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 **g**raph **lea**rning framework based on event k**n**owledge 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-need 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:
Songgaojun Deng, Huzefa Rangwala, and Yue Ning. 2020. Dynamic Knowledge Graph based Multi-Event Forecasting. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3394486.3403209

KDD '20, August 23–27, 2020, Virtual Event, CA, USA

ACM ISBN 978-1-4503-7998-4/20/08…$15.00
https://doi.org/10.1145/3394486.3403209

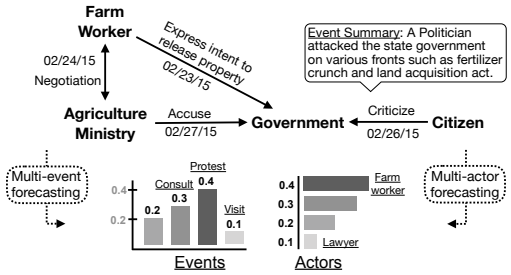


Figure 1: A motivating example of event graphs for multi-event and multi-actor forecasting. Nodes denote entities and edges denote event types and timestamps.

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 select salient actors, words, and relevant events from the historical input data based on learned weights. Our contributions are summarized as follows:

- We design a novel multi-event multi-actor forecasting framework that (1) utilizes global event information from entities, event types, and event descriptions, to predict concurrent events of multiple types; and (2) predicts potential participants (actors/entities) in these events with temporal and intrinsic inference modules based on historical events and the inherent association between entities and event types.
- We introduce a new encoding method for integrating both dynamic graph data (event graphs) and text data (documents and summaries) into graph-based relational features. We also provide a graph sampling method to obtain specific features in history for a given event-type, thereby eliminating the unwanted noise.
- Given the entities and event types in event graphs, we identify related words from event summaries and propose a context-aware embedding fusion method. We incorporate an attention mechanism to learn the importance of each identified word and capture local contextual semantics.

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 are predicting 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 $X_{1:t-1}$:

$$\mathbb{P}(\mathbf{y}_t | X_{1:t-1}). \tag{1}$$

Here $\mathbf{y}_t \in \mathbb{R}^{|\mathcal{R}|}$ 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:

$$\mathbb{P}(\mathbf{a}_t | y_t, X_{1:t-1}), \tag{2}$$

where $\mathbf{a}_t \in \mathbb{R}^{|\mathcal{E}|}$ 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 $\mathcal{E}$ be a finite set of entities and $\mathcal{R}$ 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 $s, o \in \mathcal{E}$ and $r \in \mathcal{R}$. We denote a set of events at time t as $\mathcal{G}_t = \{(s, r, o, t)\}_t$. Temporal event graphs are built upon a sequence of event sets in ascending time order $\{\mathcal{G}_1, ..., \mathcal{G}_T\}$, 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 $\mathbf{h}_u \in \mathbb{R}^d, \mathbf{o}_r \in \mathbb{R}^d$.

Table 1: Important notations and descriptions

Notations	Descriptions
$\mathcal{E}, \mathcal{R}, \mathcal{V}$	entity, event type, and vocabulary sets
$\mathcal{G}_t$, $\mathcal{D}_t$	event and word graphs at time t
$\mathcal{G}_t^r$, $\mathcal{D}_t^r$	event and word subgraphs of event type r at time t
$\mathbf{h}_u$, $\mathbf{o}_r$	learnable embedding of entity u and event type r
$\mathbf{b}_\omega$	pre-trained word2vec embedding of word ω
$\overline{\mathbf{H}}_t$	semantic embedding matrix of words at time t
$\overleftarrow{\mathbf{H}}_t$, $\overleftarrow{\mathbf{O}}_t$	relational embedding matrices of entities and event types at time t
$\mathbf{H}_t^\spadesuit$, $\mathbf{O}_t^\spadesuit$	fused embedding matrices of entities and event types at time t
d	feature dimension

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 $\mathrm{D}_t = \{\mathrm{c}\}_t$. For each timestamp t, we construct an undirected word graph $\mathcal{D}_t$ 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 ω and word φ if $\mathrm{PMI}(\omega, \varphi) > 0$, notated as $(\omega, \varphi) \in \mathcal{D}_t$, that is $\mathcal{D}_t = \{(\omega, \varphi)\}_t$. $\mathcal{V}_t$ contains all words at time t and $\mathcal{V}$ is the vocabulary set. Each word ω is initialized with a pre-trained embedding vector $\mathbf{b}_\omega \in \mathbb{R}^d$. Temporal word graphs are built from a series of summary sets, sorted in ascending time order.

In the following section, we use $\leftarrow$ over a letter to represent relational embeddings in event graphs, $-$ to denote semantic embeddings in word graphs, and $\spadesuit$ 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 $\mathbb{P}(\mathbf{y}_t)$ at time t using the temporal graphs of events ($\mathcal{G}_{t-m:t-1}$) and

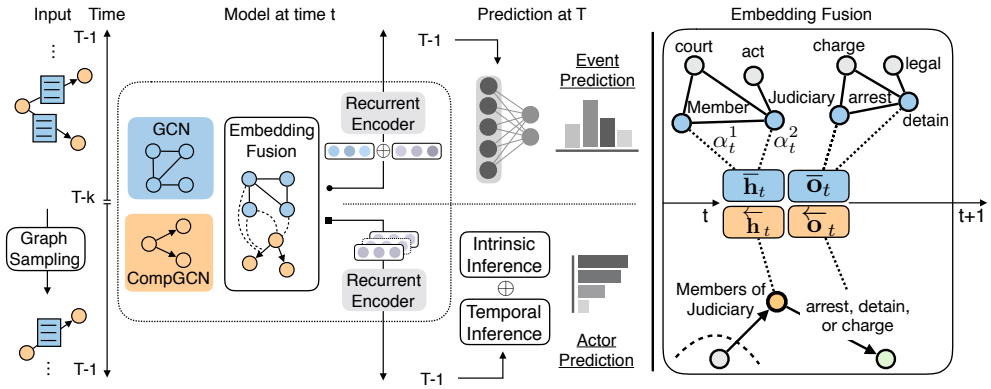


Figure 2: An overview of the proposed model. CompGCN aggregates the entity and event type embedding in event graphs at each timestamp based on Eq. 4. GCN learns the semantic embedding in each temporal word graph using Eq. 6. Embedding fusion is applied to entities and event types with related words in the word graphs according to Eq. 7-8. The recurrent encoder models temporal information for final predictions. In the multi-event forecasting task, we model the global (graph-level) temporal information for predicting the candidate event types. For the multi-actor prediction task, we model the individual-level temporal feature of entities and the given event type.

words ($\mathcal{D}_{t-m:t-1}$) at previous m steps:

$$\mathbb{P}\left(\mathbf{y}_t \middle| \mathcal{G}_{t-m:t-1}, \mathcal{D}_{t-m:t-1}\right) = \sigma\left(\mathbf{W}_\gamma \mathbf{g}_{t-1}\right) \in \mathbb{R}^{|\mathcal{R}|}, \tag{3}$$

where $\mathbf{g}_{t-1} \in \mathbb{R}^d$ is the global latent embedding at timestamp $t-1$. We compute the probabilities of different event types by passing the latent embedding into a linear layer parameterized by $\mathbf{W}_\gamma \in \mathbb{R}^{|\mathcal{R}|\times d}$ followed by an element-wise sigmoid function for our final prediction. To obtain the latent embedding $\mathbf{g}_{t-1}$, we propose three modules, the graph aggregation, the context-aware embedding fusion, and the recurrent encoder, to fully integrate the features from the historical temporal graphs. Next, we introduce these modules in detail.

Graph Aggregation To learn the representations of temporal event graphs and word graphs, we employ graph neural networks (GNN), which captures the dependency of features via message passing among adjacent nodes in a graph. Given the heterogeneous graph structure of our data, we propose to learn relational embeddings and semantic embeddings, respectively.

Relational Embedding (Event Graph). In event graphs, the edge between a pair of nodes is directed and labeled as the event type between these two nodes, e.g., "A criticizes B". Relational embedding encodes the hidden influence or actions from the historical events.

For each time t, we learn the relational embeddings of both entities and event types by capturing directed influence flows in $\mathcal{G}_t$. The features of both entities and event types are learned by a multi-layer graph convolution network. At layer $(l+1)$, for a single node v, the embedding vector is learned by the CompGCN model [33], which updates the node embedding as well as the edge embedding from relational data:

$$\overleftarrow{\mathbf{h}}_v^{(l+1)} = f\Big(\sum_{(r,u)\exists(v,r,u)\in\mathcal{G}_t} \mathbf{W}_q^{(l)} \phi\big(\overleftarrow{\mathbf{h}}_u^{(l)}, \overleftarrow{\mathbf{o}}_r^{(l)}\big)\Big), \tag{4}$$

where $\overleftarrow{\mathbf{h}}_u^{(l)}$, $\overleftarrow{\mathbf{o}}_r^{(l)}$ denotes features in l-th layer for node u and event type r, respectively. $\mathbf{W}_q^{(l)}$ is the weight parameter for aggregating node and edge features in the l-th layer. ϕ is a non-parametric composition operator (e.g. multiplication). $f(\cdot)$ is the activation function. We update the event type (edge) embedding vector by:

$$\overleftarrow{\mathbf{o}}_r^{(l+1)} = \mathbf{W}_{\text{edge}}^{(l)} \overleftarrow{\mathbf{o}}_r^{(l)}, \tag{5}$$

where $\mathbf{W}_{\text{edge}}$ is a learnable transformation matrix which projects the edges to the same embedding space as nodes. At the first layer, $\overleftarrow{\mathbf{h}}_u^{(0)} = \mathbf{h}_u$, $\overleftarrow{\mathbf{o}}_r^{(0)} = \mathbf{o}_r$ are the learnable embeddings of entity u and event type r. After the aggregation, we obtain the relational embedding matrices of all entities $\overleftarrow{\mathbf{H}}_t \in \mathbb{R}^{|\mathcal{E}_t|\times d}$ and event types $\overleftarrow{\mathbf{O}}_t \in \mathbb{R}^{|\mathcal{R}_t|\times d}$ 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 ω in $\mathcal{D}_t$, we apply a multi-layer GCN model [16] to update its embedding vector:

$$\overline{\mathbf{h}}_\omega^{(l+1)} = f\Big(\sum_{(\omega,\varphi)\in\mathcal{D}_t} \mathbf{W}_g^{(l)} \overline{\mathbf{h}}_\varphi^{(l)}\Big), \tag{6}$$

where $\overline{\mathbf{h}}_\varphi^{(0)} = \mathbf{b}_\varphi$ is the word2vec vector of φ at the first layer. $\mathbf{W}_g^{(l)}$ is the weight matrix. Semantic embedding matrix $\overline{\mathbf{H}}_t \in \mathbb{R}^{|\mathcal{V}_t|\times d}$ 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, (*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 *Government* after aggregation only involves the relational information, e.g., involved

subjects and event types. While the word *Government* in the text incorporates detailed background information of the event such as *fertilizer crunch* and *land acquisition act*. Further integration of this information can greatly enhance the context of events (including entities, event types, and useful details) for downstream tasks. We introduce a novel context-aware embedding fusion module to enhance the information of entities and event types 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) $\omega \in \mathcal{W}_i$, where $\mathcal{W}_i$ is the set of words in $\mathcal{D}_{t-m:t-1}$ that are semantically related to i:

$$\alpha_{t,(i,\omega)} = \frac{\exp\left(\text{Attn}\left(\overleftarrow{\mathbf{h}}_{t,(i)}, \overline{\mathbf{h}}_{\omega}\right)\right)}{\sum_{\varphi \in \mathcal{W}_i} \exp\left(\text{Attn}\left(\overleftarrow{\mathbf{h}}_{t,(i)}, \overline{\mathbf{h}}_{\varphi}\right)\right)} \in \mathbb{R}, \tag{7}$$

where $\text{Attn}(\cdot)$ denotes the attention mechanism (e.g., general attention [18]). $\overleftarrow{\mathbf{h}}_{t,(i)}$ is the relational embedding of entity i in $\mathcal{G}_t$, and $\overline{\mathbf{h}}_{\omega}$ represents the semantic embedding of word ω in $\mathcal{D}_{t-m:t-1}$ 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:

$$\mathbf{h}^{\spadesuit}_{t,(i)} = \tanh\left(\mathbf{W}_{\alpha}^{\intercal} \cdot \Big[\underbrace{\overleftarrow{\mathbf{h}}_{t,(i)}}_{\text{rel.}}; \underbrace{\sum_{\omega \in \mathcal{W}_i} \alpha_{t,(i,\omega)} \overline{\mathbf{h}}_{\omega}}_{\text{semantic}}\Big]\right) \in \mathbb{R}^d, \tag{8}$$

where $\mathbf{W}_{\alpha} \in \mathbb{R}^{2d \times d}$ is a weight matrix, and ; 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 $\mathcal{D}_{t-2}$ and $\mathcal{D}_{t-1}$, we use the semantic embedding of the word in $\mathcal{D}_{t-1}$. If there are no related words for an entity, we use zero vectors for semantic information. For event type i, we replace $\overleftarrow{\mathbf{h}}_{t,(i)}$ with $\overleftarrow{\mathbf{o}}_{t,(i)}$ in Eq. 7-8 to get its fused embedding $\mathbf{o}^{\spadesuit}_{t,(i)}$.

Recurrent Encoder Given a sequence of fused embeddings of entities and event types, as well as the word embeddings, i.e., $\{\mathbf{H}^{\spadesuit}_{t-m:t-1}, \mathbf{O}^{\spadesuit}_{t-m:t-1}, \overline{\mathbf{H}}_{t-m:t-1}\}$, 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 $\mathbf{g}_t$:

$$\mathbf{g}_t = \text{RNN}^1\Big(\big[\text{p}(\mathbf{H}^{\spadesuit}_{t-1}); \text{p}(\mathbf{O}^{\spadesuit}_{t-1}); \text{p}(\overline{\mathbf{H}}_{t-1})\big], \mathbf{g}_{t-1}\Big) \in \mathbb{R}^d. \tag{9}$$

where $\text{p}(\cdot)$ indicates the max pooling operation applied element-wise over all nodes or edges. The latent embedding $\mathbf{g}_t$ 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 $\mathcal{G}_t$ at time t, we first sample the quadruples where the event type is r, notated as $\text{SET}_r = \{(s, r, o)_i | r_i = r \cap (s, r, o)_i \in \mathcal{G}_t\}$. We then obtain the subject set $\text{SET}_s = \{s | s \in \text{SET}_r\}$ to find the neighboring edges. The first level neighboring edge set is $\text{SET}^{\text{nbr}}_r = \{(s, r, o) | o \in \text{SET}_s\}$. The total sampled edge set is $\text{SET}_r \cup \text{SET}^{\text{nbr}}_r$. 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 $\mathcal{G}^r_t$. 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 $\mathcal{D}^r_t$. 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 $\mathbb{P}\big(\mathbf{a}_t \big| y_t = r\big)$ in the following way:

$$\mathbb{P}\big(\mathbf{a}_t \big| y_t, \mathcal{G}^{y_t}_{t-m:t-1}, \mathcal{D}^{y_t}_{t-m:t-1}\big) = \sigma\big(\tilde{\mathbf{z}} + \mathbf{z}\big) \in \mathbb{R}^{|\mathcal{E}|}, \tag{10}$$

where $\tilde{\mathbf{z}}$ is the temporal inference, which encodes temporal historical features for each actor. $\mathbf{z}$ 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:

$$\mathbf{z} = \mathbf{H} \cdot \mathbf{W}_{\beta} \cdot \mathbf{o}_r \in \mathbb{R}^{|\mathcal{E}|}, \tag{11}$$

where $\mathbf{o}_r \in \mathbb{R}^d$ is the learnable embedding vector specified for event type r. $\mathbf{H} \in \mathbb{R}^{|\mathcal{E}| \times d}$ is the embedding matrix for all entities in the predefined entity set $\mathcal{E}$, where each row ($\mathbf{h}_u$) represents the embedding of an entity (u). $\mathbf{W}_{\beta} \in \mathbb{R}^{d \times d}$ 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

affects its future occurrence, we model the features individually for the nodes and edges. We compute the fused embedding matrix of entities in $\mathcal{G}_t^r$, i.e., $\mathbf{H}_t^{\spadesuit} \in \mathbb{R}^{|\mathcal{E}_t^r| \times d}$, and the fused embedding vector of event type r, i.e., $\mathbf{o}_{t,(r)}^{\spadesuit} \in \mathbb{R}^d$ according to Eq. 7-8. $\mathcal{E}_t^r \subseteq \mathcal{E}$ is the entity set in $\mathcal{G}_t^r$.

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

$$\widetilde{\mathbf{h}}_{t,(i)} = \text{RNN}^2\big(\mathbf{h}_{t,(i)}^{\spadesuit}, \widetilde{\mathbf{h}}_{t-1,(i)}\big) \in \mathbb{R}^d, \tag{12}$$

$$\widetilde{\mathbf{o}}_{t,(r)} = \text{RNN}^3\big(\mathbf{o}_{t,(r)}^{\spadesuit}, \widetilde{\mathbf{o}}_{t-1,(r)}\big) \in \mathbb{R}^d. \tag{13}$$

We model the latent embedding for each entity i using a shared RNN^2, and use $\widetilde{\mathbf{H}}_t$ 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 $\widetilde{\mathbf{o}}_{t,(r)}$. We then model the temporal features at time t by using the latent embedding of the event type r and all entities at time $t-1$:

$$\tilde{\mathbf{z}} = \widetilde{\mathbf{H}}_{t-1} \cdot \widetilde{\mathbf{W}}_\beta \cdot \widetilde{\mathbf{o}}_{t-1,(r)} \in \mathbb{R}^{|\mathcal{E}|}, \tag{14}$$

where $\widetilde{\mathbf{W}}_\beta \in \mathbb{R}^{d \times d}$ is a learnable weight matrix. Both intrinsic embedding $\mathbf{z}$ 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 $\mathcal{Y} = \{e_i\}^L$. e_i is a decimal in [0,1.0] and $\sum_i^L e_i = 1$. 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 $\mathbf{y} \in \mathcal{Y}$, 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 $|\mathcal{R}|$ or entities $|\mathcal{E}|$ 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:

$$-\frac{1}{L}\sum_{i \in L} y_i \log\Big(\frac{\exp(\hat{y}_i)}{\sum_{j \in L}\exp(\hat{y}_j)}\Big), \tag{15}$$

where $\hat{y}_i$ is the model prediction for label i before the nonlinear function (σ) as in Eq. 3 and 10.

At inference time, we utilized a sigmoid function over the class score $\hat{y}_i$ 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 $\hat{y}_i$ among $\{\hat{y}_i\}^L$.

Table 2: Dataset Statistics. $|c|$ is the average number of words in event summaries. #s(o)/r denotes the average number of subject (object) that interacted with one event type per day.

dataset	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{R}\rvert$	$\lvert\mathcal{V}\rvert$	#events	$\lvert c\rvert$	#s/r	#o/r
India	6,298	234	23,373	479,649	19	7.0	7.9
Russia	7,798	237	16,299	373,364	22	4.7	4.7
Nigeria	3,896	221	13,933	237,096	21	4.0	3.7
Afghanistan	3,756	218	10,468	279,859	19	4.0	4.0
Iran	7,347	229	13,008	328,266	22	4.4	4.1

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 type/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**β 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

Table 3: Prediction results of the proposed method vs. baselines for the multi-event forecasting task over all datasets (%).

Method	India			Russia			Nigeria			Afghanistan			Iran		
	F1	F2	Recall	F1	F2	Recall	F1	F2	Recall	F1	F2	Recall	F1	F2	Recall
DNN	52.49	54.65	56.38	53.81	58.44	62.61	53.54	60.64	67.70	55.77	61.80	68.14	57.54	61.85	66.19
MLkNN	52.33	54.27	55.77	51.38	55.29	58.62	26.92	28.10	28.97	45.43	48.10	50.35	53.86	56.68	59.01
BRkNN	50.36	53.05	56.00	47.46	51.53	56.64	42.48	47.28	52.45	49.89	54.98	61.52	48.56	52.24	56.77
MLARAM	33.68	33.93	34.10	25.67	26.27	26.71	41.78	45.56	48.80	33.84	34.66	35.26	27.46	27.71	27.88
DynGCN	41.80	42.57	43.19	52.81	56.77	60.14	46.27	54.65	54.65	50.05	53.97	57.75	54.22	56.93	59.21
T-GCN	60.73	64.14	67.20	56.36	61.86	67.66	56.06	63.88	72.19	60.04	67.82	76.93	61.65	67.35	73.77
RENET^1	55.10	57.26	58.99	54.47	58.98	63.02	53.47	60.07	66.54	55.07	60.60	66.32	58.89	63.41	68.09
RENET^2	58.44	61.46	64.18	55.85	60.86	65.66	56.44	64.37	72.82	60.58	68.47	77.75	61.66	67.24	73.52
Glean$_{-\text{fusion}}$	65.91	70.87	75.80	**58.92**	65.60	73.47	58.13	66.95	77.07	62.28	71.14	82.36	63.84	70.78	79.60
Glean	**66.69**	**71.95**	**77.31**	**58.92**	**65.64**	**73.57**	**58.76**	**68.13**	**79.49**	**62.48**	**71.43**	**82.84**	**64.12**	**71.25**	**80.46**
% relative gain	9.8%	10.9%	15.0%	4.5%	6.1%	8.7%	4.8%	5.8%	10.1%	3.1%	4.3%	6.5%	4.0%	5.8%	9.1%

balancing factor for precision and recall. Higher β value means higher weight of recall in the F score. β is set to 1 and 2.

- **Recall** evaluates the ability of a model finding all relevant cases.
- **Hits**@k is the percentage of ranks lower than or equal to k: the higher, the better. It is often used in the evaluation of information retrieval tasks [3, 28]. k is set to 1, 3, and 10 in our experiments.

We consider the weighted averaging of **F**β 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.
- MLARAM [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]: CompGCN 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$_{-\text{fusion}}$ removes the embedding fusion step, and Glean$_{-\text{temp}}$ 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 using 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

Table 4: Performance results of the proposed method vs. baselines for the multi-actor forecasting task over all datasets (%).

Method	India			Russia			Nigeria			Afghanistan			Iran		
	H @ 1	3	10	1	3	10	1	3	10	1	3	10	1	3	10
DNN	2.09	11.01	33.87	1.46	9.72	36.40	5.10	17.06	**43.35**	8.55	17.42	35.32	10.71	19.48	26.50
RENET^3	8.87	21.57	39.85	16.52	22.31	40.21	4.02	11.53	26.95	7.28	18.65	37.44	12.81	18.36	37.44
tRGCN	9.74	22.74	41.04	18.83	30.79	44.62	6.73	15.17	31.69	9.58	24.14	49.17	12.93	22.26	34.98
tCompGCN	9.62	21.91	40.53	18.27	30.20	44.79	6.50	14.95	31.06	9.64	23.67	49.04	12.79	21.43	34.88
Glean$_{-\text{temp}}$	13.39	24.50	43.68	18.24	31.15	43.27	6.16	14.41	26.98	9.21	22.27	47.03	11.01	17.96	29.87
Glean$_{-\text{fusion}}$	13.95	27.03	**45.73**	20.25	**34.64**	**48.10**	7.63	**18.06**	35.84	12.28	29.82	**56.89**	14.27	**24.41**	39.74
Glean	**14.01**	**27.17**	**45.73**	**20.49**	34.36	**48.10**	**7.66**	18.03	35.85	**12.29**	**30.04**	56.74	**14.31**	24.27	**39.75**
% relative gain	4.6%	10.9%	4.7%	8.8%	11.2%	7.4%	13.8%	5.9%	-	27.5%	24.4%	15.7%	10.7%	9.7%	6.2%

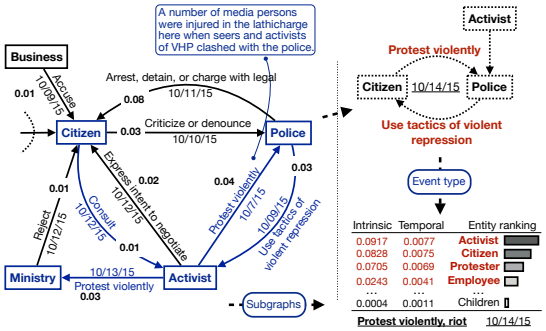


Figure 3: An example of predicted multiple events with actors in India. The blue part is from the sampled subgraph. Red is the predicted result.

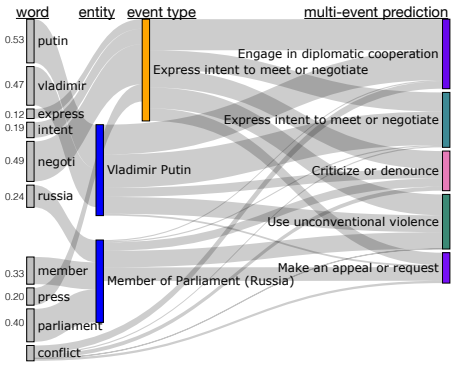


Figure 4: An example of feature flows in the multi-event forecasting task.

all datasets. DNN uses historical counts of entities, which did not achieve good performance in all datasets except Nigeria(H@10). For baseline models, RENET^3 models the global information (pooling is applied) for predicting actors, which also considers relational embedding. tRGCN and tCompGCN utilize relational information and both involve node-level temporal information for predicting the actors. These models that consider only relational features perform worse than our approach. It proves the importance of considering enriched relational features and better design of feature learning. For the ablation study of our method, we can observe that the removal of temporal inference degrades model performance, indicating that historical data is beneficial for predicting actors. The model that removes the embedding fusion achieves comparable performance. One possible reason is that after graph sampling, word subgraphs may not contain enough semantic information, making the fused embedding unhelpful. It also demonstrates the importance of relational information in predicting event participants.

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* by *Police* 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 $\mathbf{z}$ and $\tilde{\mathbf{z}}$ 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 *negoti(ate)* 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

[1] Fernando Benites and Elena Sapozhnikova. 2015. Haram: a hierarchical aram neural network for large-scale text classification. In *ICDMW*. IEEE, 847–854.
[2] Johan Bollen, Huina Mao, and Xiaojun Zeng. 2011. Twitter mood predicts the stock market. *Journal of computational science* 2, 1 (2011), 1–8.
[3] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*. 2787–2795.
[4] Elizabeth Boschee, Jennifer Lautenschlager, Sean O'Brien, Steve Shellman, James Starz, and Michael Ward. 2015. ICEWS Coded Event Data.
[5] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *EMNLP*. Association for Computational Linguistics, 1724–1734.
[6] Kenneth Ward Church and Patrick Hanks. 1990. Word Association Norms, Mutual Information, and Lexicography. *Comput. Linguist.* 16, 1 (March 1990), 22–29.
[7] Shib Sankar Dasgupta, Swayambhu Nath Ray, and Partha Talukdar. 2018. Hyte: Hyperplane-based temporally aware knowledge graph embedding. In *EMNLP*. 2001–2011.
[8] Songgaojun Deng, Huzefa Rangwala, and Yue Ning. 2019. Learning Dynamic Context Graphs for Predicting Social Events. In *KDD*. ACM, 1007–1016.
[9] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *AAAI*.
[10] Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. 2017. Protein Interface Prediction Using Graph Convolutional Networks. In *NIPS (NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 6533–6542.
[11] Yuyang Gao, Liang Zhao, Lingfei Wu, Yanfang Ye, Hui Xiong, and Chaowei Yang. 2019. Incomplete Label Multi-Task Deep Learning for Spatio-Temporal Event Subtype Forecasting. In *AAAI*, Vol. 33. 3638–3646.
[12] Alberto García-Durán, Sebastijan Dumančić, and Mathias Niepert. 2018. Learning sequence encoders for temporal knowledge graph completion. *arXiv preprint arXiv:1809.03202* (2018).
[13] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*. 249–256.
[14] Woojeong Jin, Changlin Zhang, Pedro Szekely, and Xiang Ren. 2019. Recurrent event network for reasoning over temporal knowledge graphs. *arXiv preprint arXiv:1904.05530* (2019).
[15] D Kinga and J Ba Adam. 2015. A method for stochastic optimization. In *ICLR*, Vol. 5.
[16] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.
[17] Julien Leblay and Melisachew Wudage Chekol. 2018. Deriving validity time in knowledge graph. In *WWW*. International World Wide Web Conferences Steering Committee, 1771–1776.
[18] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* (2015).
[19] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. 2018. Exploring the limits of weakly supervised pretraining. In *ECCV*. 181–196.
[20] Diego Marcheggiani and Ivan Titov. 2017. Encoding Sentences with Graph Convolutional Networks for Semantic Role Labeling. In *EMNLP*. Association for Computational Linguistics, 1506–1515.
[21] Aditya K Menon, Ankit Singh Rawat, Sashank Reddi, and Sanjiv Kumar. 2019. Multilabel reductions: what is my loss optimising?. In *NIPS*. 10599–10610.
[22] Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML*. 807–814.
[23] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data.. In *ICML*, Vol. 11. 809–816.
[24] Yue Ning, Sathappan Muthiah, Huzefa Rangwala, and Naren Ramakrishnan. 2016. Modeling precursors for event forecasting via nested multi-instance learning. In *KDD*. ACM, 1095–1104.
[25] Yue Ning, Rongrong Tao, Chandan K Reddy, Huzefa Rangwala, James C Starz, and Naren Ramakrishnan. 2018. STAPLE: Spatio-Temporal Precursor Learning for Event Forecasting. In *SIAM*. SIAM, 99–107.
[26] Jiezhong Qiu, Jian Tang, Hao Ma, Yuxiao Dong, Kuansan Wang, and Jie Tang. 2018. DeepInf: Modeling influence locality in large social networks. In *KDD*.
[27] Sashank J Reddi, Satyen Kale, Felix Yu, Dan Holtmann-Rice, Jiecao Chen, and Sanjiv Kumar. 2018. Stochastic negative mining for learning with large output spaces. *arXiv preprint arXiv:1810.07076* (2018).
[28] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *ESWC*. Springer, 593–607.
[29] Alessio Signorini, Alberto Maria Segre, and Philip M Polgreen. 2011. The use of Twitter to track levels of disease activity and public concern in the US during the influenza A H1N1 pandemic. *PloS one* 6, 5 (2011), e19467.
[30] Eleftherios Spyromitros, Grigorios Tsoumakas, and Ioannis Vlahavas. 2008. An empirical study of lazy multilabel classification algorithms. In *Hellenic conference on artificial intelligence*. Springer, 401–406.
[31] Rakshit Trivedi, Hanjun Dai, Yichen Wang, and Le Song. 2017. Know-evolve: Deep temporal reasoning for dynamic knowledge graphs. In *ICML*. JMLR. org, 3462–3471.
[32] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *ICML*. 2071–2080.
[33] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. 2019. Composition-based multi-relational graph convolutional networks. *arXiv preprint arXiv:1911.03082* (2019).
[34] Xiaofeng Wang, Matthew S Gerber, and Donald E Brown. 2012. Automatic crime prediction using events extracted from twitter posts. In *International conference on social computing, behavioral-cultural modeling, and prediction*. Springer, 231–238.
[35] Jianshu Weng and Bu-Sung Lee. 2011. Event detection in twitter. *ICWSM* 11 (2011), 401–408.
[36] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575* (2014).
[37] Min-Ling Zhang and Zhi-Hua Zhou. 2007. ML-KNN: A lazy learning approach to multi-label learning. *Pattern recognition* 40, 7 (2007), 2038–2048.
[38] Ling Zhao, Yujiao Song, Chao Zhang, Yu Liu, Pu Wang, Tao Lin, Min Deng, and Haifeng Li. 2019. T-gcn: A temporal graph convolutional network for traffic prediction. *IEEE Transactions on Intelligent Transportation Systems* (2019).
[39] Liang Zhao, Qian Sun, Jieping Ye, Feng Chen, Chang-Tien Lu, and Naren Ramakrishnan. 2015. Multi-task learning for spatio-temporal event forecasting. In *KDD*. ACM, 1503–1512.

Supplemental Material

A PSEUDOCODE

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 $\{\mathcal{G}_1, ..., \mathcal{G}_T\}$, temporal word graphs $\{\mathcal{D}_1, ..., \mathcal{D}_T\}$, pre-trained word embeddings, initial model parameters Θ.

Output: A trained multi-event classifier $f : (\mathcal{G}_{t-m:t-1}; \mathcal{D}_{t-m:t-1}) \rightarrow \mathbb{P}(\hat{\mathbf{y}}_t)$

while *stopping criterion not met* **do**
- Sample a minibatch of k examples
- **for** $\tau \leftarrow t-m$ *to* $t-1$ **do**
  - $\overleftarrow{\mathbf{H}}_\tau, \overleftarrow{\mathbf{O}}_\tau \leftarrow$ aggregate event graph $\mathcal{G}_\tau$
  - $\overline{\mathbf{H}}_\tau \leftarrow$ aggregate word graph $\mathcal{D}_\tau$
- $\{\mathbf{H}^{\spadesuit}_{t-m:t-1}, \mathbf{O}^{\spadesuit}_{t-m:t-1}\} \leftarrow$ apply embedding fusion
- $\mathbf{g}_{t-1} \leftarrow$ apply recurrent encoder
- Compute $\mathbb{P}(\hat{\mathbf{y}}_t)$ based on Eq. 3
- Compute loss based on Eq. 15 and update Θ

Algorithm 2: Multi-actor forecasting algorithm

Input: Temporal event subgraphs $\{\mathcal{G}^r_1, ..., \mathcal{G}^r_T\}$, temporal word subgraphs $\{\mathcal{D}^r_1, ..., \mathcal{D}^r_T\}$, pre-trained word embeddings, initial model parameters Θ.

Output: A trained multi-event classifier $f : (\mathcal{G}^r_{t-m:t-1}; \mathcal{D}^r_{t-m:t-1}) \rightarrow \mathbb{P}(\hat{\mathbf{a}}_t|r)$

while *stopping criterion not met* **do**
- Sample a minibatch of k examples
- $\mathbf{z} \leftarrow$ compute intrinsic inference
- $\tilde{\mathbf{z}} \leftarrow$ compute temporal inference
- Compute $\mathbb{P}(\hat{\mathbf{a}}_t|r)$ based on Eq. 10
- Compute loss based on Eq. 15 and update Θ

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 ϕ 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 RGCN or CompGCN 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:

$$\begin{aligned}
\mathbf{x}_t &= \left[\mathrm{p}(\mathbf{H}^{\spadesuit}_t); \mathrm{p}(\mathbf{O}^{\spadesuit}_t); \mathrm{p}(\overline{\mathbf{H}}_t)\right] \\
\mathbf{r}_t &= \sigma(\mathbf{W}_r \cdot [\mathbf{g}_{t-1}; \mathbf{x}_t]) \\
\mathbf{z}_t &= \sigma(\mathbf{W}_z \cdot [\mathbf{g}_{t-1}; \mathbf{x}_t]) \\
\tilde{\mathbf{h}}_t &= \tanh(\mathbf{W} \cdot [r_t \odot \mathbf{g}_{t-1}; \mathbf{x}_t]) \\
\mathbf{g}_t &= (1 - \mathbf{z}_t) \odot \mathbf{g}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t,
\end{aligned}$$

where $\mathbf{x}_t, \mathbf{g}_t$ are the input and latent embedding (GRU hidden state) at time t. $\mathbf{r}_t, \mathbf{z}_t, \tilde{\mathbf{h}}_t$ are the reset, update, and new gates, respectively. $\sigma(\cdot)$ 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:

$$\begin{aligned}
\text{Precision} &= \frac{1}{L}\sum_{i \in L} w_i \cdot \frac{TP_i}{TP_i + FP_i}, \\
\text{Recall} &= \frac{1}{L}\sum_{i \in L} w_i \cdot \frac{TP_i}{TP_i + FN_i}, \\
\text{F}\beta &= \frac{1}{L}\sum_{i \in L} w_i \cdot \frac{(1+\beta^2) \cdot \text{Precision}_i \cdot \text{Recall}_i}{(\beta^2 \cdot \text{Precision}_i) + \text{Recall}_i},
\end{aligned}$$

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:

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

where $\mathcal{T}_j$ represents the true label set for sample j and $|\mathcal{T}_j|$ is the number of true labels. $I[C]$ is 1 iff 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

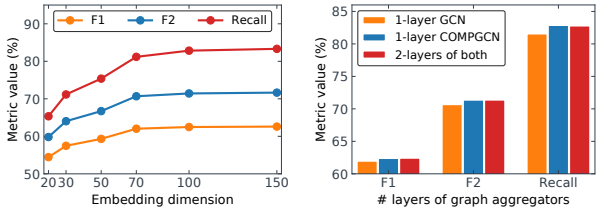
Here we analyze the time complexity of the graph aggregation and embedding fusion steps. Assume that the word graph and the event graph have the same size of nodes and edges, and $|V|$ and $|E|$ are the numbers of nodes and edges respectively. The temporal event and word graphs are sparse, so the graph 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

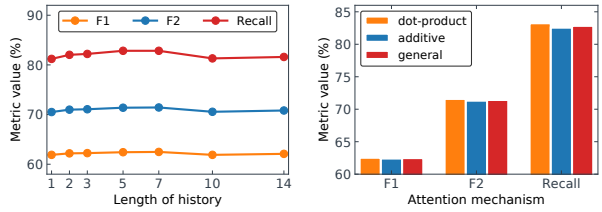
In this section, we study the parameter sensitivity of the proposed method. We report the performance changes of our method on the **Afghanistan** dataset in the multi-event prediction task.

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.



(a) Embedding dimension (b) # layers of graph aggregators



(a) Length of history (b) Attention mechanism

Figure 6: Sensitivity analysis.

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.