# Learning Dynamic Context Graphs for Predicting Social Events

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

Event forecasting with an aim at modeling contextual information is an important task for applications such as automated analysis generation and resource allocation. Captured contextual information for an event of interest can aid human analysts in understanding the factors associated with that event. However, capturing contextual information within event forecasting is challenging due to several factors: (i) uncertainty of context structure and formulation, (ii) high dimensional features, and (iii) adaptation of features over time. Recently, graph representations have demonstrated success in applications such as traffic forecasting, social influence prediction, and visual question answering systems. In this paper, we study graph representations in modeling social events to identify dynamic properties of event contexts as social indicators.

Inspired by graph neural networks, we propose a novel graph convolutional network for predicting future events (e.g., civil unrest movements). We extract and learn graph representations from historical/prior event documents. By employing the hidden word graph features, our proposed model predicts the occurrence of future events and identifies sequences of dynamic graphs as event context. Experimental results on multiple real-world data sets show that the proposed method is competitive against various state-of-the-art methods for social event prediction.

## CCS CONCEPTS

• **Information systems** → **Data mining**; • **Applied computing** → **Forecasting**; • **Computing methodologies** → *Supervised learning by classification.*

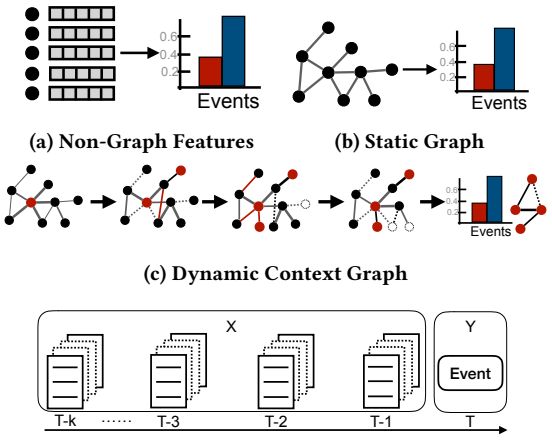
## KEYWORDS

Event Prediction; Dynamic Graph Convolutional Network; Temporal Encoding

**ACM Reference Format:**
Songgaojun Deng, Huzefa Rangwala, and Yue Ning. 2019. Learning Dynamic Context Graphs for Predicting Social Events. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3292500.3330919


*KDD '19, August 4–8, 2019, Anchorage, AK, USA*

ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330919




**(a) Non-Graph Features** **(b) Static Graph**

**(c) Dynamic Context Graph**

**(d) Raw Input Data**

**Figure 1: A motivating example of learning dynamic context graphs for event forecasting. Given a raw input data X ( 1d) and its target value y, existing models (1a) focus on semantic representations; New graph models (1b) ignore the temporal evolution of graphs in a sequence of inputs; Dynamic context graph models (1c) encode both temporal information and semantic embeddings in event modeling.**

## 1 INTRODUCTION

Societal uprisings are due to gradually changing interactions between entities including citizens, organizations and governments of countries. Forecasting social events from large, noisy and heterogeneous open source sensors (media) is a challenging problem. Prior work [19, 30] in this area seeks to not only predict the onset of a societal event but also identify related events or information (termed as *precursors*) that lead to the particular event. For example, a country's new economic policy may affect people's buying behavior and the stock market, and may also lead to civil unrest. Identifying the changes in economic policy and changes in consumer behaviors could lead to predicting and understanding of the civil unrest event.

Prior event forecasting approaches mainly focus on predicting the events of interests and do not leverage the underlying dynamic context from related events in the past. The task of learning corresponding event context while forecasting events presents many challenges:

- Traditional event forecasting relies on feature selection and engineering methods within the machine learning pipelines. These approaches seek to identify a subset of highly discriminant features that are capable of predicting events of interest in the future. Both static (such as textual and spatial information) and dynamic

features (network and time-based) have been studied [33]. However, the increasing scale of data and evolving concepts renders existing feature engineering models inaccurate and inefficient.
- Existing models on event forecasting usually consider a culmination or summation of features in a given time period. This ignores the effect of temporal dependencies across different events on a future event.
- From the perspective of intelligence analysts, information overload is overwhelming. Presenting succinct representations of events and their precursors in the form of summaries is an unmet need.

We address the above challenges by proposing a dynamic graph convolution network. It encodes temporal information in dynamic graph structures to predict events of interest in the future and generates graphs of concepts (entities and actions) representing event-related information. As shown in Figure 1, bag-of-words and distributed representations [15, 33] are focused on semantic representations of words, sentences, or documents. Models based on these representations are able to achieve good forecasting accuracy but are inadequate to provide a succinct summary or explanation for events. The graph convolutional model [11] employs an expressive graph representation of nodes and their connections but ignores the dynamic context associated with the data. In this paper, we study the dynamic graph representations of event context in event forecasting tasks. Our contributions are summarized as below:

- We develop a novel graph-based neural network for predicting events of interest and identifying key context graphs in order to understand their progression. Previous work has either focused on improving forecasting performance [33] or document level precursor identification [19]. In this paper, we present a novel graph representation for summarizing event context using graph structure while forecasting events.
- Dynamic graph structures over time are discovered in event forecasting tasks for understanding and analyzing the events' context. Instead of static features, we design a mechanism that encodes the dynamic graph structure of words from past input documents to forecast the events of interest in the future.
- In order to make a prediction, multiple samples of historical data (e.g days of tweets/news) are collected and processed. In this case, pre-trained semantic features usually can not reflect contextual changes over time. We propose a temporal encoded feature module to alleviate this problem. It takes into account both the semantic embeddings and the hidden graph embeddings from previous time periods.

We evaluate the proposed method with other state-of-the-art models on real-world event datasets. With quantitative and qualitative experiments, we demonstrate the strengths of the dynamic graph convolution network in predicting events in the future and producing event context graphs from the history.

## 2 RELATED WORK

We review feature selection methods for spatio-temporal event modeling and various models for event forecasting. We also summarize some recent work on graph convolutional networks.

### 2.1 Feature Selection in Event Modeling

Both supervised and unsupervised techniques have been explored to extract text information of interest and to examine multiple types of features such as spatial [24], temporal [30], or spatio-temporal burstiness [12]. However, static semantic features are not able to capture the dynamic trends over time in social events effectively. In our proposed method, dynamic input is processed as different examples arrive. It does not require that all the examples lie in the same static feature space.

### 2.2 Event Forecasting

Many event forecasting approaches with spatio-temporal correlations have been applied to predictions of elections [21, 27], stock market movements [3], disease outbreaks [1, 26], and crimes [29]. Linear regression models have been developed to deal with simple features, such as tweet volumes to predict the occurrence time of future events [3]. More sophisticated features such as topic related keywords [29] have also been studied. Zhao et al. [33] studied static features derived from a predefined vocabulary by domain experts and dynamic features generated from dynamic query expansion in a multi-task feature learning framework. A multi-task learning model [20] is proposed with dynamic graph constraints within a multi-instance learning framework for precursor discovery and event forecasting. These prior models do not provide an approach to reveal hidden contextual information among entities. Our proposed model is learned based on graph structures of words. This gives us the benefit of discovering the impact of hidden connections among words for forecasting future events.

### 2.3 Graph Neural Networks

In the past few years, many studies have adopted deep neural networks to adapt to arbitrary graph-structured data [11]. Graph convolutional networks utilize the adjacency matrix or the Laplacian matrix to depict the structure of a graph and capture spatial features between the nodes. Previously, Kipf and Welling presented a simplified layer-wise graph neural network model (GCN) [11]. It achieves state-of-the-art classification results on a number of graph benchmark datasets[11, 23]. Graph Attention Networks (GAT) [28] incorporated the attention mechanism into GCN to learn personalized weights for nodes from their neighbors. For natural language processing tasks, GCN has been successfully applied in semantic role labeling [14] to encode syntactic structure of sentences, text classification [11, 22, 31], and event detection based on syntax [18].

Even though graph convolutional models are able to capture spatial features among the nodes, temporal information is not easy to be taken into consideration. Given self-circulation mechanisms of Recurrent neural networks (RNN) and their variants [5, 8], Seo *et al.* [25] combined graph convolutional networks and recurrent neural networks to predict structured sequences of data. They replaced operations on sequence data by a graph convolution. Compared to the combinational models, we present an elegant solution to encode temporal information and handle dynamic graph features for predicting events.

## 3 METHODOLOGY

In this section, we present the main components of our approach to model dynamic context graphs for predicting social events. Overall, the key objectives of our proposed model are 1) to capture abstract contextual graphs for event explanation; and 2) to forecast the occurrence of a specific type of events given a historical input $X$ of event-related articles. The articles are not pre-categorized and may report one or multiple events. We first encode the input data into a sequence of graphs with node embeddings. Then we develop a graph convolutional network model based on the sequence of graphs to predict the occurrence ($y$) of a certain type of events.

$$X \xrightarrow{\text{encode}} \text{Graphs} \xrightarrow{\text{model}} y$$

Given a city $c$, in order to formulate one training instance, we collect published articles for $k$ consecutive days prior to a date of interest, $t$ as the raw input ($x_{c,t} = \{\text{docs in days } t-1 \rightarrow t-k\}$). Given the occurrence of a target event on day $t$, we annotate this sample as a positive sample ($y_{c,t} = 1$); Otherwise, if there is no event on the day $t$ as well as the previous three days, we label this instance as a negative sample ($y_{c,t} = 0$). Then we construct a corresponding graph from the raw input $x_{c,t}$ where each node is a word. Instead of considering all historical articles together, we build separate graphs for different days. Figure 2 provides an overview of the proposed approach.

### 3.1 Encoding

To predict future events based on text data and to capture dynamic graph-based context of the event, we first transform our text-based input into a graph representation.

**Dynamic Graph.** For one sample, given its collection of past historical articles, we extract $n$ number of keywords by eliminating the very common words and extremely rare words. We create multiple word relation graphs represented by a sequence of adjacency matrices $[A_{t-k}, ..., A_{t-1}]$, where $A_k \in \mathbb{R}^{n \times n}$ for each time step (day). For each graph, the nodes are words and the dimensions of the adjacency matrix are the total number of unique words. The edges are based on word co-occurrence in the collection of documents. To calculate weights between two words, we employ document-based point-wise mutual information (PMI) [6], which is a popular measure for word associations. The edge weight between node $i$ and node $j$ at time $t$ is defined as:

$$A_t[i,j] = \begin{cases} \text{PMI}_t(i,j) & \text{PMI}_t(i,j) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

For each graph, we only consider the articles in the exact time period exclusively. Therefore, the PMI value of a word pair $i$, $j$ at time $t$ is computed as:

$$\text{PMI}_t(i,j) = \log \frac{d(i,j)}{d(i)d(j)/D} \tag{2}$$

where $d(i,j)$ is the total number of articles where both $i$ and $j$ appeared at current time $t$. $d(i)$ and $d(j)$ are the total number of articles in the collection containing at least one occurrence of $i$ and $j$ respectively. $D$ is the total number of articles in the collection. Generally speaking, positive PMI values imply high semantic relevance of words in the corpus. Thus, we only add weighted edges between pairs of words with positive PMI values.

**Feature Representation** We model each word $i$ in the graph as a real-valued vector $h_i \in \mathbb{R}^F$ using the word embedding vector of word $i$ as the initial feature. This real-valued vector captures the hidden syntactic and semantic properties of the word [2]. Word embeddings are pre-trained on a large unlabeled corpus like the Wikipedia database [15]. Since we build the graphs at different times, we do not use hand-crafted chronological features of words.

### 3.2 Model Framework

Given the constructed graphs and the pre-trained word representations, we introduce our proposed model, Dynamic Graph Convolutional Network (DynamicGCN). We show how we extract the learned key patterns of graphs while making event predictions. As shown in Figure 2, our proposed model consists of input layers, graph convolutional layers, temporal encoding layers, a masked nonlinear transformation layer, and an output layer.

**Input Layer** The units at the input layer take dynamic graphs which are represented as a sequence of encoded adjacency matrices $[A_{t-k}, ..., A_{t-1}]$. The graphs are built from the raw input data. Each node in the graph is encoded with an word embedding vector. The edge weight between two nodes is calculated by their PMI.

**Static GCN Based Network Encoding** Graph Convolutional Networks (GCN) [11] operate directly on a graph and induce node feature vectors from the properties of their neighborhoods. The model can be built by stacking multiple convolutional layers to involve information from farther neighbors.

Formally, given an undirected graph $G = (V, E)$ where $V(|V| = n)$ and $E$ are sets of nodes and edges. Let $H \in \mathbb{R}^{n \times F}$ be a matrix containing features of all $n$ nodes, where $F$ is the dimension of the feature vectors, and row $h_v \in \mathbb{R}^F$ is the feature vector for node $v$. Given the adjacency matrix $A \in \mathbb{R}^{n \times n}$, a GCN layer is a nonlinear transformation that maps from $H$ to $H^{(1)} \in \mathbb{R}^{n \times F'}$, defined as:

$$H^{(1)} = g(\hat{A}HW^{(0)} + b^{(0)}) \tag{3}$$

$W^{(0)} \in \mathbb{R}^{F \times F'}, b^{(0)} \in \mathbb{R}^{F'}$ are model parameters. $g$ is a non-linear activation function. $F'$ is the output feature dimension. $\hat{A}$ is the normalized symmetric adjacency matrix which is defined as:

$$\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}} \tag{4}$$

Here, $\tilde{A} = A + I_N$ and $\tilde{D}$ is the degree matrix. $I_N$ is an identity matrix with dimensions of $N$ and $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$.

GCN models have been applied to capture the spatial features between the connected nodes. With multiple convolutional layers, the model is described as:

$$H^{(l+1)} = g(\hat{A}H^{(l)}W^{(l)} + b^{(l)}) \tag{5}$$

where $l$ denotes the layer number, $\hat{A}$ is the predefined adjacency matrix, and $H^{(l)}$ is the node embedding matrix passing by layers. The intuition is that at each layer, nodes aggregate information from their local neighbors. At deeper layers, the nodes indirectly receive more information from farther nodes in the graph.

**Dynamic GCN Based Network Encoding** In our model, the adjacency matrix differs at each time step as the word co-occurrence graph changes. Compared to the static model introduced above, the dynamic GCN layer processes both the adjacency matrix and the feature map at time $t$. At each time, the graph node learns a vector of representation by using the information of its local neighbors within the current time epoch. The embeddings of the neighbors are updated from the previous epoch. Given different graphical structures at different times, temporal dependency is passed through

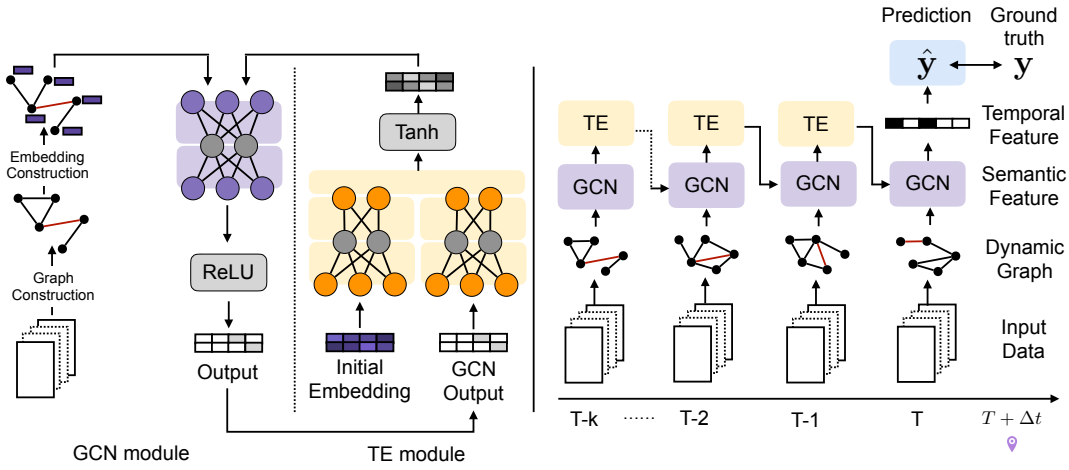



**Figure 2: System Framework of the Dynamic Graph Convolutional Network. Input data consists of event related articles ordered by time. We construct dynamic graphs based on input data and feed them into the GCN layers by time. For each GCN layer except the first one, the input features are processed by a Temporal Encoded (TE) module, involving the output of the last GCN layer and the current word embeddings, to capture temporal features. We add a masked nonlinear transformation layer to unify the final output vector from the final GCN layer. The loss is calculated between the model output and ground truth.**

each GCN layer to the graph nodes. The new convolutional layer is defined as:

$$H_{t+1} = g(\hat{A}_t \tilde{H}_t W^{(t)} + b^{(t)}) \tag{6}$$

where $\hat{A}_t \in \mathbb{R}^{n \times n}$ is the normalized symmetric adjacency matrix at time $t$ as defined in Eq. 4. $W^{(t)} \in \mathbb{R}^{F^{(t)} \times F^{(t+1)}}$ and $b^{(t)} \in \mathbb{R}^{F^{(t+1)}}$ are model parameters of the GCN layer at time step $t$. Notice that the $\tilde{H}_t$ is not the output of the last GCN layer, but the temporal encoded (TE) embeddings calculated from the last TE layer.

**Temporal Encoded Features (TE)** At $t = 0$, the node features $H_0 \in \mathbb{R}^{n \times F^{(0)}}$ are the pre-trained semantic embeddings. The GCN layer at this time learns the node representation by involving the semantic information from the connected nodes that is similar to the standard learning procedure of the GCN layer. Inspired by gate units in recurrent neural networks, we process the input feature of GCN layer by considering the information at the last moment. For $t > 0$ in the past time window, we employ a temporal encoded layer to re-encode the feature, including both the semantic information and the learned GCN features of each node. In order to combine the GCN encoded features at each time, as well as the initial word embedding features, we employ a learnable linear transformation on each of them. Thus, the initial step, two linear transformations, parametrized by two weight matrices, are applied to $H_t$ and $H_0$, respectively. Then we apply concatenation on the two transformed features and pass the result to a tanh function.

The encoding procedures are defined as below:

$$H_p^{(t)} = H_t W_p^{(t)} + b_p^{(t)} \tag{7}$$

$$H_e^{(t)} = H_0 W_e^{(t)} + b_e^{(t)} \tag{8}$$

$$\tilde{H}_t = \tanh([H_p^{(t)} \parallel H_e^{(t)}]) \tag{9}$$

where $W_p^{(t)} \in \mathbb{R}^{F^{(t)} \times \alpha}, W_e^{(t)} \in \mathbb{R}^{F \times (F^{(t)} - \alpha)}$, and $0 \leq \alpha \leq F^{(t)}$ ensuring the feature dimensions between GCN layers match. $F$ is the dimension of the initial word embedding features. $\parallel$ represents the concatenation operation. Tanh activation is used to regulate the values flowing through the network, and $\tilde{H}_t$ is the temporal encoded embeddings applied at the GCN layer at time $t$.

**Masked Nonlinear Transformation Layer** By stacking the dynamic GCN layers, we obtain a scalar feature representation for each node in the graph by setting the output feature dimension of the last layer to 1. Due to dynamic graph encoding, the output feature vector of the last GCN layer is a combined representation of all graph nodes, which is different for instances with different graph nodes. To guarantee the consistency of our model, we employ a masked nonlinear transformation layer to map the final output vector to the prediction of the task. First, we pass $H_T \in \mathbb{R}^{n \times 1}$, GCN encoded feature vectors at time $T$ (the last time state), through a masked zero padding layer to obtain a masked feature vector. The length of the vector is the size of the vocabulary $N_v$, which varies in different datasets. We then apply a nonlinear transformation to the incoming data. Here, the transformations are defined as:

$$z_T = \text{zero\_padding}(H_T^\intercal) \tag{10}$$

$$\hat{y} = \sigma(z_T w_m^\intercal + b_m) \tag{11}$$

where $H_T^\intercal$ is the transposed row vector of the last GCN output and $z_T \in \mathbb{R}^{N_v}$ is the masked transformation. $z_T$ has the same dimension for all samples in one dataset. $w_m \in \mathbb{R}^{N_v}$ and $b_m \in \mathbb{R}$ are learnable weights and $\hat{y}$ is the prediction of the occurrence of a future event. $\sigma$ is the sigmoid function.

**Optimization** Finally, we compare the prediction value with the ground truth and optimize the binary cross entropy loss:

$$\mathcal{L} = -\sum y \ln \hat{y} \tag{12}$$

where $y$ is the ground truth and $\hat{y}$ is the model prediction. All model parameters can be trained via back-propagation and optimized using the Adam algorithm [10] given its efficiency and ability to avoid overfitting.

### 3.3 Vocabulary Sampling

The continuous evolution of language makes it difficult to identify the important words over a long period of time. Training processes with long-term examples require node flexibility when building word relation graphs. However, with the rapid development of social media, the usage of words tend to have short-term stability. The dimensions of GCN model parameters are independent of the number of nodes. This allows us to sample different nodes for different examples. In our design, we select a certain number of keywords in a short time and build graph representations based on these nodes. As training samples overlap, the proposed model is able to capture the trends of words. The proposed model is based on dynamic graphs with changing words to ensure validation over time. If the vocabulary is out of date, we can enlarge with new words or delete outdated words. As such, the vocabulary can be updated to accommodate changes in the input data.

### 3.4 Context Graph Generation

Context graph is extracted from the input dynamic graphs given the learned model parameters. To generate a context graph, we first extract the important nodes from the trained model and then construct the subgraph of the input dynamic graph.

**Extract Nodes** Given our trained model, we first collect the graph-structured instances of an event, with a sequence of the adjacency matrix as the input. We obtain the scalar value $h_{i,t}$ from the last GCN layer and $w_{i,m}$ from the masked nonlinear transformation layer of each word node $i$. Based on the two scalar values, we calculate an indicator for each word node as:

$$I_i = h_{i,t} \times w_{i,m}$$

where $\sum I_i$ is the non-bias data of the sigmoid function as seen in Eq. 11. The word indicator correlates with event prediction and leads to identification of key words associated with an event. We analyzed the $I$ distribution of multiple instances and notice that it follows a Gaussian distribution $\mathcal{N}(\mu \approx 0, \sigma^2)$. We set the threshold range $(\mu, \mu + 2\sigma]$ for sampling nodes to represent the dynamic contextual graphs associated with the target events.

**Subgraph Construction** Based on these nodes, we then extract the isomorphic subgraph of the input dynamic graph at each time, respectively. The weight of the edge in the subgraph is the sum of the indicator values of the two endpoint nodes.

## 4 EXPERIMENT SETUP

In this section we evaluate our proposed model on the event forecasting tasks. Specifically, we want to answer the following key questions: 1) whether our model is able to achieve satisfactory forecasting results compared to other baseline approaches; and 2) can it recognize the key graph patterns of the task and/or the evolution of the graphs by time?

### 4.1 Datasets

The experimental evaluation was performed on the event data from Integrated Conflict Early Warning System (ICEWS) [4]. It contains political events with the goal of evaluating national and international crisis events. These events are encoded with 20 major categories and their subcategories. Each event has been encoded with geolocation (city, state, country), time (day, month, year), category, and its associated text. In this paper we focus on one major category of events, protest, and select datasets from four countries.

**Table 1: Dataset Statistics.**

| dataset | #documents | #vocabulary | #sample | #pos | #neg |
|---|---|---|---|---|---|
| India | 111,653 | 75,994 | 12,249 | 4,586 | 7,663 |
| Egypt | 30,867 | 19,680 | 3,788 | 1,469 | 2,319 |
| Thailand | 19,410 | 27,281 | 1,883 | 715 | 1,168 |
| Russia | 85,527 | 49,776 | 3,552 | 1,171 | 2,381 |

**India** Protest events in India mainly consist of demonstrations, rallies, strikes, and passage obstructions. We focus on data from up to 15 cities every year from 2012 to 2016, including New Delhi, Delhi, Calcutta, Hyderabad, and others.

**Egypt** The number of events varies from year to year in Egypt. The main types of protests in this country are rallies and violent protests. On average, we choose the top cities including Cairo, Tahrir Square, and Alexandria. These cities have the highest frequency of events in each year from 2012 to 2016.

**Thailand** We collect seven years of data from 2010 to 2016 and focus on the capital of Thailand, Bangkok, to ensure a balance between the number of instances and events. Rallies, violent protests and passage obstructions are the most frequent events.

**Russia** The protest data is limited, so we focus on Moscow, the capital of Russia, from 2010 to 2016. We also involve two other cities from 2010 to 2013 where protests have occurred for more than 20 days in a year.

### 4.2 Data Preparation

For each city, we take the documents within $k$ history days before the event as the raw input and the occurrences of target events as ground truth. Table 1 lists the key statistics of the four datasets. #documents is the total number of articles we utilized as raw input data and #vocabulary is the size of vocabulary after removing low frequency words appearing less than 5 times to assure the generality of the model. The vocabulary size varies from country to country due to the data we obtained. We disregard the samples with very few articles. Overall, for all datasets, the ratio of negative and positive samples is about 5 to 3.

### 4.3 Evaluation Metrics

To quantitatively evaluate our model, we use the following performance metrics:

**Prediction Performance** We evaluate the predictive performance of our our model in terms of Precision (Prec.), Recall (Rec.), and F1-Score (F1).

**Dynamic Context Graphs** We demonstrate the dynamic context graphs learned from the model for events that occurred in one country and discuss the effectiveness of our proposed model.

**Hyper Parameter Sensitivity** We analyze several hyper parameters in our model and test how different hyper parameter choices affect prediction performance.

**Table 2: Performance comparison on test set. (average over 20 trials)**

| | | Thailand | | | Egypt | | | India | | | Russia | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | F1 | Rec. | Prec. | F1 | Rec. | Prec. | F1 | Rec. | Prec. | F1 | Rec. | Prec. |
| Non-Temp. | LR (Count) | 0.7701 | 0.7129 | **0.8372** | 0.7945 | 0.7468 | 0.8488 | 0.6182 | 0.5589 | 0.6916 | 0.7389 | 0.7205 | 0.7582 |
| | LR (word TF-IDF) | 0.7151 | 0.6337 | 0.8205 | 0.7795 | 0.7511 | 0.8102 | 0.543 | 0.4335 | 0.7266 | 0.7048 | 0.6894 | 0.7208 |
| | LR (N-Gram TF-IDF) | 0.7293 | 0.6535 | 0.825 | 0.761 | 0.7039 | 0.8283 | 0.5515 | 0.4411 | **0.7355** | 0.7143 | 0.7143 | 0.7143 |
| | GCN | 0.7613 | 0.758 | 0.7663 | 0.8491 | 0.8161 | 0.8787 | 0.6533 | 0.6271 | 0.6853 | 0.784 | **0.8262** | 0.7469 |
| Temporal | nMIL | 0.7304 | 0.6614 | 0.8155 | 0.7234 | 0.7969 | 0.6623 | 0.6277 | **0.7193** | 0.5567 | 0.7595 | 0.7692 | 0.750 |
| | GCN+GRU | 0.7825 | 0.7686 | 0.7999 | 0.85 | 0.825 | 0.8775 | 0.6547 | 0.6215 | 0.6963 | 0.7866 | 0.8087 | 0.7677 |
| | GCN+LSTM | 0.7813 | 0.7702 | 0.7938 | 0.8507 | 0.8271 | 0.8766 | 0.6493 | 0.6137 | 0.6914 | 0.7858 | 0.7914 | 0.7812 |
| | GCN+RNN | 0.7566 | 0.7553 | 0.7585 | 0.851 | 0.8204 | 0.8851 | 0.6416 | 0.6016 | 0.6892 | 0.7868 | 0.8088 | 0.7667 |
| | **DynamicGCN** | **0.797** | **0.7734** | 0.8248 | **0.8617** | **0.8285** | **0.8984** | **0.6692** | 0.6275 | 0.7196 | **0.804** | 0.7988 | **0.8101** |

## 4.4 Comparison Methods

We compare our model with multiple event forecasting methods and their variants as below:

- Logistic Regression (LR) [17]: LR is a basic and popular algorithm for classification. In this experiment, we choose features including word count vectors and both word level and N-gram level TF-IDF vectors.
- Nested Multi-Instance Learning (nMIL) [19]: This is a hierarchical multi-instance learning framework for forecasting events and identifying historical documents as event precursors. It uses paragraph vectors as document level embeddings.
- Graph Convolutional Network (GCN) [11]: We employ the basic GCN model without using temporal data and let our transformation layer be the last layer. See Section 3.2 for details.
- GCN+GRU [32]: A Temporal Graph Convolutional Network for Traffic Prediction, which combines GCN and GRU to capture spatio-temporal correlations in traffic data. Each GCN layer of each time step processes the graph at this time, and the GCN output is processed by our transformation layer.
- GCN+LSTM: A variation of GCN+GRU, we change the GRU module to a LSTM module and use the LSTM layers to process the spatial features from the GCN layer at each time step.
- GCN+RNN: Similar to GCN+LSTM, we replace the LSTM module with a simpler RNN module.

## 4.5 Data Preprocessing

Given the datasets described in Section 4.1, we first preprocess all the data by cleaning and tokenizing words. Then we remove stop words and keep only stemmed words. We aim to find hidden relations between keywords in the prediction of future events. Therefore, our task is different from natural language processing tasks where the nuances of words should not be ignored. For each raw input ($x_{c,t}$ = {docs in days $t-1 \rightarrow t-k$}), we extract keywords with TF-IDF by ignoring words that have high document frequency (>80%) and then from which, we remove words appear in less than 5 times among all the documents in the dataset. On average, the number of nodes per graph is around 600. Then the dynamic graphs are constructed on these identified keywords following Section 3.1.

## 4.6 Parameter Setting

We pre-train a 100-dimensional word2vec embedding for each word in the vocabulary using all the documents in each country.

For hyper-parameter setting, the number of stacked dynamic GCN layers is set to be the number of history days. We adopt rectified linear units (RELU) [16] as nonlinearity (function $g$ in Eq. 6). The hyperparameter $\alpha$ in the TE module is set to 50. All the parameters are initialized with Glorot initialization [7] and trained using the Adam [10] optimizer with learning rate 5e-4, weight decay 5e-4, and dropout rate 0.2. We use 70%, 15%, 15% instances for training, validation and testing, respectively. The batch size is set to 1 across all datasets.

For the LR baseline models, we use the same number of features as the graph-structured data. For word count features, we remove the stop words, tokenize the text and convert the documents into a matrix of token counts. For TF-IDF features, we consider word-level and N-gram level features, where N is set to 2 and 3.

For the basic GCN model, we use non-temporal graph data where one graph is represented all the articles in historical days. For other GCN models involve temporal features, we use the same data and settings as our model. At each time step, we use one layer of GCN. For models combined with recurrent neural networks, the size of hidden states is optimized based upon grid search. The number of hidden layers is optimized to 1. In order to make the models converge faster, we apply the batch normalization [9]. The best models are selected by early stopping when the validation accuracy does not increase for 5 consecutive epochs.

# 5 EXPERIMENTAL RESULTS

## 5.1 Prediction Performance

We compare the prediction performance of all methods across the four datasets. Table 2 reports the prediction performance of the proposed DynamicGCN model[1] in comparison to other state-of-the-art approaches for the task of forecasting protests. The standard deviation of prediction performance in different trials is close to 0.016 among all models. We divide the compared approaches into two categories depending on whether they model temporal characteristics. We notice that for the non-temporal models, GCN has the highest F1 scores for Egypt, India and Russia, and outperforms LR with

[1]Code is available at https://github.com/amy-deng/DynamicGCN.

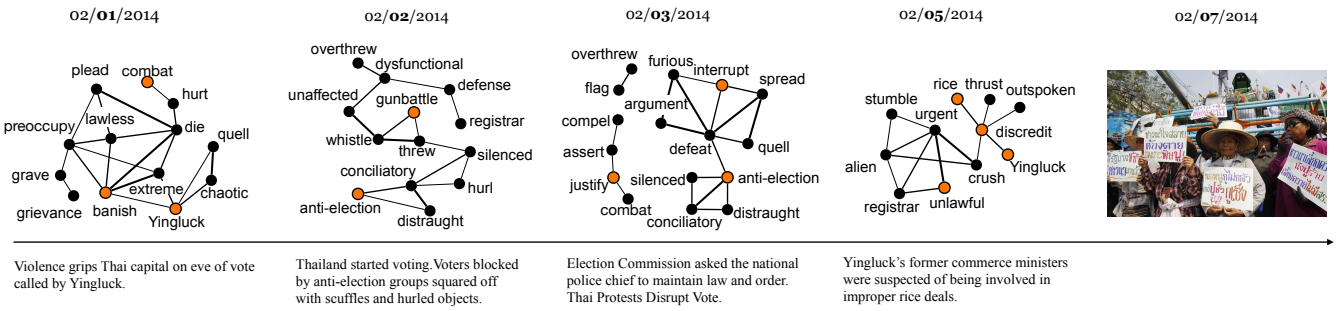



**Figure 3: An event in Thailand: Hundreds of rice farmers are heading towards the Ministry of Commerce in Bangkok to demand the government give their rice back and trying to oust Yingluck due to deception and waste in the rice scheme.**

count, word and N-Gram TF-IDF features by 6.2%, 13.5% and 13.3% in terms of average F1 score respectively. It highlights the insights that the GCN model effectively extracts hidden features of target events from the constructed word graph. Comparing the temporal models with non-temporal ones, using temporal information yields better predictive performance for most cases.

Focusing on the temporal graph models, the proposed DynamicGCN with temporal features achieves the best performance in terms of F1 scores across all the four datasets, and has the best recall for Thailand and Egypt. Russia and India datasets are more imbalanced and have relatively large vocabulary. On average, compared to the best baseline model, the DynamicGCN achieves 1.94% relative performance gain in F1 score over four datasets.

## 5.2 Dynamic Context Graphs

We present a case study to show how the proposed model captures the evolution of the dynamic context represented by graphs over time. We select a protest event that occurred in Thailand on Feb. 7, 2014. Figure 3 shows the extracted dynamic graphs over time based on our model. The texts below are simple summaries of the relevant event articles for the given day. We illustrate four subgraphs in the history days before the event. We construct these graphs following the procedure in Section 3.4 where $\sigma \approx 0.05$ in this case. In each subgraph, we highlight the nodes that are closely related to the target event. The thickness of the edge represents the weight. The three types of edges from thin to thick demonstrates weights for (0, 0.05), (0.05, 0.1) and (>0.1).

**Interpretation** The main story revolves around an election in Thailand. *Yingluck Shinawatra* was the Prime Minister in Thailand. The *rice scheme* was a program in Thailand to increase national rice export revenue. A farmer protest event happened in Feb 7. Two days before this event, the proposed model detected that Yingluck's former commerce minister was suspected to be involved in improper rice deals. Thus keywords such as *discredit, unlawful, rice,* and *Yingluck* in the graph were observed. The thick edges *rice-discredit* and *discredit-Yingluck* suggested a possible case of fraud involving rice traders and some politicians. The concext graph connected two issues from the past: *rice* and *discredit*. Then we look at the graphs two days earlier on Feb 3: the election was apparently interrupted by the anti-election activity. Backtracking to the voting day, the anti-election groups blocked the voters. The weighted thick

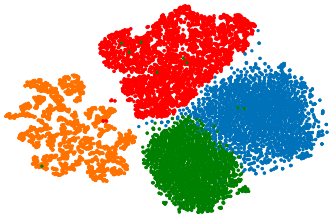
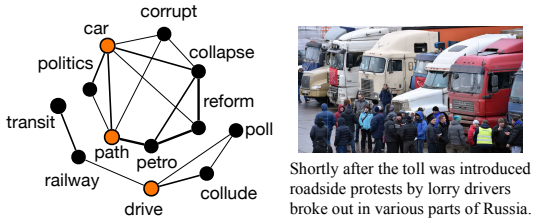


**Figure 4: t-SNE [13] visualization of the third GCN layer feature representations of four countries.**




Shortly after the toll was introduced roadside protests by lorry drivers broke out in various parts of Russia.

**Figure 5: Event summary graph for one event in Russia.**

edges *furious-defeat* and *threw-whistle-unaffected* show that there might be an intense collective struggle. On the eve of the voting day, a violent incident occurred against the Prime Minister.

The election riots in the past days and the new rice incident reinforced the farmers' protests. Based on this contextual information the model predicts that protest events are very likely to happen in the near future. Compared to the non-graph or static graph approaches, dynamic context graphs help analysts in capturing the evolving information before events of interest, making manual inference easier and clearer.

We also provide a t-SNE [13] visualization of the computed feature representations of the pre-trained dynamic GCN model on four datasets in Figure 4. For each country, we select about 3000 words. The feature representation is calculated by $H_0 W^{(2)}$, where $H_0$ is the word embedding features of words and $W^{(2)}$ is the learned weight matrix in the third GCN layer. Points of different colors are representations of the learned feature in different countries, verifying the discriminative representations learned from the model.

Figure 5 demonstrates an example of an event summary graph. We build subgraphs according to Section 3.4 and combine them into

a single summary graph. In this event, lorry drivers were protesting against the toll on roadside. In the detected summary graph, we found two related storylines. One is about the transportation system including railway, drive, transit etc. The other is about a reform on road systems. Both are highly related to the protest event.

**Table 3: Runtime comparison of graph models on Egypt dataset. Structure is the number of GCN layers. Runtime is the time spent on single GPU per epoch.**

| Architecture | Structure | Parameters | Runtime(s) |
|---|---|---|---|
| GCN | $K = 2$ | 37,685 | 20 |
| GCN+GRU | $K = 7$ | 5,251,396 | 22 |
| GCN+LSTM | $K = 7$ | 7,001,604 | 24 |
| GCN+RNN | $K = 7$ | 1,750,980 | 18 |
| DynamicGCN | $K = 7$ | 149,785 | 32 |

## 5.3 Model Complexity

We use a sparse representation for the adjacency matrix and perform stochastic gradient descent in the training process. For GCN layers, by applying sparse-dense matrix multiplications, the computational complexity of evaluating Eq. 3 and the memory requirements of the adjacency matrix are linear with respect to the number of graph edges. We sample the keywords for graphs so that we can easily put a batch of data into memory. The $K$th-order (dynamic) neighborhood for a GCN with $K$ layers has to be stored in memory for an exact procedure. In our experiment design ($K$ up to 7), the model works well for all datasets.

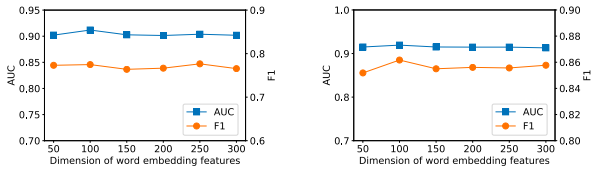
All experiments were run on the same machine using one GPU. Table 3 shows the comparison of runtimes and numbers of parameters for each model on the Egypt dataset. The basic GCN model processes on non-temporal data with 2 graph convolutional layers. For temporal baseline models, the size of hidden states in recurrent units is set to 64 here. Our model has much less parameters than GCN+LSTM, GCN+GRU and GCN+RNN. We introduced the temporal encoded module for processing temporal features. Combining GCN and recurrent neural networks involves more parameters in each of the time-dependent gates.

## 5.4 Hyper-parameter Sensitivity

We investigate the prediction performance with varying dimensions of word embeddings and variance with respect to both the number of prior days and lead time.

**Dimension of Word Embedding Features** Each node is encoded with Word2vec embedding features with dimension 100. We vary the dimension of the embedding feature from 50 to 300, where the embeddings are trained on the same data. We show results for only the Thailand and Egypt datasets. Figure 6 shows the prediction performance in terms of AUC and F1 with varying number of embedding dimensions. We observe that the word2vec dimension does not affect the performance of the model significantly.

**Number of Prior (Historical) Days** Historical days denotes the number of days over which the articles are extracted as input to the prediction algorithms. We trained four different models that use different numbers of history days from 5 to 8 respectively. For each country, we consider the data for the year with the largest




**(a) Thailand** **(b) Egypt**

**Figure 6: Sensitivity analysis on the dimension of word embedding features.**

amount of data. We present the F1 score and AUC results for all countries in Figure 7. It is not always true that modeling with more data will achieve better predictive performance. For instance, in Egypt and Russia, the best F1 scores are obtained when using 6 historical days instead of 8.

**Prediction of Future Days** We also study the performance with varying lead time from 1 to 4, where lead time indicates the number of days in advance a model makes predictions. For each country, we use one year's data. Figure 8 shows the results of the F1 score and AUC in different lead time settings of the four datasets. The best performance is achieved when leadtime = 2 or 4 for all the four datasets. This can be explained by the idea that social events usually take days to be planned.

## 6 CONCLUSION AND FUTURE WORK

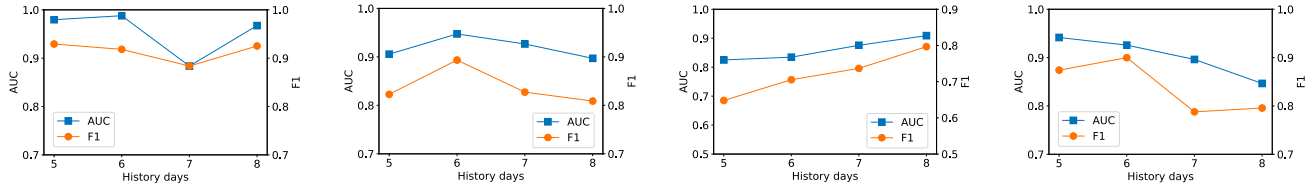
Event forecasting and event context detection are important tasks for decision makers and policy analysts. This paper presents a novel dynamic graph convolutional model with a temporal encoded feature module for event forecasting and for identifying dynamic context graphs. We demonstrated the effectiveness of the proposed model on large-scale real-world open source datasets. In the future, we plan to explore a few directions: 1) Automatic relationship extraction for entities in dynamic graphs. Entity interactions have an impact on social event development. Accurate entity relation extraction will be helpful for generating narrative graph lines. 2) Long-term event dependency. In this work, we focus on short-term event forecasting and explanation. However, social events can also be related to activities further in the past or change events further in the future. 3) Another important direction is to consider multiple geolocations simultaneously and study the influence from neighbor locations. Social events are usually not independent of each other. Thus, considering various geolocations is critical in spatio-temporal event modeling.

## ACKNOWLEDGMENTS

This material is based upon work supported by the U.S. Department of Homeland Security under Grant Award Number 2017-ST-061-CINA01. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Department of Homeland Security.

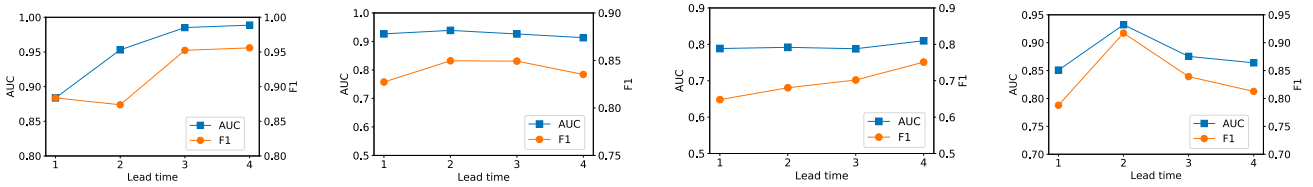
## REFERENCES

[1] Harshavardhan Achrekar, Avinash Gandhe, Ross Lazarus, Ssu-Hsin Yu, and Benyuan Liu. 2011. Predicting flu trends using twitter data. In *IEEE Conference on Computer Communications Workshops*. IEEE, 702–707.




(a) Thailand (b) Egypt (c) India (d) Russia

**Figure 7: Sensitivity analysis on the number of history days.**




(a) Thailand (b) Egypt (c) India (d) Russia

**Figure 8: Sensitivity analysis on the number of day of lead time.**


[2] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research* 3, Feb (2003), 1137–1155.

[3] Johan Bollen, Huina Mao, and Xiaojun Zeng. 2011. Twitter mood predicts the stock market. *Journal of computational science* 2, 1 (2011), 1–8.

[4] Elizabeth Boschee, Jennifer Lautenschlager, Sean O'Brien, Steve Shellman, James Starz, and Michael Ward. 2015. ICEWS Coded Event Data. (2015).

[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 *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. 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] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 249–256.

[8] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[9] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*. JMLR.org, 448–456.

[10] D Kinga and J Ba Adam. 2015. A method for stochastic optimization. In *International Conference on Learning Representations*, Vol. 5.

[11] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

[12] Theodoros Lappas, Marcos R Vieira, Dimitrios Gunopulos, and Vassilis J Tsotras. 2012. On the spatiotemporal burstiness of terms. *Proceedings of the VLDB Endowment* 5, 9 (2012), 836–847.

[13] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, Nov (2008), 2579–2605.

[14] Diego Marcheggiani and Ivan Titov. 2017. Encoding Sentences with Graph Convolutional Networks for Semantic Role Labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 1506–1515.

[15] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[16] Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning*. 807–814.

[17] John Neter, Michael H Kutner, Christopher J Nachtsheim, and William Wasserman. 1996. *Applied linear statistical models*. Vol. 4. Irwin Chicago.

[18] Thien Nguyen and Ralph Grishman. 2018. Graph Convolutional Networks With Argument-Aware Pooling for Event Detection. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*.

[19] Yue Ning, Sathappan Muthiah, Huzefa Rangwala, and Naren Ramakrishnan. 2016. Modeling precursors for event forecasting via nested multi-instance learning. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1095–1104.

[20] 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 *Proceedings of the 2018 SIAM International Conference on Data Mining*. SIAM, 99–107.

[21] Brendan O'Connor, Ramnath Balasubramanyan, Bryan R Routledge, Noah A Smith, et al. 2010. From tweets to polls: Linking text sentiment to public opinion time series. (2010), 2 pages.

[22] Hao Peng, Jianxin Li, Yu He, Yaopeng Liu, Mengjiao Bao, Lihong Wang, Yangqiu Song, and Qiang Yang. 2018. Large-Scale Hierarchical Text Classification with Recursively Regularized Deep Graph-CNN. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. 1063–1072.

[23] Jiezhong Qiu, Jian Tang, Hao Ma, Yuxiao Dong, Kuansan Wang, and Jie Tang. 2018. DeepInf: Modeling influence locality in large social networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[24] Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. 2010. Earthquake shakes Twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*. ACM, 851–860.

[25] Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. 2018. Structured sequence modeling with graph convolutional recurrent networks. In *International Conference on Neural Information Processing*. Springer, 362–373.

[26] 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.

[27] Andranik Tumasjan, Timm Oliver Sprenger, Philipp G Sandner, and Isabell M Welpe. 2010. Predicting elections with twitter: What 140 characters reveal about political sentiment. *Icwsm* 10, 1 (2010), 178–185.

[28] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. *International Conference on Learning Representations* 1, 2 (2018).

[29] 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.

[30] Jianshu Weng and Bu-Sung Lee. 2011. Event detection in twitter. *International AAAI Conference on Web and Social Media* 11 (2011), 401–408.

[31] Liang Yao, Chengsheng Mao, and Yuan Luo. 2018. Graph Convolutional Networks for Text Classification. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*.

[32] Ling Zhao, Yujiao Song, Min Deng, and Haifeng Li. 2018. Temporal Graph Convolutional Network for Urban Traffic Flow Prediction Method. *arXiv preprint arXiv:1811.05320* (2018).

[33] Liang Zhao, Qian Sun, Jieping Ye, Feng Chen, Chang-Tien Lu, and Naren Ramakrishnan. 2015. Multi-task learning for spatio-temporal event forecasting. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1503–1512.

# A PSEUDOCODE FOR THE PROPOSED DYNAMIC GCN

We present the training steps of the proposed DynamicGCN model in Algorithm 1.

**Algorithm 1: DynamicGCN**

**Input:** Initial embeddings, adjacency matrices, event indicators
**Output:** Model parameters $\mathbf{W} = \{W_p, b_p, W_e, b_e, W, b, w_m, b_m\}$

**for** *each epoch* **do**
  b ← random sample a batch
  **for** *each instance* $\in b$ **do**
    $\tilde{H}_t \leftarrow H_0$
    **for** $t \leftarrow 0$ *to* $T-1$ **do**
      **if** $t > 0$ **then**
        $H_p^{(t)} \leftarrow H_t W_p^{(t)} + b_p^{(t)}$
        $H_e^{(t)} \leftarrow H_0 W_e^{(t)} + b_e^{(t)}$
        $\tilde{H}_t \leftarrow \tanh([H_p^{(t)} \parallel H_e^{(t)}])$
      $H_{t+1} \leftarrow \text{relu}(A_t \tilde{H}_t W^{(t)} + b^{(t)})$
    $z_t \leftarrow \text{zero\_padding}(H_t^\intercal)$
    $\hat{y} \leftarrow \sigma(z_t w_m^\intercal + b_m)$
    $\Delta_{\mathbf{W}} \mathcal{L} \leftarrow \text{BackProp}(\mathcal{L}, y, \hat{y}, \mathbf{W})$
    $\mathbf{W} \leftarrow \mathbf{W} - \eta \Delta_{\mathbf{W}} \mathcal{L}$ ▷ SGD step

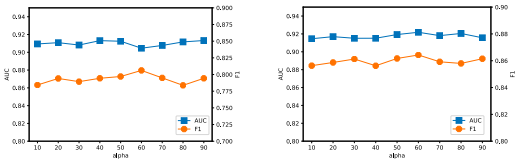
# B EXPERIMENTS OF THE TEMPORAL ENCODED MODULE

## B.1 Analysis of the TE Module

We compare the performance of the model with no TE module settings in the four datasets. We also report the results when using different historical days. The F1 score results are shown in Figure 9. In most data settings, our model with the TE module outperforms the one without TE modules, which proves its effectiveness in handling temporal features.

## B.2 Analysis of Hyper-Parameter $\alpha$ in the TE Module

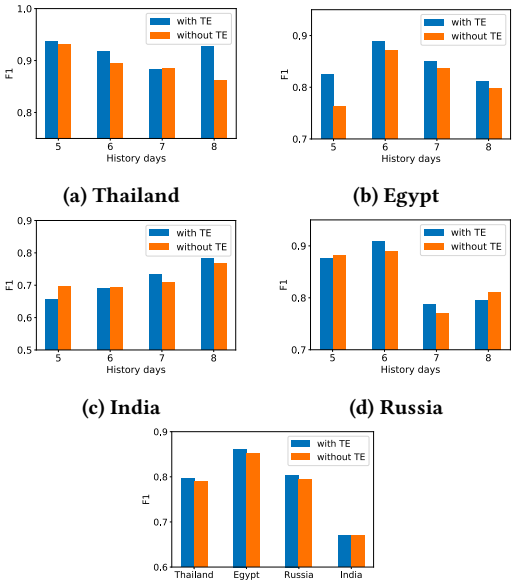
In the Temporal Encoded module design, we consider both the GCN output feature and the initial word embedding which includes the semantic information of the words. The two features are combined in the TE module with a hyper-parameter $\alpha$.




**(a) Thailand** **(b) Egypt**

**Figure 10: Sensitivity analysis on $\alpha$.**

We set the dimension of word embedding to be 100. To vary the combination of the two features, we change $\alpha$ value to 10, 20, 30, 40, 50, 60, 70, 80, 90, and test our model performance of F1 and AUC scores on the Egypt and Thailand datasets. The results are presented in Figure 10. It shows that the performance does not get affected by the variance of this hyper-parameter. We obtain the highest F1 score when $\alpha$ is around 60.




**(a) Thailand** **(b) Egypt**

**(c) India** **(d) Russia**

**(e) Four countries**

**Figure 9: Sensitivity analysis on TE Module.**