td>
</tr>
</tbody>
</table>

Table 1: Important notations and descriptions
The edge weight computed from PMI is based on the co-occurrence frequency of these two words. At each time step \( t \), we obtain the relational embedding matrices of all entities \( \mathcal{E}_t \) and event types \( \mathcal{R}_t \) in \( \mathcal{G}_t \) at time \( t \). \( \mathcal{E}_t \subseteq \mathcal{E} \) is a set of entities that only appear in \( \mathcal{G}_t \) and likewise for \( \mathcal{R}_t \subseteq \mathcal{R} \).

Semantic Embedding (Word Graph). In word graphs, edges represent the semantic relevance of two connected words. The edge weight computed from PMI is based on the co-occurrence frequency of these two words. At each time \( t \), each word in the word graph \( \mathcal{D}_t \) learns its semantic features through all its undirected neighbors. For a single word \( \omega \) in \( \mathcal{D}_t \), we apply a multi-layer GCN model [16] to update its embedding vector:

\[
\mathbf{h}_\omega^{(l+1)} = f \left( \sum_{(\omega, \varphi) \in \mathcal{D}_t} \mathbf{W}_\varphi^{(l)} \mathbf{h}_\varphi^{(l)} \right),
\]

where \( \mathbf{h}_\varphi^{(0)} = \text{word2vec} \) vector of \( \varphi \) at the first layer. \( \mathbf{W}_\varphi^{(l)} \) is the weight matrix. Semantic embedding matrix \( \mathbf{H}_t \in \mathbb{R}^{|\mathcal{V}| \times d} \) of all words in \( \mathcal{D}_t \) is obtained at time \( t \).

Context-aware Embedding Fusion. After graph aggregation, entities and event types are encoded with relational structure information. Yet, entities, event types, and words are usually closely related. For instance, in the introduction example, \( \text{Citizen, Criticizes, Government, 02/26/15} \) with description “A Politician attacked the state government on various fronts such as fertilizer crunch and land acquisition act.”. The embedding of the entity \( \text{Government} \) after aggregation only involves the relational information, e.g., involved
We introduce a novel context-aware embedding fusion module to enhance the information of events and entities from words.

For each entity or event type, we identify the related words in temporal word graphs. For instance, the entity member of the judiciary (india) with related words {india, member, judiciary} and the event type Arrest, detain, or charge with legal action with words {arrest, detain, charge, legal action}.

A simple and straightforward method is to combine the embedding of all related words by concatenation or linear transformation. Treating all words equally often results in noise. In the above example, india and judiciary contribute more semantic meanings to the entity phrase, and we usually care more about the context of these words. To evaluate the importance of related words, we propose a context-aware attention method to implicitly integrate more useful words.

Formally, given an entity $i$ (query), we compute the attention score for each related word (context) $\omega \in W_i$, where $W_i$ is the set of words in $D_{t-m_{t-1}}$ that are semantically related to $i$:

$$a_{t,(i,\omega)} = \frac{\exp(\text{Attn}(\tilde{h}_{t,(i)}, \tilde{g}_{\omega}))}{\sum_{\varphi \in W_i} \exp(\text{Attn}(\tilde{h}_{t,(i)}, \tilde{g}_{\varphi}))} \in \mathbb{R},$$

where $\text{Attn}()$ denotes the attention mechanism (e.g., general attention [18]), $\tilde{h}_{t,(i),\omega}$ is the relational embedding of entity $i$ in $G_t$, and $\tilde{g}_{\omega}$ represents the semantic embedding of word $\omega$ in $D_{t-m_{t-1}}$ that related to entity $i$. The attention scores are used to compute a linear combination of the semantic embeddings. We then concatenate the relational embedding of the entity with the context-aware semantic features, and obtain the fusion feature of the entity through a non-linear transformation:

$$h_{t,(i)}^{*} = \tanh(W^{T}_a \cdot \frac{\sum_{\omega \in W_i} a_{t,(i,\omega)} \tilde{g}_{\omega}}{\text{rel}}} \in \mathbb{R}^d,$$

where $W_a \in \mathbb{R}^{d \times d}$ is a weight matrix, and $\cdot$ is concatenation. If a candidate word appears in multiple word graphs, we consider its semantic embedding in the latest graph, which involves the latest context. For instance, if the word worker appeared in both $D_{t-2}$ and $D_{t-1}$, we use the semantic embedding of the word in $D_{t-1}$. If there are no related words for an entity, we use zero vectors for semantic information. For event type $i$, we replace $\tilde{h}_{t,(i)}$ with $\tilde{g}_{t,(i)}$ in Eq. 7-8 to get its fused embedding $\tilde{g}_{t,(i)}$.

**Recurrent Encoder**

Given a sequence of fused embeddings of entities and event types, as well as the word embeddings, i.e., $(\tilde{H}^{*}_{t-m_{t-1}}, \tilde{O}^{*}_{t-m_{t-1}}, \tilde{P}_{t-m_{t-1}})$, we employ a recurrent neural network to model temporal information. To reduce the spatial size of the feature representation and obtain salient features, we apply a max pooling layer over the embedding of the entities, event types, and words, respectively. A recurrent neural network is then applied to update the global temporal embedding recurrently, to obtain the final latent embedding $g_t$:

$$g_t = \text{RNN}(\left[ p(\tilde{H}^{*}_{t-1}); p(\tilde{O}^{*}_{t-1}); p(\tilde{P}_{t-1}) \right], g_{t-1}) \in \mathbb{R}^d,$$

where $p(\cdot)$ indicates the max pooling operation applied element-wise over all nodes or edges. The latent embedding $g_t$ is then utilized in the final prediction in Eq. 3.

### 4.2 Multi-Actor Forecasting

We introduce a novel method to infer actors from both temporal historical data and the inherent correlation between entities and event types. Here, the context is constrained given that we only focus on entity interactions for a given event type. We first introduce a graph sampling method for a given event type and then our method for predicting multiple actors that may involve in an event.

**Graph Sampling**

We sample subgraphs from global temporal event graphs using edge sampling. Formally, given an event graph $G_t$ at time $t$, we first sample the quadruplets where the event type is $r$, denoted as $\text{SET}_r = \{(s, r, o)|r \in \text{SET}_r \}$.

The first level neighboring edge set is $\text{SET}_r^{nbr} = \{(s, r, o) \in \text{SET}_r\}$.

The total sampled edge set is $\text{SET}_r \cup \text{SET}_r^{nbr}$. We can continue to find incoming edges from the subject set to sample higher-level neighboring edges. At time $t$, the event subgraph of $r$ can be represented as $G_r^t$.

Accordingly, for word graphs, we consider only the text summaries of the events/edges sampled in the event subgraph. The temporal word subgraph of $r$ at time $t$ is denoted as $D_r^t$. The sampled graph provides a restricted context, focusing on one event type $r$, which ensures that most of the noise is eliminated.

Based on a predicted event type $y_t = r$ in multi-event prediction along with the sampled subgraphs, we model the multi-actor forecasting problem $\mathbb{P}(a_t | y_t = r)$ in the following way:

$$\mathbb{P}(a_t | y_t, G_r^{t}, D_{t-m_{t-1}}) = \sigma(\tilde{z} + z) \in \mathbb{R}^{|E|},$$

where $\tilde{z}$ is the temporal inference, which encodes temporal historical features for each actor. $z$ is the intrinsic inference over the inherent correlation between entities and the given event type.

We first introduce the intrinsic inference and then the temporal inference. We use $r$ to represent $y_t = r$ for simplicity.

**Intrinsic Inference**

We model the inherent correlation for entities and each given event type in a modified way as in RESCAL [23]. RESCAL is a method that uses a tensor factorization model to learn the inherent structure of relational data. Given an event, a shared bilinear layer for all entities is applied:

$$z = H \cdot W_{\beta} \cdot o_r \in \mathbb{R}^{|E|},$$

where $o_r \in \mathbb{R}^d$ is the learnable embedding vector specified for event type $r$. $H \in \mathbb{R}^{d \times d}$ is the embedding matrix for all entities in the predefined entity set $E$, where each row ($h_u$) represents the embedding of an entity (u). $W_{\beta} \in \mathbb{R}^{d \times d}$ is the weight matrix.

The intrinsic inference module especially helps when there is no historical data for some entities.

**Temporal Inference**

Actors change over time and event types are the glue that connects different entities. We model latent embeddings from these subgraphs in a similar way to the multi-event prediction problem. As the historical information of each entity
We model the latent embedding for each entity \( i \in \mathcal{X} \) with \( \mathbf{H}_i \in \mathbb{R}^{d} \), and the fused embedding vector of event type \( r \), i.e., \( \mathbf{o}_{t,r} \in \mathbb{R}^{d} \) according to Eq. 7-8. \( \mathcal{E}_r \subseteq \mathcal{E} \) is the entity set in \( \mathcal{G}_r \).

Then recurrent neural networks take the fused embedding as input and learn the hidden temporal features as:

\[
\tilde{\mathbf{h}}_{t,(i)} = \text{RNN}^2(\mathbf{h}_{t,(i)}, \tilde{\mathbf{h}}_{t-1,(i)}) \in \mathbb{R}^{d},
\]

\[
\mathbf{o}_{t,r} = \text{RNN}^3(\mathbf{o}_{t,(r)}, \tilde{\mathbf{o}}_{t-1,(r)}) \in \mathbb{R}^{d}.
\]

We model the latent embedding for each entity \( i \) using a shared RNN\(^2\) and use \( \mathbf{H}_i \) to denote the entity latent embedding matrix at time \( t \) for all entities. For a given event type \( r \) at time \( t \), the latent embedding learned from an RNN is denoted as \( \mathbf{o}_{t,(r)} \). We then model the temporal features at time \( t \) by using the latent embedding of the event type \( r \) and all entities at time \( t-1 \):

\[
\tilde{\mathbf{z}} = \mathbf{H}_{t-1} \cdot \tilde{\mathbf{W}}_{\beta} \cdot \mathbf{o}_{t-1,(r)} \in \mathbb{R}^{|\mathcal{E}|},
\]

where \( \tilde{\mathbf{W}}_{\beta} \in \mathbb{R}^{d \times d} \) is a learnable weight matrix. Both intrinsic embedding \( \tilde{\mathbf{z}} \) and temporal embedding \( \tilde{\mathbf{z}} \) are then integrated in the final prediction as shown in Eq. 10.

### 4.3 Learning and Inference

In our multi-label learning problems, we define an instance space \( \mathcal{X} \) and label space \( \mathcal{Y} = \{e_l\}^L \); \( e_l \) is a decimal in \([0,1]\) and \( \sum_l e_l = 1 \). Instead of using binary values, we consider the true label set as a distribution, which is calculated by the frequency of each label. \( L \in \mathbb{N}_+ \) represents the total number of labels. Given an instance \( x \in \mathcal{X} \) with label vector \( y \in \mathcal{Y} \), we interpret \( y_i > 0 \) to mean that the label \( i \) is “relevant” to the instance \( x \). In our problems, \( L \) is the size of relations \(|R|\) or entities \(|E|\) in problem 1 and 2, respectively.

It is challenging to predict the most relevant labels for a given input when the size of the output space is relatively large [27]. Especially, in the multi-actor forecasting task, the relevant label/entity sets are very small compared to the size of entity set, e.g., 4 out of 7k labels. In problem 1, we aim to detect as many labels as possible. Instead, for problem 2, where the true labels have high sparsity, our goal is to find a ranking over labels given an instance. We adopt the categorical cross-entropy loss [19, 21] which is defined as:

\[
- \frac{1}{L} \sum_{l \in L} y_i \log \left( \frac{\exp(y_i)}{\sum_{j \in L} \exp(y_j)} \right),
\]

where \( y_i \) is the model prediction for label \( i \) before the nonlinear function \( \sigma \) as in Eq. 3 and 10.

At inference time, we utilized a sigmoid function over the class score \( \tilde{y}_i \) and a threshold of 0.5 to determine the occurrence of an event in the multi-event prediction problem. The multi-actor forecasting problem is formulated similar to ranking problems. We aim to find a ranking over the entity labels, where a higher ranking indicates that the label has a higher probability of occurrence. Formally, given the event type, we compute the rank of each candidate entity, that is, the rank of class score \( \tilde{y}_i \) among \( \{\tilde{y}_i\}^L \).

| Dataset     | \(|E|\) | \(|R|\) | \(|V|\) | #events | \(|e|\) | \(|s/o|\) |
|-------------|--------|--------|--------|---------|-------|--------|
| India       | 6,298  | 234    | 23,373 | 479,649 | 19    | 7.0    |
| Russia      | 7,798  | 237    | 16,299 | 373,364 | 22    | 4.7    |
| Nigeria     | 3,896  | 221    | 13,933 | 237,096 | 21    | 4.0    |
| Afghanistan | 3,756  | 218    | 10,468 | 279,859 | 19    | 4.0    |
| Iran        | 7,547  | 229    | 13,808 | 328,266 | 22    | 4.4    |

### 4.4 Interpretability

Providing explanations for prediction results is helpful for human analysts and decision makers. Here, we introduce the interpretabilities of our approach from the perspectives of both event graphs and semantic information.

**Important historical events.** In multi-event prediction, we apply max pooling over entities, event types, and words respectively to extract salient features at each history timestamp. For instance, element-wise max pooling selects features in each dimension across all entities. We select entities that partially retain their features after maximum pooling. In our temporal event graphs, each event/edge is encoded with an event ID which allows us to track the unique and complete event quadruple \((s, r, o, t)\). We then take advantage of max pooling to select important entity, words, and event quadruples (by event type) at each history step. The importance scores are obtained based on the ratio of the dimensions whose features are selected through max pooling.

**Important semantic contexts.** In the context-aware embedding fusion module, we use Eq. 7 to model the attention score of a related word for a given entity/event type. The attention score can be further employed to quantify the importance of words in contributing the entity/event type embedding for prediction.

### 5 EXPERIMENT SETUP

#### 5.1 Datasets

The experimental evaluation was conducted on event datasets of five countries from Integrated Conflict Early Warning System (ICEWS) [4]. It contains political events designed to assess national and international crisis events. These events are coded using 20 main categories and their subcategories. Each event is encoded with geolocation, time (day, month, year), category, entity (subject, object) and its associated text, etc. In this paper, we focus on all categories of events and select country-level datasets from five countries, **India, Russia, Nigeria, Afghanistan** and **Iran**. The main event types include make public statement, appeal, express intent to cooperate, protest, etc. All data range from Jan. 1, 2010 to Feb. 22, 2016. Due to the lack of data from Feb. 22, 2012, to Jan. 1, 2013, we have a total of 1931 days of data. The time granularity is one day. Data statistics are shown in Table 2.

#### 5.2 Evaluation Metrics

To evaluate our method quantitatively, we use the following metrics:

- \( F \beta \) is the weighted harmonic mean of precision and recall, reaching its optimal value at 1 and its worst value at 0. \( \beta > 0 \) is the...
We compare our method with several state-of-the-art baselines. Considering the weighted averaging of relational information are also ignored as actor modeling relies on the F score, $\beta$ is set to 1 and 2. We evaluate the ability of a model finding all relevant cases. The higher weight of recall in the F score, $\beta$ is set to 1 and 2. We are interested in the evaluation of information retrieval tasks, $k$ is set to 1, 3, and 10 in our experiments.

We consider the weighted averaging of Fβ and Recall in multi-event forecasting. Hits@k is adopted in multi-actor forecasting.

### 5.3 Comparative Methods

We compare our method with several state-of-the-art baselines.

- **DNN**: A simple deep neural network consisting of three dense layers using the same loss as Eq. 15.
- **MLKNN** [37]: A multi-label lazy learning approach, which is derived from the traditional k-Nearest Neighbor (kNN) algorithm.
- **BRKNN** [30]: A binary relevance multi-label classifier based on k-Nearest Neighbors method.
- **MLRAM** [1]: A scalable extension to the fuzzy Adaptive Resonance Associative Map neural network.
- **DynGCN** [8]: A dynamic graph model with a temporal encoded feature module for binary event forecasting.
- **T-GCN** [38]: A spatiotemporal graph network that combines GCN and GRU for traffic prediction.
- **RENET** [14]: A neural architecture for modeling complex event sequences, which consists of a recurrent event encoder and a neighborhood aggregator.
- **tRGCN** [28]: A variant of RGCN for node classification, which models node-level temporal features after graph convolution of each historical step. GRU is applied to model temporal features.
- **tCompGCN** [33]: ComGCN is a graph convolutional framework which jointly embeds both nodes and relations in a graph. We adapt this model for temporal event graphs similar to tRGCN.

The first four models use basic count features, the next two (DynGCN, T-GCN) use word graphs, and the last three (RENET, tRGCN, tCompGCN) use event graphs. All methods except tRGCN and tCompGCN, are compared in multi-event forecasting. In actor prediction, we compare our method with DNN and baselines that use event graphs. Benchmarks that use count features are ignored given their poor performance. Methods using word graphs irrespective of the relational information are also ignored as actor modeling relies on relational data. To evaluate the importance of different components of our method, we varied our base model in different ways: Glean−fusion removes the embedding fusion step, and Glean−temp eliminates temporal inference in the multi-actor forecasting task. Hyperparameter setup is introduced in the supplemental material.

### 6 EXPERIMENTAL RESULTS

Our evaluation includes a detailed comparison of our proposed method with baselines described in Section 5.3 for multi-event and multi-actor forecasting tasks. We examine the sensitivity of different hyper-parameters (in Supplementary Material), and provide a case study to demonstrate the discovered historical event-actor graphs as interpretations for predicted events.

#### 6.1 Results of Multi-Event Forecasting

Table 3 presents comparison results of the multi-event forecasting tasks. Our method outperforms baselines across all datasets in terms of F-score and recall. We provide the relative performance gain of our method to the best baseline model. The improvement on the Indian dataset is the most significant. Models using the count feature (history event count) have the worst performance, showing that simple historical features are less effective effective in multi-event prediction. For models using word graphs, DynGCN performs poorly on all datasets, which indicates that the method fails on multi-label classification tasks. T-GCN utilizes word graphs and produces better performance than previous baselines, which indicates that the semantic information of word graphs can be helpful for multi-event prediction. For models that use event graphs, RENET$^1$ directly predicts the events, while RENET$^2$ follows the logic in the original paper by first predicting the subject and then events. RENET models achieve good performance in all datasets, which shows that the use of relational information in event graphs can improve prediction performance. Comparing variants of our method, we observe that the removal of the embedding fusion step results in performance degradation in four out of five datasets. The results suggest that embedding fusion plays a key role in performance enhancement.

#### 6.2 Results of Multi-Actor Forecasting

In multi-actor forecasting, we aim to predict potential actors related to a given event. As shown in Table 4, our method and its variants achieve better overall performance than the baselines across...
It proves the importance of considering relational information in predicting event participants. These models that consider only relational features perform worse than our approach. It shows the importance of enriching relational features and better design of feature learning. For the ablation study of our method, we can observe that the fused embedding unhelpful. It also demonstrates the importance of enriched relational features and better design of feature learning.

### 6.3 Case Study

In this section, we showcase the interpretability of our method based on multi-event and multi-actor prediction tasks.

Identifying important historical events. We select a test instance from the India dataset. In this example, we successfully predicted multiple events on Oct. 14, 2015 using the temporal event and word graphs from one past week. Given one predicted event type and sampled subgraphs, we then predicted the subjects that may be involved in this event. In Figure 3, the event graph on the left is the extracted important historical events (quadruples) with importance scores based on Section 4.4. We can observe Activist and Police had clashes on Oct. 7 and 9. Then on Oct. 11, a group of Citizens got involved in an event of Arrest, detain, or charge with legal repression with a relatively high weight of 0.08. On Oct. 12, there are events suggesting connections between Citizen and Activist. The ongoing conflict among Activist, Citizen and Police is observed. In the prediction, our model correctly predicted Protest violently and Used tactics of violent repression on the top right. The extracted history event graph provides strong evidence for the prediction.

For multi-actor prediction, the approach predicted the potential subjects that may be involved in the event, and the sampled subgraphs are served as the input feature. We show the actor prediction results on the bottom right, including the intrinsic and temporal scores and the entity ranking. The top four entities are all involved in the event. We can observe that Activist, Citizen, and event type Protest violently form two correct subject actor predictions, as shown by the dotted line in the upper right. The intrinsic and temporal scores are obtained by normalizing $z$ and $\tilde{z}$ respectively using softmax. Note that Activist and Citizen achieve high intrinsic scores in this instance. This is reasonable since they participated in previous events. The temporal scores of the correctly predicted entities achieve relatively high scores, which is mainly based on historical events. An entity Children shows very low intrinsic and temporal scores since it is unrelated to the event.
Identify Semantic Contexts and Feature Flows. For a given entity or relation, we selectively fuse the embeddings of related words in the corresponding temporal word graph. The word embedding involves the context information after the graph aggregation. Attention scores are then calculated for each related word to select important features. Based on a test instance from the Russia dataset, we provide a feature flow diagram in Figure 4 to show how the embedding fusion works. For the entity Member of Parliament (Russia), we obtain the highest score 0.4 from parliament, and for relation Express intent to meet or negotiate, word negotiate has the highest attention. For those words whose features have not yet been fused, e.g., conflict, they also serve as an additional semantic indicator for event prediction. Note that the words are stemmed and related words are defined by whether they are included in the entity.

7 CONCLUSION AND FUTURE WORK

Predicting concurrent events of different types and inferring involved actors are important tasks for decision-makers and policy analysts. We present a novel dynamic knowledge graph based model with context-aware embedding fusion to handle heterogeneous graph data. We demonstrated the effective prediction performance as well as the interpretability of the proposed model on large-scale and real-world datasets. Future work will consider modeling unseen and rare event types or entities and dynamically expanding actor and event type sets when new data arrives. Another important direction is to model the hidden relationship between event subjects and objects to infer a complete fact.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers and NVIDIA Corporation with the donation of the Titan V GPU. This work is supported in part by the US National Science Foundation under grant IIS-1948432. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

Supplemental Material

A PSEUDO CODE

We present the training steps of the proposed models in the Algorithms 1 and 2.

Algorithm 1: Multi-event forecasting algorithm

| Input: | Temporal event graphs \(\{G_1, \ldots, G_T\}\), temporal word graphs \(\{D_1, \ldots, D_T\}\), pre-trained word embeddings, initial model parameters. |
| Output: | A trained multi-event classifier \(f : (G_t, m) \rightarrow \hat{\Psi}(\hat{\gamma})\) |

1 while stopping criterion not met do
2 Sample a minibatch of \(k\) examples
3 for \(r \leftarrow t - m \text{ to } t - 1\) do
4 \(\tilde{H}_r, \tilde{O}_r \leftarrow \text{aggregate event graph } G_r\)
5 \(\tilde{H}_r \leftarrow \text{aggregate word graph } D_r\)
6 \(\{H_{t-m+1}, O_{t-m+1}\} \leftarrow \text{apply embedding fusion}\)
7 \(g_{t-1} \leftarrow \text{apply recurrent encoder}\)
8 Compute \(\hat{\Psi}(\hat{\gamma})\) based on Eq. 3
9 Compute loss based on Eq. 15 and update \(\Theta\)

Algorithm 2: Multi-actor forecasting algorithm

| Input: | Temporal event subgraphs \(\{G'_1, \ldots, G'_T\}\), temporal word subgraphs \(\{D'_1, \ldots, D'_T\}\), pre-trained word embeddings, initial model parameters \(\Theta\). |
| Output: | A trained multi-event classifier \(f : (G'_{t-m+1}, D'_{t-m+1}) \rightarrow \hat{\Psi}(\hat{\gamma})\) |

1 while stopping criterion not met do
2 Sample a minibatch of \(k\) examples
3 \(z \leftarrow \text{compute intrinsic inference}\)
4 \(\bar{z} \leftarrow \text{compute temporal inference}\)
5 Compute \(\hat{\Psi}(\hat{\gamma})\) based on Eq. 10
6 Compute loss based on Eq. 15 and update \(\Theta\)

B PARAMETER SETTING

We pre-train a 100-dimensional word2vec embedding for each word in the vocabulary using all text data in each country. We split the data into training, validation, and test sets in chronological order at a ratio of 80%-10%-10% for each dataset. We preprocess text data and keep only the stemmed words for constructing the word graph. For hyper-parameter setting, the history step (day) \(m\) is set to 7. The feature dimension \(d\) is 100. The composition operator \(\phi\) in Eq. 4 is the element-wise multiplication. The activation function \(f\) in Eq. 4 and 6 is ReLU [22]. General attention is used in the context-aware embedding fusion module. For temporal event subgraphs, we sample the first-level neighboring edges for the given event type. We employ 1-layer GRU [5] as our recurrent encoder, where the size of hidden states is 100. For graph aggregators, we consider two hidden layers in both CompGCN and GCN. Unlike the link prediction task in knowledge graph, we ignore inversed edges and only consider the original edges. In actor prediction, the subject and object that interact with the event are considered as two samples, which are trained with different parameters in the output layer. Due to the huge amount of data, we randomly select 50 event types and predict the participants of each event type per day. All parameters, including the embedding of all entities and event types, are initialized with Glorot initialization [13] and trained using the Adam [15] optimizer with learning rate 1e-3, weight decay 1e-5, and dropout rate 0.5. The batch size varies for each dataset due to the different size of the input files. The best models are selected by early stopping when the validation hamming loss does not decrease for 3 consecutive epochs. Hamming loss computes the fraction of the wrong labels to the total number of labels. All experimental results are the average of 5 randomized trials. All code is implemented using Python 3.7.4 and Pytorch 1.0.1 with CUDA 9.2.

Experimental Settings for Baselines

For model tRGCN and tCompGCN, we use RGNC or ComGCN to aggregate relational features of entities in the event graph at each timestamp. Then we use a GRU to model node temporal features similar to Eq. 12. A shared linear layer is applied to the final hidden state of each entity for individual predictions. The loss function is the same as ours.

C RECURRENT ENCODER

We employ the RNN model as the recurrent encoder to model the sequential information of the relational and semantic information at each timestamp as in Eq. 9, 12 and 13. Specifically, we use Gated Recurrent Units [5]. For Eq. 9, given a single layer GRU, the detailed steps are defined as:

\[
x_t = [p(H_t); p(O_t); p(H_t)]
\]

\[
r_t = \sigma(W_r \cdot [g_{t-1}; x_t])
\]

\[
z_t = \sigma(W_z \cdot [g_{t-1}; x_t])
\]

\[
\tilde{h}_t = \text{tanh}(W \cdot [r_t \odot g_{t-1}; x_t])
\]

\[
g_t = (1 - z_t) \odot g_{t-1} + z_t \odot \tilde{h}_t,
\]

where \(x_t, g_t\) are the input and latent embedding (GRU hidden state) at time \(t\), \(r_t, z_t, \tilde{h}_t\) are the reset, update, and new gates, respectively. \(\sigma()\) is the sigmoid function and \(\odot\) is the Hadamard product.

D EVALUATION METRICS

We now describe the evaluation metrics used for assessing the quality of the models. We use weighted average metrics to evaluate multi-event prediction tasks. The weighted precision, recall, and Fβ are defined as follows:

\[
\text{Precision} = \frac{1}{L} \sum_{i \in L} w_i \cdot \frac{TP_i}{TP_i + FP_i},
\]

\[
\text{Recall} = \frac{1}{L} \sum_{i \in L} w_i \cdot \frac{TP_i}{TP_i + FN_i},
\]

\[
F_\beta = \frac{1}{L} \sum_{i \in L} w_i \cdot \frac{1 + \beta^2 \cdot \text{Precision}_i \cdot \text{Recall}_i}{\beta^2 \cdot \text{Precision}_i + \text{Recall}_i},
\]

where \(w_i = \frac{n_i}{N}\) is the sample weight for label \(i\), \(n_i\) is the number of samples of label \(i\) among all samples \(N\). \(TP_i, FP_i, FN_i\) are true
positive rate, false positive rate, and false negative rate for label i, respectively. Precision is provided just to better explain Fβ.

For measuring the quality of the ranking in the multi-actor forecasting task, we employ the Hits@k metrics. Since we predict a ranking for all candidate entities, the calculations defined is slightly different than usual, as shown below:

$$\text{Hits@}k = \frac{1}{\sum_{j \in N} |T_j|} \sum_{j \in N} \sum_{q \in T_j} I[\text{rank}_q \leq k],$$

where $T_j$ represents the true label set for sample $j$ and $|T_j|$ is the number of true labels. $I[C]$ is 1 if the condition $C$ is true, and 0 otherwise. rank$_q$ is the rank of $q$ in predicting the subject or object given the relation. We calculate the average metric of predicting the subject and object in the multi-actor prediction.

### E COMPUTATIONAL COMPLEXITY

Here we analyze the time complexity of the graph aggregation and embedding fusion steps. Assume that the word graph and the event graph have the same size of nodes and edges, and $|V|$ and $|E|$ are the numbers of nodes and edges respectively. The temporal event and word graphs are sparse, so the graph aggregation can work efficiently by using batching and sparse matrix operation. Given that we use a two-layer GCN and the filter dimension for both layers is $d$, the time complexity of word graph aggregation is $O(2m|E|d^2)$ where $m$ is the length of history. Using a two-layer CompGCN, event graph aggregation costs $O(2m|E|d^2 + 2m|E|d)$. The total time complexity of the graph aggregation is $O(m|E|d^2)$. The embedding fusion module fuses the entity/event type embedding and word embedding of interest by the attention mechanism. Given the dot-product attention, the matrix-vector multiplication for one entity is the product of a $1 \times d$ vector multiplied by a $d \times n$ matrix, resulting in an $O(nd)$ complexity where $n$ is the maximum sequence length, that is, the maximum number of related words for an entity/event type. Then applying softmax and linear combination requires $O(n + d)$. Therefore, the attention for an entity takes $O(nd)$. When computing the fused embeddings of all entities and relationships, the time complexity is $O(|V| + |E|)nd$. Both the graph aggregation and embedding fusion can run efficiently through batching.

### F SENSITIVITY ANALYSIS

In this section, we study the parameter sensitivity of the proposed method. We report the performance changes of our method on the Afghanistan dataset in the multi-event prediction task.

**Embedding Dimension** We investigate how the embedding feature dimension affects model performance by changing the dimension $d$. Figure 5a shows that when the dimension is greater than 100, the model can obtain better performance. This indicates that including more features can improve prediction performance. When $d$ is larger ($d = 150$), the prediction results only have slight increases.

**Layers of Graph Aggregators** We examine the number of layers in the graph aggregators, e.g., GCN for aggregating the word graphs and CompGCN for aggregating the event graphs. The number of layers in the aggregator means the depth to which the node reaches. Figure 5b shows the performances according to different numbers of layers of both GCN and CompGCN. We noticed that using 2-layer GCN can improve performance, which shows the effectiveness of aggregating semantic information. 1-layer CompGCN has comparable performance compared to 2-layer CompGCN, which illustrates the advantages of CompGCN in aggregating relational information.

**Length of History** The recurrent encoder takes the sequence of history data up to $m$ steps. Figure 6a shows the performances with varying lengths of histories. When the historical data length is around 7, the model performance reaches the best. When the length becomes longer, the performance decreases slightly, which may due to the amount of historical information that the model can handle is saturated. Note that when the length of the history is relatively short ($m = 2$), our model still get very impressive results. It shows that including semantic information from temporal word graphs can enhance the predictive power of the model.

**Attention Mechanisms** We test different attention mechanisms in the embedding fusion step. The dot-product, additive, and general attention mechanisms are considered. We can observe that different mechanisms show comparable performance in Figure 6b, which indicates that our model is insensitive to different attention mechanisms.