Text-enhanced Multi-Granularity Temporal Graph Learning for Event Prediction

Xiaoxue Han
Stevens Institute of Technology
xhan26@stevens.edu

Yue Ning
Stevens Institute of Technology
yue.ning@stevens.edu

***Abstract*—When working with forecasting the future, it is all about learning from the past. However, it is non-trivial to model the past due to the scale and complexity of available data. Recently, Graph Neural Networks (GNNs) have shown flexibility to process different forms of data and learn interactions among entities, giving them advantages in real-life applications. More and more researchers have started to apply GNNs and temporal models for event forecasting because events are formalized in knowledge graphs. However, most of these models are based on the Markov assumption that the probability of a event is only influenced by the state of its last time step (or recent history). We claim that the occurrence of an event not only has short-term but also long-term dependencies. In this work, we propose a temporal knowledge graph (KG)-based model that considers different granularties of histories when forecasting an event; this method also integrates news texts as auxiliary features during the graph learning process. Extensive experiments on multiple datasets are conducted to examine the effectiveness of the proposed method. Code is available at: https://github.com/yuening-lab/MTG.**

***Index Terms*—Text-enriched Knowledge Graphs; Dynamic Graph Neural Networks; Multiple Temporal Granularities**

I. Introduction

Graphs have been used to represent data and knowledge in many real-life applications [1], [2]. For instance, social networks [3] are graphs where nodes represent individuals and edges represent connections. Knowledge graphs (KGs) [4] store information between entities by their relations. KGs can also be used to demonstrate events between entities (e.g., A attacks B). While human data evolve over time, graphs also change over time (e.g., new relation types, deletions of edges/nodes). How to effectively model temporal information in graphs and make predictions based on historical changing graphs becomes a challenging problem.

In recent years, more and more researchers started to leverage advanced machine learning technologies on graph-based applications such as forecasting societal events [5]–[7]. However, most of the approaches are based on the assumption that the Markov property is held, which means the occurrence of an event is only dependent on very recent history. In reality, this assumption does not hold most of the time. We claim that events not only have short-term dependencies on the past few days but also have long-term dependencies that can be traced back to months or even years before. For instance, long-term economic depression affects the occurrence of many types of societal events. Intuitively, if we train a machine learning model to predict if target events such as protests would happen

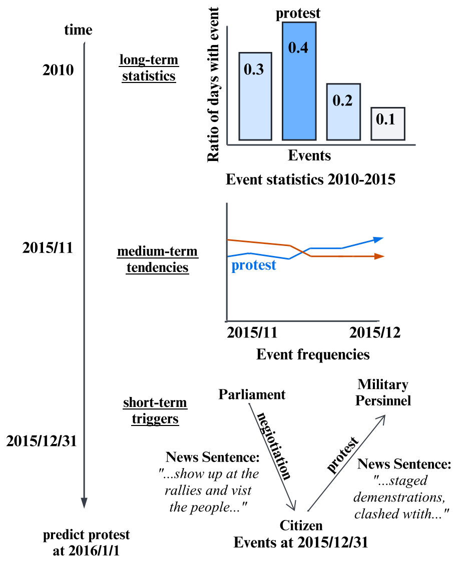


Fig. 1. A prediction example using multi-granular data including long-term statistics, medium-term tendencies, and short-term triggers. To predict if a protest will happen on 2016/6/1, three factors should be taken into considerations: 1) *Short-term triggers*. Events from 2015/12/31 may trigger a protest. 2) *Medium-term tendency*. In the previous two months, people tend to demonstrate and protest. 3) *Long-term statistics*. Historically, protests happened on 40% of days in the past five years.

tomorrow in a city, there are several factors that should be taken into consideration: 1) *Short-term triggers*. Are there any events that happened in the last few days that may trigger a protest? 2) *Medium-term tendency*. Do people tend to protest these days? This could be affected by many factors such as the current economy and unemployment rates, and a look-back period more than few days is required to estimate such a tendency. 3) *Long-term statistics*. Is this society in a stable status? Statistics obtained on the scale of years can act as prior knowledge for a prediction model. A demonstration of this multi-granular data is provided in Figure 1. Most existing work only models *short-term triggers* but ignores the other two factors, which limits the performance of those models in making more accurate and reliable predictions.

In this work, we focus on the task of forecasting large-scale societal events. Historical event information is formulated as quadruples (source_actor, event_type, target_actor, time) that summarize the key factors of events. For instance, the event *Thailand Citizen demonstrated a protest against Thailand Military Personnel on 7/22/2010* can be summarized as (Thailand Citizen, protest, Thailand Military Personnel, 7/22/2010). These quadruples further form a knowledge graph. In addition, the corresponding piece of text describes the event: "The sources said Thaksin gave his version of the troubles that transpired in Bangkok between March and May this year, in which the fugitive former premier's supporters staged anti-government demonstrations, clashed with security forces and rioted". We view the quadruples as compact representations of events and the corresponding texts as detail-enriched auxiliary features. Our goal is to propose a temporal graph learning model that fully utilizes historical information in different forms to make predictions. Specifically, we aim to solve two open problems: 1) How to design a knowledge graph model such that it can capture historical information in different hierarchies (short-term triggers, medium-term tendencies, and long-term statistics)? 2) How to leverage text information to enhance the prediction performance?

To solve these problems, we propose an end-to-end text-enriched graph learning model, **MTG**, that takes into account **m**ultiple **t**emporal **g**ranularities. It mainly consists of three parts: a *memory* module to learn long-term statistics; a *cache* module to learn medium-term tendencies; and a dynamic graph module to learn recent short-term triggers. We propose a text-enriched knowledge graph-based network, Text-CompGCN, such that text features can be seamlessly integrated with the model. To validate the proposed method we conduct experiments on four different datasets. A variety of baselines are compared to prove the effectiveness of our model.

Our contributions can be summarized as below:

- We develop an end-to-end temporal graph-based model that learns embedding vectors for entities and event types (i.e., entity relations) through different historical hierarchies. We design a message passing function to learn the evolution of relation types. We also introduce a method to update and preserve long-term historical information.
- We introduce a new GNN architecture that fuses text features into the aggregation and message passing processes. This method learns coherent representations via events and texts and frees us from the need to design separate modules for texts.
- We conduct extensive experiments and perform ablation studies to analyse the effectiveness of leveraging different components in our proposed method. We also compare our method with baselines utilizing different forms of input data. We provide readers useful insights on what kind of information might be most useful and how to efficiently construct desired features for temporal event forecasting tasks.

II. Related Work

A. Dynamic Graph Learning

With its capacity to efficiently model interactions in graph data, graph representation learning has gained increasing popularity in machine learning [8], [9]. Traditional graph learning methods often focus on static graphs [10]–[13]. However, in many real-life systems such as social networks [14], [15], product review networks [1], and protein-protein interactions (PPIs) [16], node interactions are dynamic, which means these interactions change over time. This temporal information is critical for graph-based predictions [17]. In recent years, there are many studies on dynamic graph learning [18], [19]. In general, these approaches can be summarized into two categories: *discrete snapshots* and *continuous-time graphs* based on the method of modeling temporal information [20], [21].

The general way to implement *discrete snapshots* is to model the interactions within a coarse-grained time interval as a static graph [7]; a series of static snapshots are connected in the graph-level with time series models such as Recurrent Neural Networks (RNNs) [22]. *Continuous-time graph learning* methods have more diverse approaches. Rossi *et al.* [17] propose Temporal Graph Networks (TGN) which learns the temporal embedding of nodes by continuously updating the memory of a node with messages generated from node interactions. Deng *et al.* [7] propose a Dynamic Graph Convolutional Network (DynamicGCN) that captures node-level temporal information through temporal encoded features. Temporal Point Process (TPP)-based approaches [23], [24] also demonstrate impressive performance in learning long-term evolution processes by modeling the conditional intensity of past events. Another approach treats time stamps as an additional feature of a link and assigns time-dependent weights to links [25].

Both approaches have their advantages and disadvantages: discrete methods treat snapshots as static graphs and only learn graph-level changes over time. Continuous-time graph methods, on the other hand, are able to keep track of the dynamics at node- and edge- levels, thus are considered to be fully "dynamic" compared to discrete methods. However, such approaches can not provide a precise representation of graphs due to the "staleness" problem [26] where node or edge representations that are updated long time ago are outdated for the present time. In our work, we aim to combine these approaches: we use a continuous dynamic graph-based module to learn the long-term evolution of entities (nodes) and event types (edges), and a discrete graph-based module to learn short-term node-level interactions and make final predictions.

B. Event Forecasting

Human events, such as protests, strikes, and occupations, often show recurring patterns. Forecasting these events based on historical event records has become a promising topic [27]. Analyzing such events can provide people opportunities to evaluate their strategies or take actions to minimize their losses. As deep learning has achieved success in a wide range of fields, more and more researchers have attempted to develop deep learning models for event prediction tasks which have demonstrated superiority over traditional methods [6], [7], [28], [29].

Cortez *et al.* [28] develop a Long Short-Term Memory network (LSTM)-based model that predicts emergency events (e.g. environmental emergency, traffic accidents, and crime) by learning the historical evolution with a look-back of 5 or 8 days. Deng *et al.* [6] introduce a dynamic graph convolutional network to predict the occurrence of protest events while providing dynamic graph contexts in short-term history. Wu *el at.* [29] propose a hierarchically structured Transformer Network that predicts spatial events based on spatial-temporal information retrieved in past time steps. All the methods mentioned above are based on the Markov assumption that the probability of an event is only dependent on very recent history, which does not always hold in real-life scenarios and thereby limits their performance. Also, aforementioned methods mainly utilize events or text data as model input. Deng *et al.* [7] combine both event knowledge graphs and text data in a fusion module for predicting multiple types of future events. However, this work only utilizes text data (e.g., news articles) as word graphs which have limited semantic representations and cannot capture contextual information. We believe that it is critical to integrate text as auxiliary features more coherently. Thus we aim to design a knowledge graph neural network that integrates text while making predictions based on historical temporal graphs, such that it simultaneously learns text (semantic) and event (relational) patterns during message passing processes instead of modeling them with separate networks.

III. Problem Formulation

Given the historical input data $X_{1:t}$, the goal of this work is to model the probability of a certain type of events occurring at a future timestamp t:

$$\mathbb{P}(y_{t+\Delta t}|X_{1:t}), \tag{1}$$

where $\Delta t > 0$ is the lead time, $y_{t+\Delta t} \in [0,1]$ represents the occurrence of the target event type. $X_{1:t}$ is the sequence of encoded features of the past events $E_{1:t}$ where $E_t = \{e_1, e_2, ..., e_m\}$ is a set of events (m) occurred at time t. In this work, each event is formulated as a tuple $e_i = (s_i, r_i, o_i, \mathbf{f}_i, t)$ and events that happen on the same day can be organized into a knowledge graph. For the i-th event, s_i is the source entity, o_i is the target entity, r_i is the event type, and $\mathbf{f}_i$ is an embedding vector that encodes the news sentences describing the event. Note that t is a changing variable and our work focuses on making online predictions while new data keep arriving.

The objective of this work is to propose a graph neural network to capture multi-granular temporal information in the past and to forecast target events (e.g. protests) at a future time step. A summary of important mathematical notations is provided in Table I.

TABLE I
Important notations and descriptions

Notation	Description
s, r, o	source entity, relation type, and object entity
$\mathbf{f_i}$	sentence embedding of the i-th event
$\mathbf{E_t}$, $\mathcal{G}_t$	event summary and event graph at time t
$\mathbf{C}$, $\mathbf{M}$	cache matrix and memory matrix
M, P	cache size and memory size
$\mathbf{c}_{t,(v)}$, $\mathbf{c}_{t,(r)}$	caches for entity v and event type r at time t
$\mathbf{m}_{t,(v)}$, $\mathbf{m}_{t,(r)}$	memories for entity v and event type r at time t
$\mathbf{h_v}'$, $\mathbf{o_r}'$	learnable embeddings for entity v and event type r
$\mathbf{H}^t$	the embedding matrix for all entities of $\mathcal{G}_t$
$\mathbf{h}_{\mathcal{G}_\mathbf{t}}$	graph representation of event graph $\mathcal{G}_t$

IV. Proposed Method

In this section, we present the overview of our proposed framework, **MTG**, as shown in Figure 2. Its components can be summarized as: 1) A cache module to learn *medium-term tendencies* from the past events and news data. 2) A memory module to learn *long-term statistics* from past events and cached memories. 3) A dynamic CompGCN module to capture *short-term triggers* through interactions among entities and the corresponding news data. Hidden signals learned through these three modules are combined to predict the occurrence of target event in the future.

A. Model Framework

1) Cache Module: The first component of **MTG** is an RNN-based cache module that keeps track of the evolution for both entities and relations from previous events and their corresponding text descriptions. This module is inspired by the memory module of Temporal Graph Networks (TGN) [17]. However we think it is more appropriate to name our module "cache" as we observe that it tends to forget long-term history owing to the nature of RNN. Also we can distinguish it with our own memory module that preserves long-term history, which is introduced in the next section.

To start, we create a cache matrix for all entities and all relations as $\mathbf{C} \in \mathbb{R}^{(N+K)\times M}$, where N is the number of entities, K is the number of relations (i.e., event types), and M is the memory size. The matrix is initialized with zeros. Let $E_t = \{e_1, e_2, ..., e_m\}$ be the list of events that happened at time t, where $e_i = (s_i, r_i, o_i, \mathbf{f}_i, t)$. For simplicity, we use $\mathbf{c}_{s_i}$ or $\mathbf{c}_{o_i}$ to denote the cache vectors for subject entity s_i and object entity o_i; we use $\mathbf{c}_{r_i}$ to denote the cache vector for relation r_i. Let $\mathcal{G}_t$ be the event graph built by the set of events E_t.

We create messages for entities and relations in each event e_i to save up-to-date information as follows. Theses messages are regarded as embedding inputs used in later temporal learning processes.

$$\mathbf{msg}_{t,(s_i)} = \underbrace{\mathbf{z}_{t,(s_i)} \parallel \mathbf{z}_{t,(o_i)}}_{\text{graph embeddings}} \parallel \underbrace{\mathbf{c}_{t-1,(r_i)}}_{\text{relation cache}} \parallel \underbrace{\mathbf{t}_t}_{\text{time embedding}} \tag{2}$$

$$\mathbf{msg}_{t,(o_i)} = \mathbf{z}_{t,(o_i)} \parallel \mathbf{z}_{t,(s_i)} \parallel \mathbf{c}_{t-1,(r_i)} \parallel \mathbf{t}_t \tag{3}$$

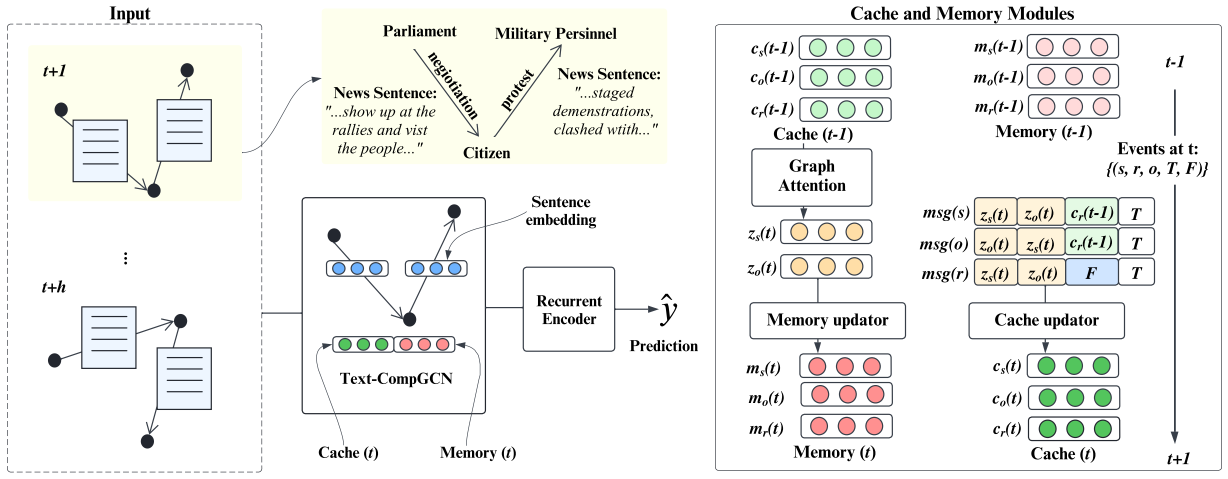


Fig. 2. An overview of **MTG**. The text-CompGCN aggregates the entity, event type, and news text features at timestamps $t+1$ to $t+h$. The features of entities and events types are learned from previous timestamps with the cache and memory modules: at timestamp t, it learns the entity embeddings with a temporal graph attention embedding model and construct messages, update the caches, and update the memories. A recurrent encoder is applied to model the temporal information for final prediction. The model is constructed end-to-end and all parameters are learned at the same time.

$$\mathbf{msg}_{t,(r_i)} = \mathbf{z}_{t,(s_i)} \parallel \mathbf{z}_{t,(o_i)} \parallel \tilde{\mathbf{f}}_i \parallel \mathbf{t}_t \tag{4}$$

$$\tilde{\mathbf{f}}_i = \mathbf{W}_s \mathbf{f}_i \tag{5}$$

$$\mathbf{z}_{t,(v)} = \mathbf{emb}(\mathcal{G}_t, v, t) \tag{6}$$

where $\mathbf{msg(s_i)_t}$ and $\mathbf{msg(o_i)_t}$ are the messages generated for subject s_i and object o_i, respectively, and $\mathbf{msg}_{t,(r_i)}$ is the message generated for the event type r_i. $\parallel$ is the concatenation operation. $\mathbf{c}_{t-1,(r_i)}$ is the memory of the event type r_i from the previous time step. $\mathbf{emb}()$ is a trainable graph embedding model consisting of L temporal graph attention layers (see Appendix A for detailed equations). $\mathbf{z}_{t,(s_i)}$ and $\mathbf{z}_{t,(o_i)}$ are learned embeddings of subject s_i and object o_i through the graph embedding model. We use the entity embedding $\mathbf{z}_{t,(s_i)}$ and $\mathbf{z}_{t,(o_i)}$ to construct the messages instead of using the caches $\mathbf{c}_{t-1,(s_i)}$ and $\mathbf{c}_{t-1,(o_i)}$ to alleviate the potential "staleness" problem [26], because the cache vectors that are updated a long time ago may no longer be present in the recent tendencies. We attach the relation cache $\mathbf{c}_{t-1,(r_i)}$ to the messages of entities so that the relational information between source and target entities can be included. $\mathbf{f}_i$ is the pretrained embedding for the news sentence that describes the current event e_i using BERT [30], [31]. $\mathbf{W_s}$ is a trainable weight that transforms f_i to the desired dimension in Equation 4, and $\mathbf{t}_t$ is the temporal embedding that encodes the temporal information of the current time step. If more than one message is generated for each entity or relation at the timestamp t, we just use the mean of the messages instead for further operations.

With the messages generated, the memory of the entities and relations are updated via Gated Recurrent Unit (GRU) [32] memory updaters:

$$\mathbf{c}_{t,(s_i)} = \mathrm{GRU}_e(\mathbf{c}_{t-1,(s_i)}, \mathbf{msg}_{t,(s_i)}) \tag{7}$$

$$\mathbf{c}_{t,(o_i)} = \mathrm{GRU}_e(\mathbf{c}_{t-1,(o_i)}, \mathbf{msg}_{t,(o_i)}) \tag{8}$$

$$\mathbf{c}_{t,(r_i)} = \mathrm{GRU}_r(\mathbf{c}_{t-1,(r_i)}, \mathbf{msg}_{t,(r_i)}), \tag{9}$$

where GRU can be replaced by other RNN models.

2) Long-term Memory Module: Despite its ability to keep track of recent trends of events, an RNN-based memory module only has short-term memory and is not able to memorize key information for a long time. However, we believe long-term information is an important statistical metric of past data. So we propose a long-term memory module that "forces" the model to remember the information from all previous events.

First we create a memory matrix for all entities and relations, $\mathbf{M} \in \mathbb{R}^{(N+K)\times P}$ and initialize these embedding vectors as zeros, where P is the embedding size. Specifically, $\mathbf{m}_v$ refers to the memory of an entity v (it can be either a subject or an object), and $\mathbf{c}_r$ refers to the memory of a relation r. At time step $t > 0$, the history of each entity v is updated when it appears in the current time stamp:

$$\mathbf{m}_{t,(v)} = (\mathbf{m}_{t-1,(v)} \cdot (t-1) + \mathbf{W_h}\mathbf{z}_{t,(v)})/t, \tag{10}$$

where $\cdot$ represents multiplication operation, and $\mathbf{W_h}$ is a trainable parameter. In this way, the memory for each entity can always keep the average over past time steps, and all past time steps contribute equally. By doing so, the module is able to remember information learned over a long history.

If an entity v does not appear in the current time, we update the history embedding (i.e., memory) as follows:

$$\mathbf{m}_{t,(v)} = (\mathbf{m}_{t-1,(v)} \cdot (t-1))/t. \tag{11}$$

By doing so, we "shrink" the memory embedding of the entity by a scaled factor $(t-1)/t$ such that the memories also record the frequency of the entity. It is intuitive that the memory embedding of a high-frequency entity should have larger values compared with less frequent entities. It is worth noting that when t is large, the memory embedding of an entity will become close to zero if this entity has not appeared for a long time. In such case, an alternative divider function such as $\log(t)$ could be applied instead of t. Similarly, we also update the history embedding for each event type (relation).

3) Dynamic Text-enhanced Knowledge Graph Learning: The cache and memory modules provide compact representations of entities and relation types learned from past events. However, they cannot provide us fine-grained information about events in a more recent history window (e.g., past few days). These events often play an important role in predicting the occurrence of target events. Thus, we introduce a module that encodes events in the last few days as dynamic knowledge graphs. We organize events that happened on each day as a discrete snapshot of a knowledge graph. Let the length of historical window be h days, such that there are h snapshots $\mathcal{G}_{t+1}, ..., \mathcal{G}_{t+h}$ with each of them modeling the events at time $t+i$ ($i \in \{1, 2...h\}$).

For each of the graph snapshots $\mathcal{G}_{t+i}$, we aim to learn the entities' embeddings in an integrated graph model through both their interactions/relations and the associated texts describing these interactions. Thus, we modify the aggregation and update functions in CompGCN [33] such that it can well combine text information during message passing processes. We name the modified version as Text-CompGCN. A two-layer Text-CompGCN model is applied to learn the embeddings of nodes and edges. At layer $l+1$, the embedding vector of a node v is learned as:

$$\mathbf{h}_v^{(l+1)} = f\Big(\sum_{(r,u,\mathbf{f}) \exists (v,r,u,\mathbf{f}) \in \mathcal{G}_{t+i}} \mathbf{W}_q^{(l)} \phi(\mathbf{h}_u^{(l)}, \mathbf{o}_r^{(l)}) \parallel \mathbf{W}_f^{(l)} \mathbf{f}\Big), \tag{12}$$

where $\mathbf{h}_u^{(l)}$ and $\mathbf{o}_r^{(l)}$ donate the features of node u and event type r learned at l-th layer, $\mathbf{W}_q^{(l)}$ is a trainable parameter for aggregating node and edge features at l-th layer, $\phi()$ represents a composition operation. $\mathbf{W}_f^{(l)}$ is a trainable parameter for aggregation text features. $f()$ is a activation function. The embedding vector of a event type r is learned as:

$$\mathbf{o}_r^{(l+1)} = \mathbf{W}_e^{(l)} \mathbf{o}_r^{(l)} \parallel \mathbf{W}_f^{(l)} \mathbf{f}, \tag{13}$$

where $\mathbf{W}_e^{(l)}$ is a trainable parameter for aggregating event type features at l-th layer. At the first layer, we initialize the embedding vectors as follows:

$$\mathbf{h}_v^{(0)} = \mathbf{c}_{t,(v)} \parallel \mathbf{m}_{t,(v)} \parallel \mathbf{h}'_v \tag{14}$$

$$\mathbf{o}_r^{(0)} = \mathbf{c}_{t,(r)} \parallel \mathbf{m}_{t,(r)} \parallel \mathbf{o}'_r, \tag{15}$$

where $\mathbf{c_{t,(v)}}$ and $\mathbf{c_{t,(r)}}$ are cache vectors for node v and event type r learned from previous timestamps, $\mathbf{m_{t,(v)}}$ and $\mathbf{m_{t,(r)}}$ are the corresponding memory vectors. $\mathbf{h}'_v$ and $\mathbf{o}'_r$ are learnable embeddings of node v and event type r. After the aggregation process, we obtain the embedding matrices for all entities as $\mathbf{H}^{t+i} \in \mathbb{R}^{|\mathcal{E}_{t+i}| \times d}$, where $\mathcal{E}_i$ is the set of entities appears in the snapshot $\mathcal{G}^{t+i}$; and we get a graph-level representation of the snapshot through a element-wise max pooling operation:

$$\mathbf{h}_{\mathcal{G}_{\mathbf{t+i}}} = \mathbf{maxpool}(\mathbf{H}_{t+i}) \tag{16}$$

After we get graph-level embeddings for all snapshots, a recurrent encoder is applied to learn through the temporal information and obtain a representation of the time series $t+1, ..., t+h$:

$$\mathbf{g}_{t+h} = \text{RNN}(\mathbf{h}_{\mathcal{G}_{\mathbf{t+h}}}, \mathbf{g}_{t+h-1}) \in \mathbb{R}^d, \tag{17}$$

Finally, we make a prediction for the target event at time $t+h+\Delta t$:

$$\hat{y} = \mathbb{P}(y_{t+h+\Delta t} | X_{1:t+h}) = \sigma(\mathbf{W}_\gamma \mathbf{g}_{t+h}) \in \mathbb{R}, \tag{18}$$

where $\mathbf{W}_\gamma \in \mathbb{R}^d$ is the parameter of the linear layer, and σ represents the sigmoid activation function. Since it is a binary classification problem, we choose Binary Cross Entrophy as the loss function:

$$-\sum_{i=0}^{n} \big(y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)\big), \tag{19}$$

where y_i is the ground truth of i-th sample.

Algorithm 1 The Proposed Method **MTG**

```
1:  Input: Event summaries $E_1, ...E_T$, temporal event graphs $\mathcal{G}_1, ...\mathcal{G}_T$, pre-trained news text embeddings
2:  Output: A trained binary classifier for predicting target events in the future: $f : (E_{1:t}; \mathcal{G}_{t+1:t+h}) \rightarrow \bar{y}_{t+h+\Delta t}$
3:  $b \leftarrow$ number of time steps for backpropagation
4:  $n \leftarrow (T-h)/b$
5:  for epoch = 0 to num_epochs do
6:      Initialize cache $\mathbf{C}^{(0)}$ and memory $\mathbf{M}^{(0)}$ matrix
7:      for i = 0 to n do
8:          initialize loss $l = 0$
9:          for j = 0 to $b$ do
10:             $t \leftarrow i \times b + j$
11:             With the event summaries $E_t$:
12:             Update $\mathbf{C}^{(t)}$ based on Eq. (2-9)
13:             Update the $\mathbf{M}^{(t)}$ based on Eq. (10-11)
14:             Construct feature vectors Eq. (14-15)
15:             With the event graphs $\mathcal{G}_{t+1}$,...,$\mathcal{G}_{t+h}$:
16:             $\mathbf{h}_{\mathcal{G}_{\mathbf{t+1}}}, ..., \mathbf{h}_{\mathcal{G}_{\mathbf{t+h}}} \leftarrow$ message passing
17:                         based on Eq. (12), (13), and (16)
18:             $g_{t+h} \leftarrow$ apply recurrent encoder with Eq. (17)
19:             Compute $\mathbb{P}(y_{t+h+\Delta t})$ with Eq. (18)
20:             $l \leftarrow l+$ loss computed based on Eq. (19)
21:         end for
22:         Backpropogate error and update model parameters
23:         Detach cache and memory matrix
24:     end for
25: end for
```

An overview of **MTG** is provided in Figure 2. We also present the training steps of **MTG** in Algorithm 1. In the training process, we first initialize the cache and memory

TABLE II
STATISTICS OF DATA. POSITIVE INDICATES THE RATIO OF POSITIVE SAMPLES WHERE PROTEST EVENTS OCCURRED. EVENTS, ENTITIES AND EVENT TYPES INDICATE TOTAL NUMBER OF EVENTS, ENTITIES AND EVENT TYPES IN EACH DATASET.

Dataset	Positive	Events	Entities	Event Types
Bangkok (Thailand)	40.1%	41,274	2,000	204
Cairo (Egypt)	62.5%	97,341	3,714	219
Moscow (Russia)	54.0%	217,834	5,833	233
New Delhi (India)	53.3%	95,222	3,245	213

matrix (line 6). At each timestamp t, we update the caches and memories with event summaries at t (line 11-13). We then construct feature vectors for entities and edge types with the updated caches and memories (line 14). Next, we aggregate event graphs with text features at $t+1$ to $t+h$ with the proposed text-CompGCN in a recurrent network. We make a prediction for the target event at $y_{t+h+\Delta t}$ (line 15 to 18) and compute the loss (line 19). For every b timestamps, we backpropogate the error and detach the memory and cache tensors from the loss computation for backpropogation to improve computational efficiency (line 21-22).

V. Experiment Setup

A. Dataset

The experiments are performed on four real-world datasets collected from the Integrated Conflict Early Warning System (ICEWS) [34]. ICEWS contains the political events and their associated news. Events are classified into 20 main categories and subcategories based on the Conflict and Mediation Event Observations (CAMEO) guidelines. For each event, the dataset records its source entity, target entity, event type, date, and a corresponding piece of news text to describe the details of the event. In this paper, we investigate four locations from different countries, including Bangkok (Thailand), Cairo (Egypt), Moscow (Russia), and New Delhi (India) from 2010-2016. The time granularity of each timestamp when constructing a sample is one day for all datasets. We train and evaluate **MTG** as well as other baseline models on these four datasets. The statistics of the datasets are presented in Table II.

B. Training Setup

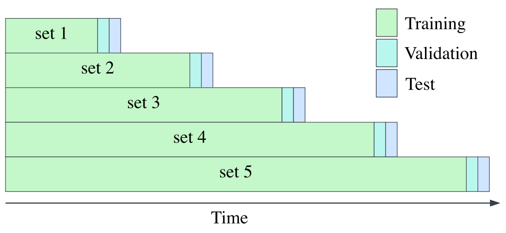


Fig. 3. A demonstration of the Walk-forward validation method.

Each dataset is split to training, validation, and test sets with a 80%, 10%, 10% ratio. A common method is to split the dataset by chronological order. However, observing the fact that the distribution of target events shifts over time, there are two major drawbacks for such split: first the information learned from training samples can be "outdated" due to the large gap between the training and validation/test samples. Second, the evaluation on only one set of test samples may not be comprehensive enough to demonstrate the overall performance of the model on different distributions. Inspired by Walk-forward [35] validation method, we choose five different sets of validation/test samples chronologically. A demonstration of the process is provided in Figure 3. We order the data samples by ascending time. We choose the first 20% data as the first set. Within this set, we chose the first 16% as training and the last 4% data as validation and test sets. Next, we move on to the second set which contains the first 40% of all data samples, similarly, we use the first 36% as training and the last 4% as validation/test. We continue this until we get to the last set (the full set). By doing so, we have a total of 20% samples for validation/test and we reduce the gap between training and validation/test samples so that the learned information is more up to date. Also it is a fairer evaluation as the total test set covers samples from different years. Another advantage of such an evaluation scheme is that eventually more samples are incorporated for training without sacrificing the amount of validation or test samples.

C. Comparation Methods

We compare **MTG** with several baseline models. Among the models, CompGCN+RNN and TGN use event KGs as inputs; DynamicGCN and T-GCN use word graphs; Glean uses both event KGs and word graphs; LSTM and TGN use long-term information.

- **Logistic Regression (LR)**. We utilize two different types of input for this model: $\mathbf{LR_{event}}$ uses accumulative counts of different event types in the short-term historical window as input features. $\mathbf{LR_{text}}$ uses the accumulative word counts of event descriptions (news).
- **Deep Neural Network (DNN)**. We build a DNN with three dense layers. Similar to LR, we use two different types of input for this model. $\mathbf{DNN_{event}}$ uses the same features as $\mathbf{LR_{event}}$ while $\mathbf{DNN_{text}}$ uses the same features as $\mathbf{LR_{text}}$. The sizes of the hidden layers are 256 and 64.
- **Long Short-term Memory (LSTM)** model uses the time series of previous timestamps that summarizes the counts of different event types of each day as input features. The size of the hidden layer is 100.
- **CompGCN+RNN** [33] is a temporal graph-based model that combines CompGCN with RNN. The feature size for entities and event types is 128.
- **Temporal Graph Network (TGN)** [17] makes predictions based on long-term information learned from past events with a continuous dynamic graph-based model. The feature size for entities and event types is 128.
- **DynmaicGCN** [6] makes predictions with dynamic word graphs constructed from news sentences in a short-term historical window. The word embedding size is 100.

- **Temporal Graph Convolutional Network (T-GCN)** [36] combines GCN with RNN. The same input features as DynamicGCN are used.
- **Glean** [7] makes predictions with both dynamic event knowledge graphs in a short-term window and text graphs built from the news texts. The feature size for entities and event types is 128.

D. Hyper-parameter setting

We utilize a pre-trained Bert model [30], [31] to generate a 384-dimensional sentence embedding for each piece of news text. For **MTG**, we set both cache size and memory size as 32, entity and relation embedding size as 64, and the size of node/edge features as 128. For CompGCN+RNN, Glean, and TGN models, we also set the size of node and edge features as 128. We set both the lead time and the prediction window as 1 day. For the methods considering short-term interactions of events, we set the history window as 7 days.

E. Evaluation Metrics

We use below metrics to evaluate the performance of **MTG**:

- **F1 score** is the harmonic mean of precision and recall.
- **BACC** stands for balanced accuracy score; it quantifies the sensitivity and specificity of the classifier and is often used when dealing with unbalanced datasets. The sensitivity, specificity, and BACC are defined as follows:

$$\text{sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{20}$$

$$\text{specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}, \tag{21}$$

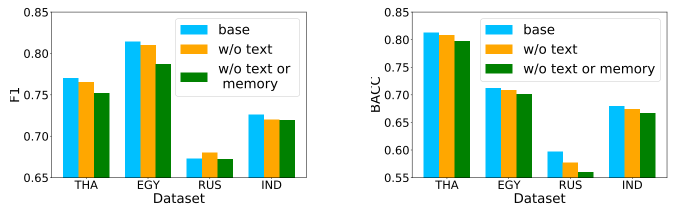
$$\text{BACC} = \frac{\text{sensitivity} + \text{specificity}}{2}, \tag{22}$$

where TP, TN, FP, and FN are true positive rate, true negative rate, false positive rate, and false negative rate.

VI. Experimental results

A. Prediction Performance

We evaluate the performance of **MTG** and other baseline models on four datasets. The overall prediction results are presented in Table III. We group the methods based on their inputs: only events, only texts, and both events and texts. The results show that **MTG** outperforms all the baseline methods in terms of F1 and BACC score across all datasets. Compared to the best baseline method, **MTG** achieves 3.0% and 2.6% of relative gain in terms of F1 and BACC on average. We observe that the prediction performance of the DynamicGCN model is not as good as the original paper [6]. We infer that this is caused by the different experimental setting: in our setting, first, we do not filter samples with less events as the original paper did. Second, for each set in the walk-forward validation, we split the training, validation, and test samples by chronological order; However, the original DynamicGCN paper shuffled all the samples and chose test samples randomly. In addition, Glean does not show its superiority as reported in the original paper [7] when compared with baselines such as T-GCN; we infer that this is due to the different datasets, prediction tasks, and experimental settings we use. We also observe that event-based models in general present more stable results compared with text-based models. The LR_{text} model performs poorly in the Egypt and Russia datasets, but outperforms most event-based methods in the Thailand dataset. We infer that rich semantic information extracted from news sentences can be potentially useful for predicting future events. Thus, we claim that it is beneficial to combine both event and text features in a unified model. That also explains why **MTG** that considers both events and text features achieves better performance. Moreover, we notice that **MTG** defeats the baselines that only consider short-term histories. Owing to the vanishing gradient problem, LSTM and TGN do not perform well even though they consider long-term histories. Thus we conclude that **MTG** achieves the best performance attributable to its ability of efficiently capturing and preserving long-term event and text information.



(a) F1 (b) BACC

Fig. 4. Ablation study on the text feature and the memory module.

B. Ablation Study

To examine the effectiveness of the components of **MTG** , we conduct an ablation study by: 1) removing the text features from **MTG** (w/o text); this is to explore the effectiveness of adding text features. 2) removing both text features and the memory module from **MTG** (w/o text or memory). Under this setting we set the cache size as 64 such that the feature size remains unchanged. This is to explore the effectiveness of the proposed memory module. The comparison results between the two variations and the original model (base) in terms of BACC and F1 scores are provided in Figure 4. The F1 score and BACC drop for all datasets when text and memory are both removed.

C. Model Complexity

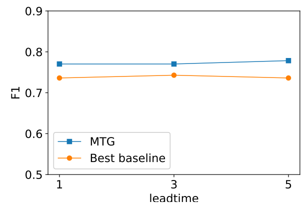
We compare the number of parameters of **MTG** and the baseline methods as Table IV. For all event graph-based models, we set the feature size for the entities and event types as 128; and for the text graph based-models, we set the dimension of word embeddings as 100. Compared with CompGCN+RNN and Glean, **MTG** has less parameters as it composes entity and event type features with caches and memories, which involve less trainable parameters.

TABLE III
EVENT FORECASTING RESULTS OF **MTG** VERSUS BASELINES ON TEST SETS. F1 AND BACC ARE EVALUATED OVER FIVE TEST SETS FOR EACH DATASET. % RELATIVE GAIN INDICATES THE GAIN OF **MTG** COMPARED TO THE BEST BASELINE.

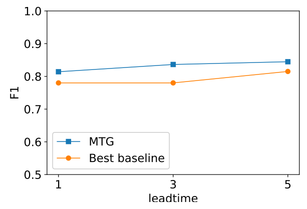
Input	**Method**	**Thailand**		**Egypt**		**Russia**		**India**	
		F1	BACC	F1	BACC	F1	BACC	F1	BACC
with events	LR_{event}	0.701	0.750	0.785	0.643	0.658	0.539	0.607	0.634
	$\text{DNN}_{\text{event}}$	0.691	0.750	0.765	0.672	0.591	0.541	0.677	0.647
	LSTM	0.688	0.753	0.787	0.700	0.580	0.481	0.574	0.561
	CompGCN+RNN	0.632	0.706	0.775	0.697	0.661	0.545	0.648	0.579
	TGN	0.667	0.745	0.748	0.683	0.639	0.508	0.700	0.678
with texts	LR_{text}	0.736	0.804	0.667	0.680	0.574	0.467	0.588	0.629
	DNN_{text}	0.736	0.804	0.707	0.701	0.627	0.534	0.564	0.623
	DynamicGCN	0.500	0.627	0.628	0.600	0.423	0.515	0.606	0.611
	T-GCN	0.718	0.781	0.634	0.663	0.515	0.464	0.589	0.624
with events+texts	Glean	0.658	0.714	0.780	0.701	0.670	0.556	0.598	0.588
	MTG	**0.770**	**0.813**	**0.814**	**0.712**	**0.673**	**0.597**	**0.726**	**0.679**
	% relative gain	4.6%	1.2%	3.4%	1.6%	0.5%	7.4%	3.7%	0.1%

TABLE IV
COMPARISON OF MODEL PARAMETERS ON THE THAILAND DATASET. FOR GLEAN, COMPGCN+RNN, TGN AND **MTG** , DIMENSION IS THE EMBEDDING DIMENSION OF ENTITIES AND EVENT TYPES. FOR DYNAMICGNN AND TGCN, DIMENSION IS THE SIZE OF WORD EMBEDDINGS.

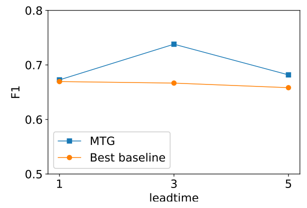
Model	**Dimension**	**#. Parameter**
LR_{event}	–	205
LR_{text}	–	31,079
$\text{DNN}_{\text{event}}$	–	68,993
DNN_{text}	–	7,972,737
LSTM	–	122,501
CompGCN+RNN	128	1,061,407
TGN	128	616,322
DynamicGCN	100	872,379
Glean	128	1,265,004
T-GCN	100	1,061,407
MTG	128	736,062



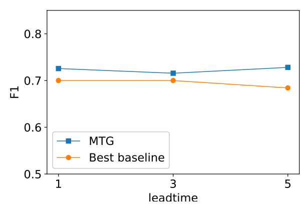
(a) Thailand



(b) Egypt



(c) Russia



(d) India

Fig. 5. Comparison between the F1 scores of **MTG** and the best-performed baseline for lead time = 1, 3, and 5.

D. Hyper-parameter Sensitivity

1) Lead time: We evaluate the sensitivity of **MTG** on different lead time values (1, 3, and 5 days). The performance of **MTG** in term of F1 score vs. the best performed baseline across all four countries are provided in Figure 5. The results show that **MTG** performs consistently better than the baseline methods. We also observe that the models tends to have better performance when the lead time is 3 or 5. This can be explained by the fact that there is often a delay between the triggers and the occurrence of events.

2) Feature size: We evaluate the sensitivity of the performance of **MTG** with different feature sizes of the entities and relations. The ratio between the components (as in Eq. (14) and Eq. (15)) of the features remains unchanged. We vary the feature size to 32, 64, 128 and 256. The performance of **MTG** in terms of F1 and BACC scores across four datasets are provided in Figure 6. We observe that in general the model tends to have a better performance with greater feature sizes. However, it becomes less sensitive at a certain point. For Thailand, Egypt and Russia, an increasing feature size can improve the performance significantly. For India, a feature size over 64 does not affect the performance remarkably.

3) Cache and memory sizes: We also investigate the sensitivity of **MTG** on different ratios between the cache and memories. We keep the sum of cache and memory size as 64, but change the memory size to 8, 16, 32, and 48. The performance of **MTG** in terms of F1 and BACC scores across four countries are provided in Figure 7. We observe that in general, the memory size does not affect the performance when it contributes less than half (32) of the total size; however, the model tends to perform worse when the it reaches 48.

E. Analysis of learned caches and memories

We visualize the evolution of learned cache and memory vectors through time for Thailand. The heatmaps of the caches and memories for both entities and the target relation type (protest) are shown in Figure 8; a deep color represents a higher value. Each row of the cache and memory heatmaps

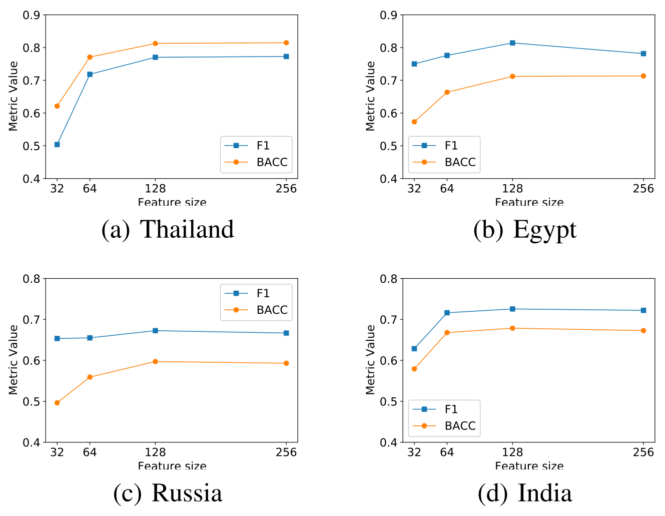


Fig. 6. Prediction results of **MTG** with varying feature sizes.

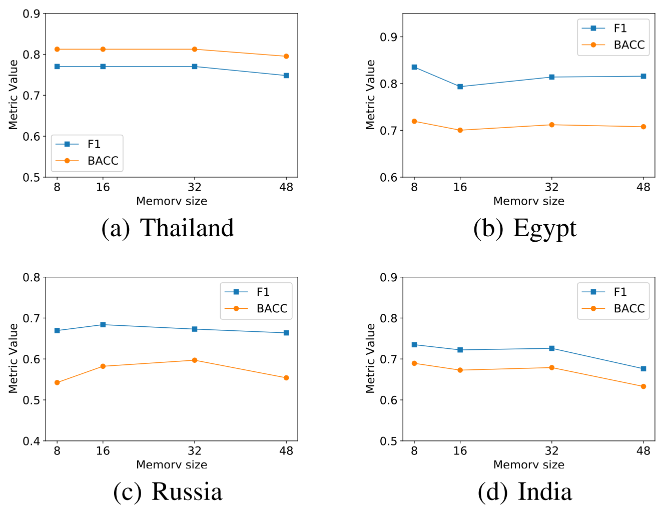


Fig. 7. Prediction results of **MTG** with varying memory sizes.

represents element-wise max-pooled values across all entities. We plot the caches and memories every 100 days for the first 1800 days, and we only present the first 20 dimensions of vectors for visualization. We observe that the heatmaps of memories are relatively stable and consistent through time. Conversely, the cache embeddings have gradual changes over time. This can be explained by the assumption that memories summarize previous information in a long-run; but caches capture recent tendencies and hence experience more oscillations. We also detect local maximums around 200-th to 300-th days on all four heatmaps, which correspond to the increasing number of protest events in that period of time.

VII. Broader Impacts

Our research focuses on the task of predicting societal events. To forecast such events prior to their occurrence can help decision-makers to take preventive strategies to minimize people's loss. By investigating patterns in historical data,

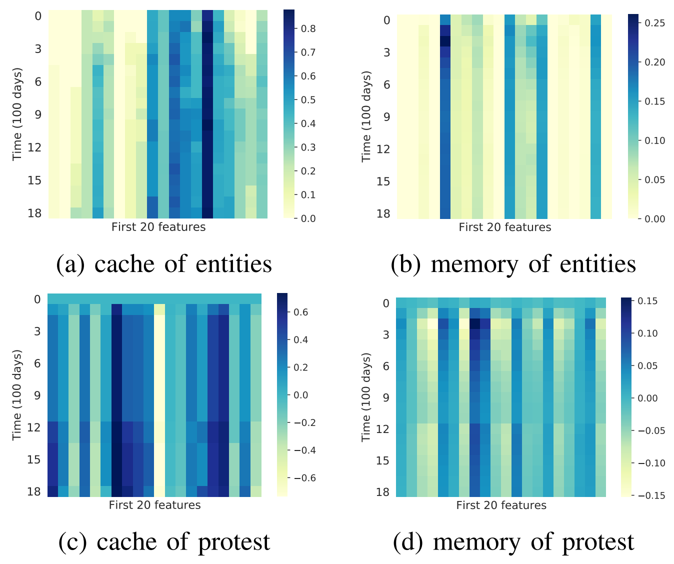


Fig. 8. Heatmap visualization of cache and memory vectors over time.

this work can also provide people a better understanding on the causes and impacts of events. Thus, this study is of significance for computational social science and human behavior analysis.

This work also presents new neural network based frameworks for temporal graph sequences. The proposed techniques can be easily adapted to other domains such as chemistry, finance, and social networks.

VIII. Conclusion

In this paper, we present a novel dynamic knowledge graph learning framework for event prediction. Our proposed model simultaneously considers long-term, medium-term, and short-term historical information. We also present a new end-to-end approach to integrate text information in event graphs coherently. We demonstrate the effectiveness of the proposed method on real-world datasets, and perform sensitivity analyses and ablation studies under different settings.

In the future, we plan to investigate different types of temporal graphs in various applications, such as social networks or product review networks. Moreover, we will explore new methods that model the evolution of multimodal data and their dependencies at the same time.

Acknowledgement

This work is supported in part by the US National Science Foundation under grants 1948432 and 2047843.

References

[1] F. Xia, K. Sun, S. Yu, A. Aziz, L. Wan, S. Pan, and H. Liu, "Graph learning: A survey," *IEEE Transactions on Artificial Intelligence*, vol. 2, no. 2, pp. 109–127, 2021.
[2] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020.
[3] S. Min, Z. Gao, J. Peng, L. Wang, K. Qin, and B. Fang, "Stgsn — a spatial–temporal graph neural network framework for time-evolving social networks," *Knowledge-Based Systems*, vol. 214, p. 106746, 2021.

[4] B. Abu-Salih, "Domain-specific knowledge graphs: A survey," *Journal of Network and Computer Applications*, vol. 185, p. 103076, 2021.

[5] W. Chen, M. Jiang, W.-G. Zhang, and Z. Chen, "A novel graph convolutional feature based convolutional neural network for stock trend prediction," *Information Sciences*, vol. 556, pp. 67–94, 2021.

[6] S. Deng, H. Rangwala, and Y. Ning, "Learning dynamic context graphs for predicting social events," in *Proceedings of the 25th ACM SIGKDD*, ser. 9. New York, NY, USA: Association for Computing Machinery, 2019, p. 1007–1016.

[7] ——, "Dynamic knowledge graph based multi-event forecasting," in *Proceedings of the 26th ACM SIGKDD*, 2020, p. 1585–1595.

[8] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," 2017.

[9] H. Wang, J. Wang, J. Wang, M. Zhao, W. Zhang, F. Zhang, X. Xie, and M. Guo, "Graphgan: Graph representation learning with generative adversarial nets," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018.

[10] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[11] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017.

[12] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *The Semantic Web*, A. Gangemi, R. Navigli, M.-E. Vidal, P. Hitzler, R. Troncy, L. Hollink, A. Tordai, and M. Alam, Eds. Cham: Springer International Publishing, 2018, pp. 593–607.

[13] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," 2017.

[14] N. N. Daud, S. H. Ab Hamid, M. Saadoon, F. Sahran, and N. B. Anuar, "Applications of link prediction in social networks: A review," *Journal of Network and Computer Applications*, vol. 166, p. 102716, 2020.

[15] A. K. Singh and L. Kailasam, "Link prediction-based influence maximization in online social networks," *Neurocomputing*, vol. 453, pp. 151–163, 2021.

[16] L. Ou-Yang, D.-Q. Dai, X.-L. Li, M. Wu, X.-F. Zhang, and P. Yang, "Detecting temporal protein complexes from dynamic protein-protein interaction networks," *BMC Bioinformatics*, vol. 15, p. 335, 2014.

[17] E. Rossi, B. Chamberlain, F. Frasca, D. Eynard, F. Monti, and M. Bronstein, "Temporal graph networks for deep learning on dynamic graphs," 2020.

[18] F. Manessi, A. Rozza, and M. Manzo, "Dynamic graph convolutional networks," *Pattern Recognition*, vol. 97, p. 107000, 2020.

[19] P. Goyal, S. R. Chhetri, and A. Canedo, "dyngraph2vec: Capturing network dynamics using dynamic graph representation learning," *Knowledge-Based Systems*, vol. 187, p. 104816, 2020.

[20] M. Xu, "Understanding graph embedding methods and their applications," 12 2020.

[21] S. M. Kazemi, R. Goel, K. Jain, I. Kobyzev, A. Sethi, P. Forsyth, and P. Poupart, "Representation learning for dynamic graphs: A survey," *Journal of Machine Learning Research*, vol. 21, pp. 1–73, 2020.

[22] A. Sherstinsky, "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network," *CoRR*, vol. abs/1808.03314, 2018.

[23] R. Trivedi, H. Dai, Y. Wang, and L. Song, "Know-evolve: Deep temporal reasoning for dynamic knowledge graphs," 2017.

[24] H. Dai, Y. Wang, R. Trivedi, and L. Song, "Deep coevolutionary network: Embedding user and item features for recommendation," 9 2016.

[25] L. Qu, H. Zhu, Q. Duan, and Y. Shi, "Continuous-time link prediction via temporal dependent graph neural network," *The Web Conference 2020 - Proceedings of the World Wide Web Conference, WWW 2020*, pp. 3026–3032, 4 2020.

[26] S. M. Kazemi, R. Goel, S. Eghbali, J. Ramanan, J. Sahota, S. Thakur, S. Wu, C. Smyth, P. Poupart, and M. Brubaker, "Time2vec: Learning a vector representation of time," 2019.

[27] F. Qiao, P. Li, X. Zhang, Z. Ding, J. Cheng, and H. Wang, "Predicting social unrest events with hidden markov models using gdelt," *Discrete Dynamics in Nature and Society*, vol. 2017, 2017.

[28] B. Cortez, B. Carrera, Y. J. Kim, and J. Y. Jung, "An architecture for emergency event prediction using lstm recurrent neural networks," *Expert Systems with Applications*, vol. 97, pp. 315–324, 5 2018.

[29] X. Wu, C. Huang, C. Zhang, and N. V. Chawla, *Hierarchically Structured Transformer Networks for Fine-Grained Spatial Event Forecasting*. New York, NY, USA: Association for Computing Machinery, 2020, p. 2320–2330.

[30] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," *CoRR*, vol. abs/1908.10084, 2019.

[31] "Sentencetransformers documentation (https://www.sbert.net/index.html)."

[32] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," 2014.

[33] S. Vashishth, S. Sanyal, V. Nitin, and P. P. Talukdar, "Composition-based multi-relational graph convolutional networks," *ArXiv*, vol. abs/1911.03082, 2020.

[34] "Icews coded event data - integrated crisis early warning system (icews) dataverse."

[35] M. Y. Hu, G. P. Zhang, C. X. Jiang, and B. E. Patuwo, "A cross-validation analysis of neural network out-of-sample performance in exchange rate forecasting," *Decision Sciences*, vol. 30, no. 1, pp. 197–216, 1999.

[36] L. Zhao, Y. Song, C. Zhang, Y. Liu, P. Wang, T. Lin, M. Deng, and H. Li, "T-gcn: A temporal graph convolutional network for traffic prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 9, pp. 3848–3858, 2020.

[37] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

Appendix A
Supplemental equations

A. Temporal Graph Attention

We use an L-layer temporal graph attention module [17], [20] to aggregate the graph $\mathcal{G}$ and computes the node embedding for v.

$$z_{t,(v)} = \mathbf{emb}(\mathcal{G}_t, v, t) = h^{(L)}_{t,(v)}, \tag{23}$$

where $h^{(L)}_{t,(v)}$ is the learned embedding for v at L-th layer. At l-th layer:

$$h^{(l)}_{t,(v)} = \mathbf{MLP}^{(l)}(h^{(l-1)}_{t,(v)} \parallel \tilde{h}^{l}_{v}), \tag{24}$$

$$\tilde{\mathbf{h}}^{(l)}_{t,(v)} = \mathbf{MultiHeadAttention}^{(l)}(q^{(l)}_t, K^{(l)}_t, V^{(l)}_t), \tag{25}$$

$$q^{(l)t} = h^{(l-1)}_{t,(v)}, \tag{26}$$

$$K^{(l)}_t = [h^{(l)}_{t,(u)} \parallel \mathbf{c}_{t-1,(r)} \quad \mathbf{for} \quad (r,u) \exists (v,r,u) \in \mathcal{G}_t], \tag{27}$$

$$K^{(l)}_t = V^{(l)}_t, \tag{28}$$

where $q^{(l)}_t$, $K^{(l)}_t$, and $V^{(l)}_t$ are query, keys, and values for the multi-head attention [37]. The node embedding is initialized as the caches in the previous timestamp:

$$h^{(0)}_{t,(v)} = \mathbf{c}_{t-1,(v)}. \tag{29}$$