2021 IEEE 37th International Conference on Data Engineering (ICDE) | 978-1-7281-9184-3/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICDE51399.2021.00099

EDGE: Entity-Diffusion Gaussian Ensemble for Interpretable Tweet Geolocation Prediction

Bo Hui
Auburn University
bohui@auburn.edu

Haiquan Chen
California State University, Sacramento
haiquan.chen@csus.edu

Da Yan
University of Alabama at Birmingham
yanda@uab.edu

Wei-Shinn Ku
Auburn University
weishinn@auburn.edu

***Abstract*—Knowing the locations of tweets can benefit a wide variety of applications such as venue recommendation, event detection, and monitoring disaster outbreaks. However, the problem of fine-grained tweet geolocation prediction is challenging since tweets are short and therefore may not contain any geo-indicative words or may contain ambiguous, noisy information. Existing solutions either yield an unsatisfactory accuracy in practical applications or make predictions that even experts struggle to interpret, failing to engender sufficient trust and actionability for real-world deployment. Our paper presents a tweet geolocation prediction framework, EDGE (Entity-Diffusion Gaussian Ensemble), which delivers predictions that are both accurate and highly interpretable without requiring any additional contextual information such as user profile and location history. In EDGE, we cast the geolocation problem as a neutral network optimization problem by learning probabilistic generative models. Compared with existing works, EDGE has two distinctive features: (1) the inference builds on mining the correlation between non geo-indicative entities and geo-indicative entities by diffusing their semantic embeddings over the constructed graph neural network (Entity Diffusion) and (2) each prediction result is represented as a Gaussian mixture instead of specific geographical coordinates (Gaussian Ensemble). Extensive experiments using real-world tweet datasets validate the superiority of EDGE over the state of the art in terms of all distance-based and POI-based metrics.**

***Index Terms*—Tweet Geolocation, Gaussian Mixture Model, Graph Neural Network**

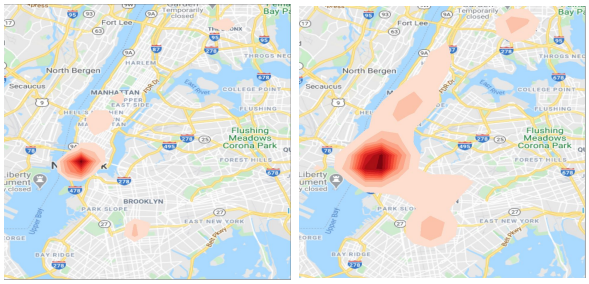
I. Introduction

With the prevalence of smart devices and the worldwide accessibility to the Internet, social media have become "the go-to platform" of the World Wide Web. Twitter is one of the most popular social media where people share their views, real-time information and reactions to the days events. On average, more than 100 million users post about 500 million tweets per day [38]. Knowing the locations of tweets can benefit a wide variety of applications such as venue recommendation, event detection and monitoring disaster outbreaks [18], [29], [36], [40]. In particular, this kind of inference allows for more robust spatio-temporal analysis for dynamic events on social media. For example, the identification of the spreading pattern of COVID-19 related tweets can provide a sense of how COVID-19 is spreading geographically over time, which is vital to developing effective strategies to contain the pandemic. Unfortunately, in reality, only a small portion (1%-2%) of tweets are posted with a precise location (GPS coordinates) [34]. Therefore, there is a pressing need to determine locations of those tweets which are not posted with precise locations. However, existing solutions to tweet geolocation prediction [11], [41] either yield an unsatisfactory accuracy from a practical perspective or make predictions that even experts struggle to interpret, therefore failing to engender sufficient trust and actionability for real-world deployment.

This paper presents a tweet geolocation prediction framework, Entity-Diffusion Gaussian Ensemble (EDGE), that **delivers predictions that are both accurate and highly interpretable without requiring any additional contextual information**. EDGE consists of three seamlessly integrated modules: (1) entity embedding extraction and diffusion, (2) attention aggregation, and (3) mixture distribution learning.

Specifically, EDGE casts the geolocation problem as a neutral network optimization problem by learning a Gaussian mixture for each tweet. EDGE has two distinctive features: (1) the inference builds upon **mining the correlation between non geo-indicative entities and geo-indicative entities** by diffusing their semantic embeddings over the constructed graph neural network and (2) each prediction result is returned as **a Gaussian mixture rather than specific geographical coordinates.**

A Running Example. Figure 1 compares the geographic distributions of the tweets mentioning "quarantine" in two separate time periods in New York during the COVID-19 pandemic. Specifically, Figure 1(a) plots the tweets posted between March 12, 2020 and March 22, 2020 while Figure 1(b) depicts the tweets posted between March 22, 2020 and April 2, 2020. Note that the location distribution of those tweets was predicted by our model. As shown in Figure 1, a

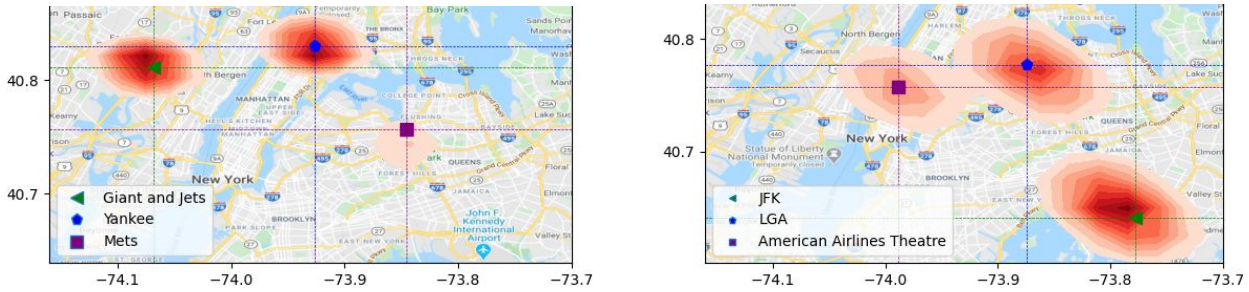


(a) March 12, 2020 to March 22, 2020

(b) March 22, 2020 to April 02, 2020

Fig. 1: Geographical distributions of the tweets mentioning *quarantine* collected in two separate durations in New York during the COVID-19 pandemic, where the locations of the non geo-tagged tweets were predicted by EDGE.

978-1-7281-9184-3/21/$31.00 ©2021 IEEE
DOI 10.1109/ICDE51399.2021.00099



(a) Tweets with Stadium in content

(b) Tweets with American Airlines in content

Fig. 2: Geographical distribution of tweets in New York. The subfigure (a) shows the location distribution of tweets mentioning *Stadium*, where the green triangle is *MetLife Stadium*, the blue polygon refers to *Yankee Stadium*, and the purple square indicates *CitiField*. The subfigure (b) shows the geographical distribution of tweets mentioning *American Airlines*, where the green triangle, the blue polygon, and the purple square represent *John F. Kennedy International Airport*, *LaGuardia Airport*, and *American Airlines Theatre*, respectively.

quick spreading of the COVID-19 cases in New York can be observed.

We summarize our contributions as follows:

- We present EDGE, a novel tweet geolocation prediction framework that delivers predictions that are both accurate and interpretable without requiring mentions of locations in the text or any additional contextual information.
- We propose *entity2vec* to extract embedding for named entities appearing in tweets instead of treating them as a composition of independent words.
- We develop an entity diffusion mechanism to capture the correlation between non geo-indicative entities and geo-indicative entities by diffusing spatially distinctive information over a graph convolutional network. This design allows us to learn the spatially smoothed embedding for each entity.
- We design an attention mechanism to weight the importance of the entities that co-occur in the same tweet while extracting the spatially smoothed embedding for each tweet. Parameters are learned to weigh among fine-grained geo-indicative entities and coarse-grained geo-indicative entities.
- We train EDGE in an end-to-end manner to maximize the likelihood that geo-tagged tweets are located in their associated locations. EDGE returns a Gaussian mixture as the prediction result and is general enough to predict the locations of tweets within a region of any size.
- We conduct extensive experiments using real-world datasets to demonstrate the superiority of EDGE over the state of the art in terms of all distance-based and POI-based metrics.

The rest of this paper is organized as follows. In Section II, we give an overview of EDGE. In Section III, we elaborate on our model design. Section IV shows the comparative evaluation with the state of the art and presents the parameter sensitivity analysis. In Section V, we demonstrate two potential use cases. Section VI reviews the related work and Section VII concludes this paper.

II. Overview of EDGE

In this section, we overview the design of our EDGE model motivated from two challenges in location inference from a tweet: (1) one challenge comes from the limited amount of geo-indicative information that can be exploited from a short tweet of up to 280 characters, and the free writing style on twitter that does not work well with conventional NLP tools; (2) another challenge is how to generate both accurate and highly interpretable predictions. To address these challenges, we obtain two important observations about the geolocation characteristics of tweets, which motivate our EDGE architecture design with three important modules.

A. Our Observations

Observation 1 (O1): Tweets that share a specific entity tend to cluster in groups geographically. This phenomenon can be explained by the fact that an entity may refer to one or a few geo-places. For example, Figure 2(a) depicts the geolocation distribution of tweets mentioning *Stadium* in their content in New York, where three clusters can be observed, corresponding to the three stadiums in New York (*MetLife Stadium, Yankee Stadium, Citi Field*). Given the tweet, "*I love my icy office!!!! I love sound checking in empty stadiums before every game*", it is reasonable to infer that the location of the tweet would be close to one of those three stadiums with a relatively high probability. Note that the distance between any two stadiums is non-negligible. Similarly, Figure 2(b) illustrates the three multi-modal geolocation distribution of the tweets mentioning "*American Airlines*" in New York, corresponding to the two airports in New York (*John F. Kennedy International Airport* and *LaGuardia Airport*) and *American Airlines Theatre* in Manhattan. In addition, streets in different regions of a city may share the same name, e.g., Fifth Avenue in Manhattan and Fifth Avenue in Park Slope, both of which are shopping streets. Another example of different places sharing the same entity name can be the restaurant chains having multiple franchise locations.

Solution to O1: Predicting the location of a tweet as a Gaussian mixture distribution. In EDGE, the geo-location of

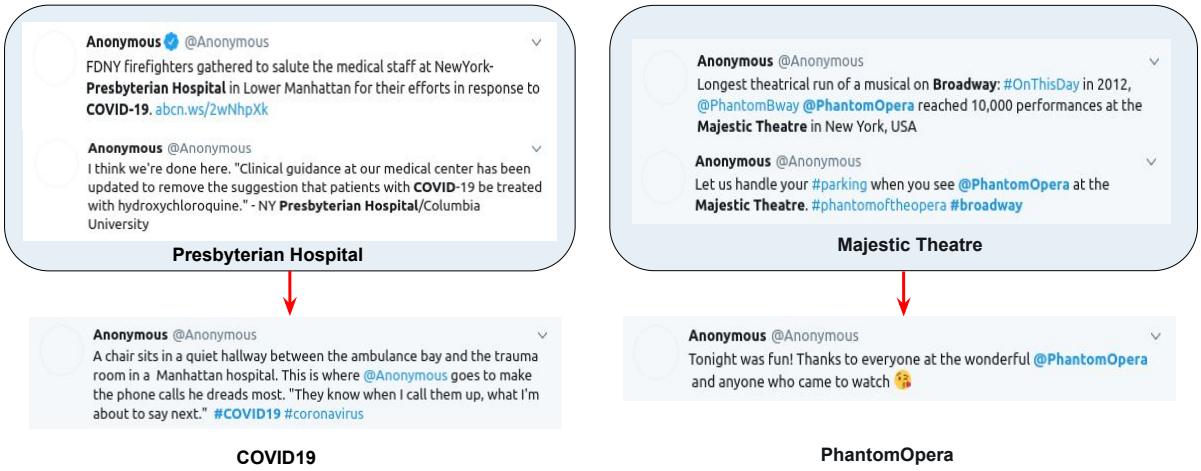


Fig. 3: Geo-indicative entities that co-appear with non geo-indicative entities can be leveraged to reinforce geolocation prediction. Specifically, the location of the lower-left tweet is likely to be close to *Presbyterian Hospital* while the location of the lower-right tweet is likely to be around *Majestic Theatre* and *Broadway*.

a tweet will be predicted as a Gaussian mixture where each component is a bivariate normal distribution.

Observation 2 (O2): Geo-indicative entities that co-occur with non geo-indicative entities can be leveraged to reinforce geolocation prediction. Due to the length limitation (280 characters) and the free writing style of tweets, the number of geo-terms in a tweet tends to be limited. However, we observe that there are bridges between geo-indicative entities and non geo-indicative entities that can help geolocation inference. For example, during the pandemic of COVID-19, we have observed frequent co-occurrence of *COVID-19* (a non geo-indicative entity) with some geo-indicative entities (e.g., *Presbyterian Hospital*). Those geo-indicative entities that co-appear with non geo-indicative entities can be leveraged to reinforce geolocation prediction. As depicted in Figure 3(a), for the tweet *"This is for real... hospital this morning during the #covid19 pandemic"*, since *COVID-19* is frequently co-mentioned with *Presbyterian Hospital* at that time in New York, the tweet might be posted close to *NY Presbyterian Hospital* with a relatively high probability. Figure 3(b) shows another example. When predicting the location of the tweet, *"@PhantomOpera was a great way to end our NY trip ...."*, since *@PhantomOpera* was frequently mentioned along with *Majestic Theatre* and *Broadway* at that time in New York, the tweet might be posted close to *Majestic Theatre* and *Broadway*. Actually, *Majestic Theatre* at *Broadway* in New York is the most famous theater that stages "The Phantom of the Opera.

Solution to O2: Bridging non geo-indicative entities with geo-indicative entities by diffusing their embedding using graph convolutions. In EDGE, each entity is first represented as an embedding using the proposed **Entity2vec** technique instead of being treated as a composition of independent words. Next, to exploit the correlation between non geo-indicative and geo-indicative words, we construct a co-occurrence based entity graph and utilize graph-based convolution [14], [42] to smooth the extracted embedding of each entity over its own ego network.

B. EDGE Architecture

Figure 4 shows the architecture of EDGE. EDGE consists of three seamlessly integrated modules: (1) entity embedding extraction and diffusion, (2) attention aggregation, and (3) mixture distribution learning. EDGE casts the geolocation problem as a neutral network optimization problem by learning a Gaussian mixture for each tweet. Specifically, (1) we first extract the embedding for each named entity by using the proposed *entity2vec* technique, which is inspired by *phrase2vector* [21]. Since an entity is not a direct composition of independent words, our *entity2vec* learns embedding by treating each entity as a whole and capturing syntactic and semantic relationships between entities. Then we construct a co-occurrence based entity graph and utilize Graph Convolutional Network (GCN) to smooth the extracted embedding of each entity over its own ego network. The goal of using graph convolution is to learn the spatially smoothed embedding for each entity, so that the embedding of a tweet without location in its text can also carry geo-indicative information. (2) Next, in the attention aggregation, the smoothed embeddings of entities are weighted to create the spatially smoothed embedding for each tweet, which is then fed into a fully-connected neural network to generate the parameters of a Gaussian mixture distribution. The attention aggregation module not only aggregates a set of entity vectors (the number of entities in tweets varies) into a tweet vector with fixed length but also differentiates the importance of entities that appear in the same tweet in terms of their capacities to infer locations. (3) Finally, a Gaussian mixture module is introduced to address *Observation 1*. We train EDGE in an end-to-end supervised manner to fit the parameters of Gaussian mixtures to maximize the likelihood that geo-tagged tweets are located in their associated locations. Given a tweet, EDGE returns a Gaussian mixture distribution as the geolocation prediction.

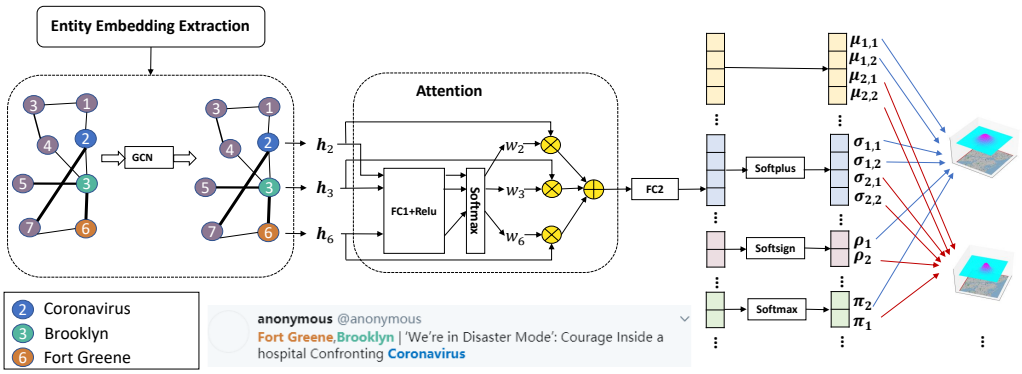


Fig. 4: EDGE Framework. Each edge in the entity graph represents the co-occurrence of two named entities. Given a tweet which contains three entities (*Coronavirus*, *Brooklyn*, and *Fort Greene*), EDGE first utilizes graph-based convolution to smooth the extracted *entity2vec* embedding of each entity over its own ego network. Subsequently, the attention model weights the importance of the embedding for each entity and extracts the embedding for that tweet. Afterwards, EDGE fits the parameters of a Gaussian mixture to maximize the likelihood that the tweet is located in its associated location. Each set of ($\boldsymbol{\mu}$, $\boldsymbol{\sigma}$, ρ π) represents a specific Gaussian distribution component.

III. Methodology

A. Entity Embedding Extraction and Diffusion

In this subsection, we first describe how EDGE extracts entity embedding from tweets and then describe how the correlation between named entities can be captured in EDGE by diffusing the embedding via graph convolutions.

1) Entity Embedding Extraction: To generate the representation of a tweet, existing works use n-grams [7], bag-of-words [27], or simply the number of words [5], [12]. In this paper, we regard each tweet as a set of named entities [32]. A *named entity* is a real-world object, such as a location, an organization, a product, etc., that can be mentioned with a proper name. It can be abstract (e.g., COVID-19) or have physical existence (e.g., Times Square). We used Chunker Named Entity Recognizer[1] [28] to extract named entities and represent each tweet as a set of named entities. For example, given the second tweet as shown in Figure 3(a), after entity extraction, it will be represented as a set of three named entities, *COVID-19*, *Presbyterian Hospital*, and *Columbia University*.

TABLE I: Examples of named entities in tweets

Tweets	# of Words in the entity
Times Square is absolutely beautiful	2
Somebody on the **George Washington Bridge** just stopped their car and jumped off the bridge	3
Dance Theatre of Harlem! Here we come!	4
Damn these streets are empty **#pandemic #coronavirus**	1

Entity2vec. Notice that there can be multiple words in an entity, as shown in Table I. Moreover, entities with more than one word may not be a direct composition of their components. For example, Times Square is a landmark, and therefore its embedding should not be the direct composition of the embeddings of Times and Square while considering a named entity as several independent words ignores the semantic segment of the named entity. To handle named entities with more than one word, in *entity2vec*, we tokenize those named entities by treating them as phrases and a *word2vec*[2] model [21] is then trained on the collected tweets to obtain the semantic embedding of each entity.

2) Entity Embedding Diffusion: In order to bridge non geo-indicative with geo-indicative entities, EDGE fuses their embeddings over a graph convolutional network, which generates the spatially smoothed embedding for each named entity.

Entity Graph Construction. EDGE constructs an undirected entity graph based on the number of co-occurrences of two entities in tweets. Formally, in the entity graph $\mathcal{G} = (V, E)$, each node corresponds to an entity and its attributes are the semantic embedding learned using *entity2vec*. If two named entities v_i and v_j appear in the same tweet, there will be an edge $e_{i,j}$ between v_i and v_j. The weight $e_{i,j}$ is the number of the co-occurrences of two referenced entities in the training set.

Embedding Diffusion. The goal of embedding diffusion is to learn the spatially smoothed embedding for each entity which carries the spatially distinctive information by taking advantage of the correlation (co-occurrences) between non geo-indicative entities and geo-indicative entities. Recently, graph convolutional networks have been widely used in various tasks such as node classification [10], link prediction [19] , and graph classification [15]. One of the most distinctive features of graph convolutional networks is that they can effectively learn node (edge) representations using both network topology and node attributes, which differs from node embedding methods such as *node2vec* [9] and *DeepWalk* [25] which only take into account network topology. EDGE uses a graph convolutional network [14] to effectively learn the smoothed embedding of each named entity by integrating the semantic embeddings (features) of its neighbors. The convolution parameters are learned automatically from the training data.

[1] http://github.com/aritter/twitter_nlp

[2] https://radimrehurek.com/gensim/models/word2vec.html

Each graph convolution layer [14] in EDGE can be written using Equation (1).

$$H^{(\ell+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(\ell)}W^{(\ell)}), \quad (1)$$

where A is the adjacency matrix of G and $\tilde{A} = A + I$ is the adjacency matrix with self-connections added, $\tilde{D}$ is a diagonal matrix with $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$, $W^{(\ell)}$ is the weight matrix to train for Layer ℓ, σ is an activation function and $H^{(\ell)}$ is the state matrix where each row is the features of a named entity at Layer ℓ. In the initial state, $H^{(0)}$ is the input matrix X where each row in X is the semantic embedding vector of a named entity learned from our *entity2vec* model. We use ReLU as the activation function for each graph convolutional layer. By stacking n layers of graph convolutions, we can diffuse the semantic embedding of each node over its n-hop ego-net. In our current implementation of EDGE, we employed a two-layer graph convolutional network. By learning on the entity graph, the graph convolutional networks generate the smoothed embedding for each named entity. Since in EDGE, a tweet t is represented by a subset of entities set V, the embedding diffusion module converts each tweet to a set of smoothed embeddings, with each embedding corresponding to a named entity. Note that if a tweet mentioned a specific entity more than one time, the entity will only be counted once in the set. We use $V(t)$ to represent the set of smoothed embeddings for tweet t, which will be forwarded to the attention aggregation module to produce an aggregated embedding for each tweet.

With GCN, our model does not require a tweet to have a location in its text for location inference. For example, the tweet in Figure 3: "*Tonight was fun! Thanks to everyone at the wonderful @PhantomOpera and anyone who came to watch*" has no location mentioned in this tweet, but the term "PhantomOpera" co-occurs frequently with other geo-indicative entities such as Majestic Theatre and "Broadway" in our entity graph. By diffusing their embeddings on the graph, the hidden states of PhantomOpera will carry information of Majestic Theatre and Broadway, so the previous tweet will be predicted to be close to Majestic Theatre with a high probability.

B. Attention Aggregation

EDGE employs an attention mechanism to aggregate entity embeddings to produce the embedding for each tweet by differentiating the importance of named entities that appear in the same tweet in terms of their capacities to identify locations. For each entity in the tweet, we learn a weight to indicate the importance to geolocation inference. Let $\mathbf{h_k}$ be kth vector in V_t, where V_t contains embeddings of all entities in a tweet. Each $\mathbf{h_k}$ is fed to a fully connected layer with bias, as shown in Equation (2) where $Q^{(1)}$ represents the weights and $b^{(1)}$ is the bias. The output of this fully connected layer is a score s_k which represents the importance (weight) of the corresponding entity (kth named entity). Then we use a softmax function to normalize those weights, as shown in Equation (3).

$$s_k = \text{ReLU}(Q^{(1)} \cdot \mathbf{h_k} + b^{(1)}) \quad (2)$$

$$w_k = \frac{exp(s_k)}{\sum_{k=1}^{K} exp(s_k)}, K = |V(t)| \quad (3)$$

To aggregate all entity embeddings in $V(t)$, EDGE employs a scaled dot-product attention method [3], as shown in Equation (4).

$$\mathbf{z} = \sum_{k=1}^{K} w_k \cdot \mathbf{h_k} \quad (4)$$

Here the output vector is the aggregation of the entity embeddings and attention scores. For example, given a tweet mentioning two entities, *"Times Square"* and *"New Year's Eve"*, such attention mechanism allows us to focus more on *"Times Square"* than on *"New Year's Eve"* since *"Times Square"* is a geo-indicative entity while *"New Year's Eve"* is a non geo-indicative entity. Notice that this design also allows us to pay more attention to the fine-grained geo-indicative entities than the coarse-grained geo-indicative entities. For example, *"William Street"* is a fine-grained geo-indicative entity while *"Brooklyn"* is a coarse-grained geo-indicative entity.

C. Mixture Distribution Learning

Unlike existing works, the goal of EDGE is to learn a Gaussian mixture as the geolocation prediction result given a tweet, i.e., the location where the tweet is posted will be predicted as a mixture of bivariate Gaussian distributions. Suppose the number of Gaussian distributions to be learned for each tweet is M. As shown in Equation (5), each bivariate Gaussian component can be represented as $\mathcal{N}(\boldsymbol{l}|\boldsymbol{\mu}_m, \sum_m)$, where $\boldsymbol{\mu}_m$ is the center (mean) of mth distribution, represented by latitude and longitude, and $\sum_m$ is the covariance matrix. The correlation parameter ρ_m in the covariance matrix measures the correlation between latitude and longitude. In addition to the mean and the covariance matrix for each distribution component, we also learn the weight of each distribution component, which indicates the cumulative probability of each distribution. For a specific location $\boldsymbol{l}$, its probability density function can be defined as Equation (6), where π_m represents the weight of mth Gaussian component. The weights here are used to differentiate the importance of Gaussian distribution components.

$$\begin{aligned} \boldsymbol{\mu}_m &= \begin{pmatrix} \mu_{m,1} \\ \mu_{m,2} \end{pmatrix} \\ \sum\nolimits_m &= \begin{pmatrix} \sigma^2_{m,1} & \rho_m\sigma_{m,1}\sigma_{m,2} \\ \rho_m\sigma_{m,1}\sigma_{m,2} & \sigma^2_{m,2} \end{pmatrix} \end{aligned} \quad (5)$$

$$pdf(\boldsymbol{l}) = \sum_{m=1}^{M} \pi_m \mathcal{N}(\boldsymbol{l}|\boldsymbol{\mu}_m, \sum\nolimits_m) \quad (6)$$

To obtain those aforementioned parameters for M distributions, we forward the output of the attention aggregation module $\mathbf{z}$ to a fully connected layer as shown in Equation 7, where the weight $Q^{(2)}$ and the bias $\mathbf{b}^{(2)}$ are learnable parameters.

$$\boldsymbol{\theta} = Q^{(2)} \cdot \mathbf{z} + \mathbf{b}^{(2)} \quad (7)$$

Here the output vector $\boldsymbol{\theta}$ can be partitioned into four groups serving as the parameters of the Gaussian mixture:

$$\boldsymbol{\theta} = \boldsymbol{\mu} \oplus \boldsymbol{\sigma} \oplus \boldsymbol{\rho} \oplus \boldsymbol{\pi} \tag{8}$$

$$\begin{aligned} \boldsymbol{\mu} &= (\mu_{1,1}, \mu_{1,2}, \mu_{2,1}, \mu_{2,1}, ..., \mu_{M,1}, \mu_{M,1}) \\ \boldsymbol{\sigma} &= (\sigma_{1,1}, \sigma_{1,2}, \sigma_{2,1}, \sigma_{2,1}, ..., \sigma_{M,1}, \sigma_{M,1}) \\ \boldsymbol{\rho} &= (\rho_1, \rho_2, ..., \rho_M) \\ \boldsymbol{\pi} &= (\pi_1, \pi_2, ..., \pi_M) \end{aligned} \tag{9}$$

Since there exists a valid range for each distribution parameter, we enforce these restrictions by converting the above parameters using specific functions. Specifically, since a standard deviation can not be negative, we convert all the standard deviations $\boldsymbol{\sigma}$ using the softplus activation function as follows:

$$\begin{aligned} \sigma_{m,n} &= softplus(\sigma_{m,n}) \\ &= ln(1 + exp(\sigma_{m,n})), m \in [1, M], n \in \{1, 2\} \end{aligned} \tag{10}$$

Also, the valid range of the correlation parameter is $[-1, 1]$, and thus the softsign activation function is used to convert $\boldsymbol{\rho}$.

$$\begin{aligned} \rho_m &= softsign(\rho_m) \\ &= \frac{\rho_m}{1 + |\rho_m|}, m \in [1, M] \end{aligned} \tag{11}$$

Since the sum of the weights of all Gaussian components for a specific tweet should be 1, we normalize the weights $\boldsymbol{\pi}$ using the softmax activation function.

$$\pi_m = \frac{exp(\pi_m)}{\sum_{m=1}^{M} exp(\pi_m)}, m \in [1, M] \tag{12}$$

To summarize, the parameters to be learned in EDGE include the weight matrix $W^{(\ell)}$ in each graph convolutional layer and the connection weights and biases in the fully connected layers, as described in Equation (2) and Equation (7). We train EDGE in an end-to-end manner to maximize the likelihood that geo-tagged tweets are located in their associated locations. We define the loss function as follows:

$$loss = -\sum_{t \in T} ln(\sum_{m=1}^{M} \pi_m \mathcal{N}(\boldsymbol{l}_t | \boldsymbol{\mu}_m, \sum_m)), \tag{13}$$

where $\boldsymbol{l}_t$ is the ground truth location of tweet t, T is a mini-batch of tweets in the training set, and $\sum_{m=1}^{M} \pi_m \mathcal{N}(\boldsymbol{l}_t | \boldsymbol{\mu}_m, \sum_m)$ is the probability density of $\boldsymbol{l}_t$. EDGE fits the parameters of Gaussian mixtures to the tweets in the training set, where each training batch is a random subset of tweets.

IV. Performance Validation

In this section, we empirically show the superiority of EDGE over the state of the art. We begin with the details of the dataset creation, the methods for comparison, and the performance metrics.

A. Dataset Creation

We collected three datasets composed of geo-tagged tweets in the New York Metropolitan Area and the Los Angeles Metropolitan Area, crawled using the Twitter API[3]. The first dataset, redNYMA, consists of 367,259 tweets in the New York Metropolitan Area posted between 08/01/2014 and 12/01/2014. The second dataset, LAMA, consists of 17,025 tweets in the Los Angeles Metropolitan Area posted between 03/12/2020 and 04/02/2020. In the third dataset, COVID-19, we focus on those tweets containing any word in the following set: {"coronavirus", "COVID", "pandemic", "quarantine", "wuhan", "masks", "vaccine", "stayhome", "toilet paper", "social distance" } posted between 03/12/2020 and 04/02/2020 in New York. For each dataset, we selected the first 75% of tweets in the timeline (i.e., those tweets that appear earlier) for training and the remaining for test.

Table II shows the timeline and the entity distributions of the three datasets. We follow the experiment setting in [23] and [7] to choose the spatial granularity and data collection methods. We remark that although our current datasets of tweets are crawled from regions of limited size (i.e., metropolitan areas), our model is general and works for data of any specified region size (e.g., state or country). Note that the output of our model is a mixture distribution of location, so it is applicable to datasets for areas of any size. Neither the parameters of our model nor its output is dependent on a concrete region size.

TABLE II: Overview of dataset

Dataset	Timeline	Entity distribution		
		Training	Test	Intersect
NYMA	08/01/2014-12/01/2014	19,950	9,628	9,516
LAMA	03/12/2020-04/02/2020	2,614	1,223	1,192
COVID-19	03/12/2020-04/02/2020	1,969	883	853

Recall from Section III-A1 that our *entity2vec* module uses Chunker Named Entity Recognizer [28] to extract named entities in tweets. This recognizer is designed for tweets and thus avoids the performance degradation of using traditional NLP tools on the corpus of tweets. The recognizer is reported to have an accuracy of 0.88 on tweets in [28], and to verify the tool's high accuracy on our datasets, we further manually labeled the entities in 100 tweets randomly sampled from our datasets, and this process is repeated 3 times to combat randomness. The recognizer recognized 86.99%, 92.78%, 94.47% entities (note that a tweet may have multiple entities) in the 3 randomly sampled 100-tweet sets, respectively. Note that we have excluded tweets that include no entity in our experiments, which is necessary and reasonable. We remark that only a small percentage of tweets (5.54%) include no entity, and there is truly no meaningful information for geolocation inference in the vast majority of these tweets. An example of such a tweet is This is amazing! which could occur anywhere. In real applications, these tweets are useless for analysis. We have randomly sampled 100 tweets from those without any entity in our datasets, and manually identified the tweets that are irrelevant to geolocation inference. This process is repeated 3 times to combat randomness, and 97%, 99%, 99% tweets in the 3 randomly sampled 100-tweet sets have no information related to geolocation inference, respectively. Since our entity graph is constructed based on the training set, our model only considers those entities that appear in our training set. Therefore, a small percentage of tweets (2.76%) in the test set that include no entities in our entity graph are also excluded.

[3]http://docs.tweepy.org/en/v3.5.0/api.html

To verify the observations described in Section II-A, we also conducted experiments to examine (i) the percentage of tweets mentioning at least one location and (ii) the percentage of tweets mentioning at least both one location and one non-location entity in all our three datasets. The entity recognizer of [28] used by our *entity2vec* module also classifies entities into 10 categories one of which is geolocation. Since the recognizer is reported to have a high classification accuracy [28] and it is infeasible to label all entities in our datasets manually, we directly use the recognizer to estimate the percentage of tweets mentioning locations. The tool reports the percentage of tweets mentioning a location entity to be 30.61%, 45.23%, 43.48% in our three datasets, respectively, while the percentage of tweets mentioning both a location entity and a non-location entity is reported as 29.86%, 33.25%, 39.68%, respectively.

We also crosscheck the results by randomly sampling 100 tweets from each dataset and manually labeling their location entities. On each dataset, this process is repeated 3 times and the average percentage over the 3 runs is reported to combat randomness. The averaged percentage of sampled tweets mentioning a location entity is 47.33%, 43.33%, 57.67% in our three datasets, respectively, and the averaged percentage of tweets mentioning both a location entity and a non-location entity is 45.67%, 37%, 53.33% in three datasets, respectively.

Note that locations are merely a subset of geo-indicative entities. For example, American Airlines is not a location, but it is geo-indicative as has been illustrated in Figure 2(b).

B. Methods for Comparison

We implemented the following state-of-the-art methods for comparison purposes using PyTorch 0.4.0 and ran experiments on a machine with a 2.20 GHz processor and 26 GB RAM, and 16GB Tesla P100 GPU. Each set of experiments in this section was repeated 3 times and all reported results were averaged over 3 runs.

- **LocKDE** [23]: This method predicts tweet locations based on the geographical probability distribution of their terms over a region. Specifically, the probabilities are estimated using kernel density estimation (KDE), where the bandwidth of the kernel function for each term is determined separately according to the location indicativeness of the term. We divided each region into 100×100 grid cells uniformly in our experiments.
- **NAIVEBAYES** [12]: This method treats the geolocation as a classification problem and uses a Naïve Bayes classifier to assign a document to a geographical grid cell by counting the number of words from each cell.
- **KULLBACK-LEIBLER** [12]: instead of counting the number of words from each geographical grid cell, this method assigns a document to a cell by calculating Kullback-Leibler (KL) divergence: it finds the cell whose word distribution best matches the word distribution of the document, i.e., the cell with the minimum KL-divergence.
- **NAIVEBAYES**$_{kde2d}$ [12]: This method replaces count-based estimates in NAIVEBAYES with the respective kernel density estimation. Specifically, a two-dimensional spherical (isotropic) Gaussian kernel is adopted to assign mass to each cell.
- **KULLBACK-LEIBLER**$_{kde2d}$ [12]: This method replaces count-based estimates in KULLBACK-LEIBLER with the respective kernel density estimation. As with NAIVEBAYES$_{kde2d}$, a two-dimensional spherical Gaussian kernel is adopted.
- **Hyper-local** [7]: This method first identifies the geo-specific n-grams by modeling the location distributions of n-grams. The discovered n-grams are then used for geotagging tweets according to the centers of the Gaussian models of the n-grams they contain.
- **UnicodeCNN** [13]: This method uses a Unicode Convolutional Neural Network (UnicodeCNN) for tweet geolocation. First, the model generates features directly from the Unicode characters in the input text. Then a character-level convolutional neural network is designed to predict the coordinates of tweets using a mixture of von Mises-Fisher (MvMF) distribution. In our experiment, we used 100 mixture components, where the components are uniformly distributed in each region. We also tested the other numbers of components but the performance is not better.
- **EDGE**: This is our proposed model. The default length of the embedding for each named entity is 400. The number of graph convolutional layers is 2. The number of Gaussian components is set as 4 by default. The model was trained using an Adam optimizer with a learning rate of 0.01 and a weight decay of 0.01.

C. Metrics for Comparison

For comparison with existing approaches, although EDGE returns a mixture distribution as the prediction result instead of a single location estimate, we derive the estimated location $\hat{l}$ from each returned mixture distribution using Equation (14).

$$\hat{\boldsymbol{l}} = \arg\max_{\boldsymbol{l}} \sum_{m=1}^{M} \pi_m N(\boldsymbol{l}|\boldsymbol{\mu}_m, \sum_m), \tag{14}$$

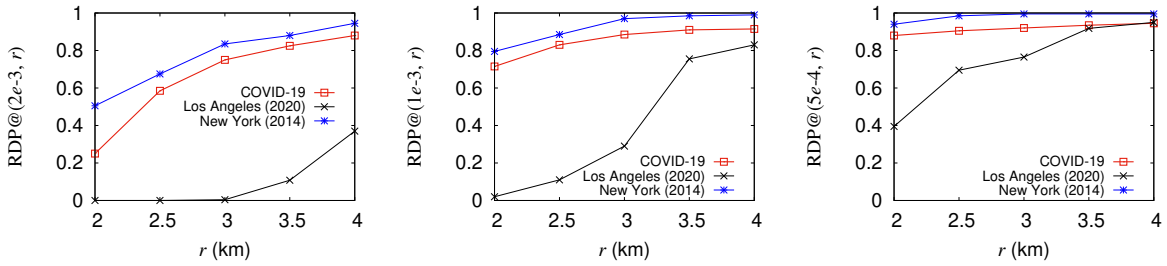
As represented in Equation (14), the location with the maximum sum of the densities for all M distribution components will be returned as the estimated location. Based on such single location conversion, we employed the following metrics [12], [23] for comparison with the state of the art.

- Mean: The mean of the distance (in km) between the true location and the estimated location for all tweets in the test set.
- Median: The median of the distance (in km) between the true location and the estimated location for all tweets in the test set.
- @3km: The fraction of tweets whose true locations fall within 3km from the predicted location.
- @5km: The fraction of tweets whose true locations fall within 5km from the predicted location.

TABLE III: Performance comparison

Data set	Algorithm	Mean(km)	Median(km)	@3km	@5km
New York Metropolitan Area (2014)	LocKDE	8.49	5.38	0.3740	0.4824
	UnicodeCNN	11.71	11.57	0.0120	0.0577
	NAIVEBAYES	8.76	7.60	0.0072	0.0856
	KULLBACK-LEIBLER	8.84	7.62	0.0062	0.0782
	NAIVEBAYES$_{kde2d}$	8.07	6.78	0.0249	0.1644
	KULLBACK-LEIBLER$_{kde2d}$	8.04	6.74	0.0262	0.1633
	Hyper-local	7.90(84%)	4.73(84%)	0.3664(84%)	0.5149(84%)
	EDGE	**6.21***	**2.92***	**0.5219***	**0.6619***
Los Angeles Metropolitan Area (2020)	LocKDE	16.08	9.52	0.3840	0.4209
	UnicodeCNN	29.38	24.29	0.0012	0.0091
	NAIVEBAYES	16.34	7.22	0.1127	0.2232
	KULLBACK-LEIBLER	15.97	6.91	0.1185	0.2483
	NAIVEBAYES$_{kde2d}$	19.35	7.02	0.1211	0.2483
	KULLBACK-LEIBLER$_{kde2d}$	20.89	7.57	0.1133	0.2360
	Hyper-local	14.12(81.31%)	7.01(81.31%)	0.2689(81.31%)	0.3840(81.31%)
	EDGE	**13.27***	**6.20***	**0.3894***	**0.4606***
COVID-19 (New York, 2020)	LocKDE	8.38	4.23	0.4440	0.5270
	UnicodeCNN	15.56	17.10	0.0072	0.0144
	NAIVEBAYES	8.41	4.03	0.4404	0.5559
	KULLBACK-LEIBLER	7.68	3.75	0.4513	0.5776
	NAIVEBAYES$_{kde2d}$	7.78	3.62	0.4585	0.5921
	KULLBACK-LEIBLER$_{kde2d}$	8.51	3.74	0.4476	0.5921
	Hyper-local	6.30	3.73	0.4259	0.6281
	EDGE	**5.38***	**2.74***	**0.5296***	**0.6444***

*** represents the best result**



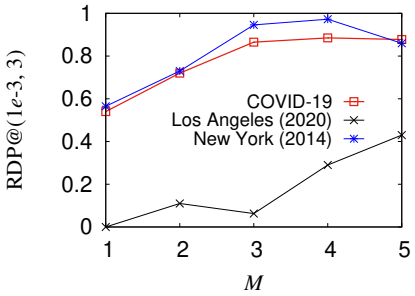
(a) $p = 2e\text{-}3$ (b) $p = 1e\text{-}3$ (c) $p = 5e\text{-}4$

Fig. 5: **The impact of r on the EDGE performance in terms of RDP (M = 4)**

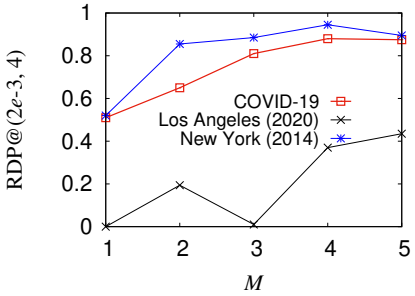
D. Result and Analysis

Table III compares the performance of EDGE and that of the baseline methods. As depicted in Table III, EDGE consistently outperformed the baselines in terms of all the metrics in all three datasets. Specifically, compared with the baselines, EDGE successfully yielded lower mean and median errors and also provided higher scores for @3km and @5km. The superiority of EDGE over the state of the art can be explained by the following:

- All the baseline methods return a single location estimate for geolocation prediction. However, in EDGE, the posting location of a tweet is predicted and returned as a trained Gaussian mixture model, where each distribution is a 2-dimensional multivariate distribution. By doing that, the ambiguous information (uncertainty) can be taken advantage of effectively.
- All the baseline methods focus only on the information in the current tweet itself and fail to take into account the information that is not mentioned in the current tweet but implicitly correlated in other tweets. However, EDGE fuses the named entities that appear together frequently in all the tweets and performs joint inference based on the diffused embedding using graph convolutions.
- In LocKDE, NAIVEBAYES, KULLBACK-LEIBLER, NAIVEBAYES$_{kde2d}$, KULLBACK-LEIBLER$_{kde2d}$ and Hyper-local, only geo-indicative terms actually contribute to predictions, although the non geo-indicative terms are also considered. On the contrary, EDGE performs joint inference based on the diffused embedding of non geo-indicative and geo-indicative entities.
- Hyper-local requires a tweet to have at least one geospecific term to make predictions. Therefore, this method does not work for all the tweets. In Table III, besides the metric scores, we also show the portion of the tweets that can be predicted using Hyper-local when reporting the corresponding results for Hyper-local.
- UnicodeCNN does not support the fine-grained geolocation prediction. UnicodeCNN represents a tweet at the character level using Unicode and makes predictions based on the languages used. Although UnicodeCNN may be effective for the country-level prediction, it does not perform well for the fine-grained geolocation tasks. In each of our datasets, all the tweets were written using the same language (English) and posted in the same city.



(a) p = 1e-3, r = 3



(b) p = 2e-3, r= 4

Fig. 6: **The impact of the number of distribution components on the EDGE performance in terms of RDP**

E. Impact on Radial Density Precision

Note that unlike existing works, EDGE returns a mixture distribution as the prediction result instead of using a single location estimate. Here we first define a new metric, the *radial density precision*, which, given a set of tweets, measures how close the predicted mixture distributions are to their corresponding true locations. The performance of EDGE is then investigated in terms of this new metric.

Definition 1: The radial density precision, RDP@(p, r), is the fraction of the tweets where given a tweet t and its predicted mixture distribution, the radial density integration for a circular region centered at the true location of t with the radius r is at least p.

Given p and r, a higher RDP score indicates a more accurate estimate of the locations for a set of tweets in terms of probabilistic distribution. Figure 5 shows the impact of r on the performance of EDGE in terms of RDP under different p values by varying r from 2 km to 4 km with an interval 0.5 km. Specifically, as shown in Figure 5(a), for New York (2014), EDGE returned a Gaussian mixture where the radial density integration from the true location within 3km is at least 2e-3 for more than 80% of the tweets in the test set. Figure 6 depicts the impact of the number of distribution components, M on the EDGE performance in terms of RDP by varying M from 1 to 5. For New York (2014) and the COVID-19 dataset, the optimal value of M is 4 while for Los Angeles (2020), the optimal value of M is 5. This result also confirms our motivating observation that learning multiple distributions will be more effective than learning only one distribution for tweet geolocation prediction.

F. Ablation Study

To verify the contribution of each component in our model, we have included more baselines:

- **BOW**: To verify the effectiveness of using *entity2vec* and GCN+Attention model, we consider a baseline that represents a tweet as bag-of-words (BOW), i.e., a vector of word frequencies, which is directly input to a dense layer that connects to our Gaussian mixture component.
- **NoGCN**: To verify the effectiveness of our GCN-based entity-diffusion component, this baseline removes GCN and directly feeds entity vectors into our Attention model.
- **SUM**: To verify the effectiveness of our Attention model, this baseline instead directly sums the entity embeddings from our GCN model to compute a tweet embedding. We choose SUM because the number of entities in different tweets can be different, so we need a way to convert them into a tweet representation of fixed length.
- **NoMixture**: To verify the effectiveness of learning a Gaussian mixture with multiple components, this baseline learns only one Gaussian distribution.

TABLE IV: Ablation study result on three data sets

Data set	Method	Mean	Median	@3km	@5km
NYMA	BOW	8.21	4.89	0.2249	0.3569
	NoGCN	7.47	4.15	0.3911	0.5597
	SUM	6.51	4.14	0.3789	0.5754
	NoMixture	9.03	6.78	0.1816	0.3605
	EDGE	**6.21**	**2.92**	**0.5219**	**0.6619**
LAMA	BOW	18.61	14.53	0.0806	0.1644
	NoGCN	18.33	9.91	0.3574	0.3881
	SUM	15.60	8.96	0.3214	0.3988
	NoMixture	16.73	13.18	0.0282	0.0555
	EDGE	**13.27**	**6.20**	**0.3894**	**0.4606**
COVID-19	BOW	9.67	7.29	0.1480	0.3068
	NoGCN	6.04	3.95	0.4444	0.5481
	SUM	6.19	3.66	0.4777	0.5925
	NoMixture	10.85	8.6	0.1263	0.2057
	EDGE	**5.38**	**2.74**	**0.5296**	**0.6444**

As shown in Table IV, replacing any of the components in our model will degrade the performance. Specifically, (1) BOW represents a tweet as separate words, but an entity may include multiple words; in contrast, our *entity2vec* module captures such entity phrases and the entity embeddings reflect the semantic relationships between entities. Thus, EDGE outperforms BOW by a large margin. (2) EDGE also outperforms NoGCN by a large margin, which demonstrates the effectiveness of entity diffusion based on GCN. (3) Compared with sum aggregation, our attention module is able to assign more weights to fine-grained geo-indicative entities than coarse-grained geo-indicative entities, and thus EDGE consistently beats SUM. (4) EDGE also beats NoMixture by a large margin in all metrics, since NoMixture ignores *Observation 2* which is captured by our Gaussian mixture component. Overall, this study justifies the purpose of each component in our model.

V. Demonstration of Use Cases

In this section, we demonstrate two potential use cases of EDGE: (1) predicting the mixture distribution of a single tweet and (2) predicting the most likely locations for a set of related tweets for event dynamics analysis.

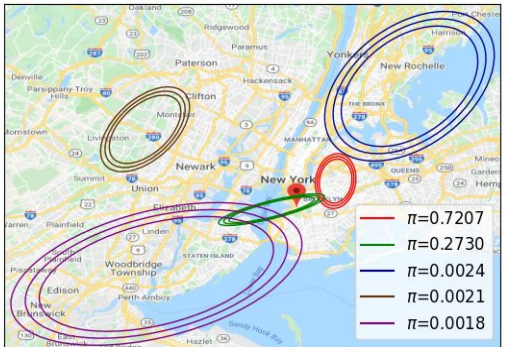


Fig. 7: The mixture distribution returned by EDGE as the prediction result for a given tweet. : *"I think the girls are staging a protest. Theyre done with this self-quarantine business.*, posted on March 22, 2020 in New York.

A. Predicting the Mixture Distribution for a Single Tweet

In this subsection, we use EDGE to predict the location for a specific tweet in terms of probabilistic mixture distribution. Given a non geo-tagged tweet, *"I think the girls are staging a protest. Theyre done with this self-quarantine business.*, EDGE predicted a mixture distribution consisting of five multivariate distributions, as shown in Figure 7.

Result interpretation. As illustrated in Figure 7, each learned distribution is represented by three confidence ellipses, where the three confidence values are 75%, 80%, and 85%, respectively. The weight of each distribution π indicates the importance of the specific distribution. Notice that the sum of the weights of all distributions equals to 1. As shown in Figure 7, the red distribution (centered at East Williamsburg/Brooklyn) and the green distribution (centered at Lower Manhattan) hold a much higher weight than other distribution. As a result, we can infer with high confidence that the possible posting location should be either in East Williamsburg/Brooklyn or Lower Manhattan since the other three distributions have very small weights (0.0024, 0.0021, 0.0018).

Result verification. By examining the tweets posted during the same time duration, we successfully located a protest in the two identified areas (East Williamsburg/Brooklyn and Lower Manhattan). The exact location of the protest is highlighted using the red marker in Figure 7.

B. Predicting the Most Likely Locations for a Set of Tweets

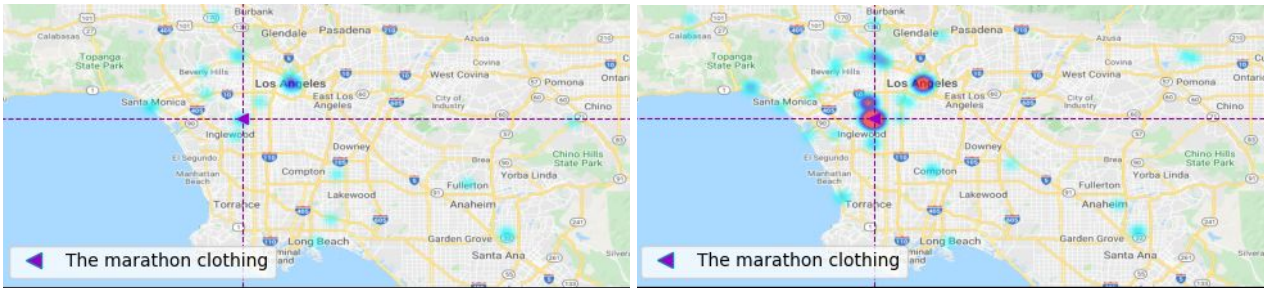
Event dynamics analysis is one of the potential applications that can benefit from EDGE. Observing the locations of tweets related to a specific keyword or hashtag enables us to analyze the dynamics of events in a real-time fashion. Here we use EDGE to predict the most likely locations for a set of non geo-tagged tweets, based on which the dynamics of a specific event can be analyzed. We focus on the two specific events happening in Los Angeles and New York.

- *The death of Nipsey Hussle.* This event is the one-year anniversary of the death of the rapper and activist Nipsey Hussle, who was assassinated on March 31, 2019, outside his store *The Marathon Clothing* in South Central Los Angeles. We first filtered out all the tweets mentioning *Nipsey Hussle* posted from March 12, 2020 to April 2, 2020 in Los Angeles and then used EDGE to predict the locations of the non geo-tagged tweets. Figure 8 shows the heat maps of the tweets mentioning *Nipsey Hussle* from March 12, 2020 to March 30, 2020 and from March 31, 2020 to April 2, 2020, respectively. A burst of tweets can be observed in Figure 8 in several geographical regions close to the place where he was shot.
- *The new colossus festival.* This event is a lower east side music festival scheduled from March 11, 2002 to March 15, 2020 in New York, with 120 bands from around the world playing multiple showcases at seven different venues (*ARLENE'S GROCERY, BERLIN, BOWERY ELECTRIC, LOLA, THE DELANCEY, MOSCOT, PIANOS*). Figure 9(a) depicts the predicted locations of the tweets mentioning *new colossus festival* between March 12, 2020 and March 15, 2020 (during the event) while Figure 9(b) visualizes the predicted locations of the tweets mentioning *new colossus festival* between March 16, 2020 and April 2, 2020 (after the event).

VI. Related works

A. Inference of Tweet Locations

Machine learning techniques have been widely used to infer the locations of tweets [20], [41] . Hulden et al. [12] discretize a geographic area into square cells uniformly and design four models to predict the most probable grid cell for a tweet. First, a classifier-based Naive Bayes is used to estimate the probability of each cell by observing the number of each word in the tweets from a cell. Then Kullback-Leibler (KL) divergence is introduced to find a cell whose word distribution most matches the distribution of the tweet. At last, the count-based estimates are replaced by the kernel density estimation. Ozdikis et al. [23] calculate the probabilities of cells by using kernel density estimation. They determine the bandwidth of the kernel function for each word according to the location indicativeness. Ajao et al. [1] propose a non-uniform grid-based approach using Quadtree spatial partitions. Instead of using individual words, Flatow et al. [7] propose to use n-grams, where the location of a tweet is assigned according to the locations of the provided geo-specific n-grams. Miura et al. [22] develop a simple neural network-based model for geolocation prediction where words are fed into the model by averaging their word embeddings. Thomas et al. [33] use pooling of word embeddings to represent a tweet, which is then forwarded to an LSTM model. In Izbicki's work [13], a convolutional neural network is designed to analyze the text at the character level (Unicode), where a Mixture of von Mises-Fisher (MvMF) distribution is utilized to exploit the Earths spherical geometry. EDGE differs from all the aforementioned works in the following aspects: (1) the inference builds upon mining the correlation between non geo-indicative entities and geo-indicative entities using graph convolution, and (2) each prediction result is represented as a Gaussian mixture



(a) March 12, 2020 to March 30, 2020 (b) March 31, 2020 to April 2, 2020

Fig. 8: A burst of the tweets mentioning *Nipsey Hussle*, who was assassinated on March 31, 2019 outside his store, *The Marathon Clothing*, marked by the purple triangle.

(a) March 12, 2020 to March 15, 2020 (b) March 16, 2020 to April 2, 2020

Fig. 9: The vanishing of the tweets mentioning *new colossus festival*, which was a Lower East Side music festival scheduled from March 11, 2002 to March 15, 2020 in New York at 7 different venues (marked by blue triangles).

instead of specific geographical coordinates, while the most likely location can be also derived from the returned Gaussian mixture distribution. Besides the fine-grained tweet location prediction, there are also studies on tweet location prediction at the country-level [43] and at the city-level [11], [17], which are beyond the scope of this paper.

B. Graph Neural Networks

Graph Neural Networks (GNNs) were first studied in [8] and extended in [30] as a form of the Recurrent Neural Networks (RNN). GNNs take graph structure as input and are able to propagate neighbor information over the underlying graph in an iterative manner for effective node and edge embedding learning. Inspired by the success of the Convolutional Neural Networks (CNNs) in computer vision applications, which extracts higher-level features from images using a sequence of interleaving convolution layers and pooling layers, Graph Convolutional Networks (GCN) become a major focuses of the recent research efforts. Bruna et al. [4] introduce the spectral graph convolutions. In a follow-up work, Defferrard et al. [6] define the ChebyNet which builds on graph convolutions using Chebyshev polynomials to remove the expensive Laplacian eigendecomposition. Kipf et al. [14] further simplify the graph convolution by using the localized first-order approximation.

C. Event Detection and Event Localization

Estimating the exact locations of tweets enables the robust spatio-temporal analysis for a wide variety of track-and-trace applications, such as global event detection [2], [16] and local event detection [39]. Watanabe et al. [37] propose an automatic geotagging method to identify a group of tweets on the same theme. Zhang et al. [40] monitor the continuous tweet streams for real-time event detection by identifying pivots in a query window. TrioVecEvent [39] is a two-step detection scheme where the tweets in a query window are divided into coherent geo-topic clusters, which are considered as candidate events. Event localization [24] focuses on delineating the location of an event. Finding the center and the trajectory of the event location, such as earthquake localization [29] and crime detection [18], can mitigate the damage caused by the event. Allan et al. [2] designed a system to track dynamic events by applying adaptive information filters. Zahra et al. [31] presented a study to localize traffic accidents using Twitter. Different from all the aforementioned works which focus on event detection and event localization, EDGE delivers location predictions that are both accurate and highly interpretable for tweets even without explicit geo-information, which is the key to boosting the effectiveness and efficiency of various downstream event detection and event localization applications.

VII. Conclusion

Our paper presents EDGE, a novel tweet geolocation prediction framework which delivers predictions that are both accurate and highly interpretable without requiring any additional contextual information. Compared with existing works, EDGE has two distinctive features: (1) the inference builds on mining

the correlation between non geo-indicative entities and geo-indicative entities by diffusing their semantic embeddings over the constructed graph neural network and (2) each prediction result is represented as a Gaussian mixture instead of specific geographical coordinates. Extensive experiments using real world tweet datasets validate the superiority of EDGE over the state of the art in terms of all distance-based and POI-based metrics.

ACKNOWLEDGMENT

This research has been funded in part by the U.S. National Science Foundation grants IIS-1618669 (III) and ACI-1642133 (CICI).

REFERENCES

[1] O. Ajao, D. Bhowmik, and S. Zargari. Content-aware tweet location inference using quadtree spatial partitioning and jaccard-cosine word embedding. In *IEEE/ACM 2018 International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2018, Barcelona, Spain, August 28-31, 2018*, pages 1116–1123. IEEE Computer Society, 2018.

[2] J. Allan, R. Papka, and V. Lavrenko. On-line new event detection and tracking. *SIGIR Forum*, 51(2):185–193, 2017.

[3] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

[4] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and locally connected networks on graphs. In *ICLR*, 2014.

[5] Z. Cheng, J. Caverlee, and K. Lee. You are where you tweet: a content-based approach to geo-locating twitter users. In *CIKM*, pages 759–768. ACM, 2010.

[6] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, pages 3837–3845, 2016.

[7] D. Flatow, M. Naaman, K. E. Xie, Y. Volkovich, and Y. Kanza. On the accuracy of hyper-local geotagging of social media content. In *WSDM*, pages 127–136. ACM, 2015.

[8] M. Gori, G. Monfardini, and F. Scarselli. A new model for learning in graph domains. In *IEEE International Joint Conference on Neural Networks*, volume 2, pages 729–734. IEEE, 2005.

[9] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *KDD*, pages 855–864. ACM, 2016.

[10] W. L. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 1024–1034, 2017.

[11] B. Huang and K. M. Carley. A hierarchical location prediction neural network for twitter user geolocation. In K. Inui, J. Jiang, V. Ng, and X. Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 4731–4741. Association for Computational Linguistics, 2019.

[12] M. Hulden, M. Silfverberg, and J. Francom. Kernel density estimation for text-based geolocation. In B. Bonet and S. Koenig, editors, *AAAI*, pages 145–150. AAAI Press, 2015.

[13] M. Izbicki, V. Papalexakis, and V. J. Tsotras. Geolocating tweets in any language at any location. In *CIKM*, pages 89–98. ACM, 2019.

[14] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*. OpenReview.net, 2017.

[15] J. Lee, I. Lee, and J. Kang. Self-attention graph pooling. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 3734–3743. PMLR, 2019.

[16] C. Li, A. Sun, and A. Datta. Twevent: segment-based event detection from tweets. In X. Chen, G. Lebanon, H. Wang, and M. J. Zaki, editors, *CIKM*, pages 155–164. ACM, 2012.

[17] P. Li, H. Lu, N. Kanhabua, S. Zhao, and G. Pan. Location inference for non-geotagged tweets in user timelines. *IEEE Trans. Knowl. Data Eng.*, 31(6):1150–1165, 2019.

[18] R. Li, K. H. Lei, R. Khadiwala, and K. C. Chang. TEDAS: A twitter-based event detection and analysis system. In *ICDE*, pages 1273–1276. IEEE Computer Society, 2012.

[19] Z. Li, Z. Liu, J. Huang, G. Tang, Y. Duan, Z. Zhang, and Y. Yang. MV-GCN: multi-view graph convolutional networks for link prediction. *IEEE Access*, 7:176317–176328, 2019.

[20] F. Melo and B. Martins. Automated geocoding of textual documents: A survey of current approaches. *Trans. GIS*, 21(1):3–38, 2017.

[21] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 3111–3119, 2013.

[22] Y. Miura, M. Taniguchi, T. Taniguchi, and T. Ohkuma. A simple scalable neural networks based model for geolocation prediction in twitter. In *Proceedings of the 2nd Workshop on Noisy User-generated Text, NUT@COLING 2016, Osaka, Japan, December 11, 2016*, pages 235–239. The COLING 2016 Organizing Committee, 2016.

[23] O. Ozdikis, H. Ramampiaro, and K. Nørvåg. Locality-adapted kernel densities for tweet localization. In *SIGIR*, pages 1149–1152. ACM, 2018.

[24] G. Panteras, S. Wise, X. Lu, A. Croitoru, A. Crooks, and A. Stefanidis. Triangulating social multimedia content for event localization using flickr and twitter. *Trans. GIS*, 19(5):694–715, 2015.

[25] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: online learning of social representations. In *KDD*, pages 701–710. ACM, 2014.

[26] A. Rahimi, T. Cohn, and T. Baldwin. A neural model for user geolocation and lexical dialectology. In *ACL*, pages 209–216, 2017.

[27] A. Ritter, S. Clark, Mausam, and O. Etzioni. Named entity recognition in tweets: An experimental study. In *EMNLP*, pages 1524–1534. ACL, 2011.

[28] T. Sakaki, M. Okazaki, and Y. Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In M. Rappa, P. Jones, J. Freire, and S. Chakrabarti, editors, *WWW*, pages 851–860. ACM, 2010.

[29] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Trans. Neural Networks*, 20(1):61–80, 2009.

[30] Z. K. Shahraki, A. Fatemi, and H. T. Malazi. Evidential fine-grained event localization using twitter. *Inf. Process. Manag.*, 56(6), 2019.

[31] W. Shen, Y. Liu, and J. Wang. Predicting named entity location using twitter. In *ICDE*, pages 161–172. IEEE Computer Society, 2018.

[32] P. Thomas and L. Hennig. Twitter geolocation prediction using neural networks. In *Language Technologies for the Challenges of the Digital Age - 27th International Conference, GSCL 2017, Berlin, Germany, September 13-14, 2017, Proceedings*, volume 10713 of *Lecture Notes in Computer Science*, pages 248–255. Springer, 2017.

[33] twitterdev. Tweet geospatial metadata, 2020. [Online; accessed 22-May-2020].

[34] S. Wang, M. Gong, Y. Wu, and M. Zhang. Multi-objective optimization for location-based and preferences-aware recommendation. *Inf. Sci.*, 513:614–626, 2020.

[35] K. Watanabe, M. Ochi, M. Okabe, and R. Onai. Jasmine: a real-time local-event detection system based on geolocation information propagated to microblogs. In *CIKM*, pages 2541–2544. ACM, 2011.

[36] Wikipedia contributors. Twitter — Wikipedia, the free encyclopedia, 2020. [Online; accessed 22-May-2020].

[37] C. Zhang, L. Liu, D. Lei, Q. Yuan, H. Zhuang, T. Hanratty, and J. Han. Triovecevent: Embedding-based online local event detection in geo-tagged tweet streams. In *KDD*, pages 595–604. ACM, 2017.

[38] C. Zhang, G. Zhou, Q. Yuan, H. Zhuang, Y. Zheng, L. M. Kaplan, S. Wang, and J. Han. Geoburst: Real-time local event detection in geo-tagged tweet streams. In *SIGIR*, pages 513–522. ACM, 2016.

[39] X. Zheng, J. Han, and A. Sun. A survey of location prediction on twitter. *CoRR*, abs/1705.03172, 2017.

[40] X. Zhu, Z. Ghahramani, and J. D. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, pages 912–919. AAAI Press, 2003.

[41] A. Zubiaga, A. Voss, R. Procter, M. Liakata, B. Wang, and A. Tsakalidis. Towards real-time, country-level location classification of worldwide tweets. *IEEE Trans. Knowl. Data Eng.*, 29(9):2053–2066, 2017.