ABSTRACT

Title of Dissertation: LOCAL NEWS AND EVENT DETECTION IN TWITTER

Hong Wei
Doctor of Philosophy, 2020

Dissertation Directed by: Professor Hanan Samet
Department of Computer Science

Twitter, one of the most popular micro-blogging services, allows users to publish short messages on a wide variety of subjects such as news, events, stories, ideas, and opinions, called *tweets*. The popularity of Twitter, to some extent, arises from its capability of letting users promptly and conveniently contribute tweets to convey diverse information. Specifically, with people discussing what is happening outside in the real world by posting tweets, Twitter captures invaluable information about real-world news and events, spanning a wide scale from large national or international stories like a presidential election to small local stories such as a local farmers market. Detecting and extracting small news and events for a local place is a challenging problem and is the focus of this thesis. In particular, we explore several directions to extract and detect local news and events using tweets in Twitter: a) how to identify local influential people on Twitter for potential news seeders; b) how to recognize unusualness in tweet volume as signals of potential local events; c) how to overcome the data sparsity of local tweets to detect more and smaller undergoing local news and events. Additionally, we also try to uncover implicit

correlations between location, time, and text in tweets by learning embeddings for them using a universal representation under the same semantic space.

In the first part, we investigate how to measure the spatial influence of Twitter users by their interactions and thereby identify the locally influential users, which we found are usually good news and event seeders in practice. In order to do this, we built a large-scale directed interaction graph of Twitter users. Such a graph allows us to exploit PageRank based ranking procedures to select top local influential people after innovatively incorporating in geographical distance to the transition matrix used for the random walking.

In the second part, we study how to recognize the unusualness in tweet volume at a local place as signals of potential ongoing local events. The intuition is that if there is suddenly an abnormal change in the number of tweets at a location (e.g., a significant increase), it may imply a potential local event. We, therefore, present DeLLe, a methodology for automatically **De**tecting **L**atest **L**ocal **E**vents from geotagged tweet streams (i.e., tweets that contain GPS points). With the help of novel spatiotemporal tweet count prediction models, DeLLe first finds unusual locations which have aggregated an unexpected number of tweets in the latest time period and then calculates, for each such unusual location, a ranking score to identify the ones most likely to have ongoing local events by addressing the temporal burstiness, spatial burstiness, and topical coherence.

In the third part, we explore how to overcome the data sparsity of local tweets when trying to discover more and smaller local news or events. Local tweets are those whose locations fall inside a local place. They are very sparse in Twitter, which hinders the detection of small local news or events that have only a handful of tweets. A system, called Firefly, is proposed to enhance the local live tweet stream by tracking the tweets of

a large body of local people, and further perform a locality-aware keyword based clustering for event detection. The intuition is that local tweets are published by local people, and tracking their tweets naturally yields a source of local tweets. However, in practice, only 20% Twitter users provide information about where they come from. Thus, a social network-based geotagging procedure is subsequently proposed to estimate locations for Twitter users whose locations are missing.

Finally, in order to discover correlations between location, time and text in geotagged tweets, e.g., "find which locations are mostly related to the given topics" and "find which locations are similar to a given location", we present LEGO, a methodology for **LE**arning embeddings of **GEO**tagged tweets with respect to entities such as locations, time units (hour-of-day and day-of-week) and textual words in tweets. The resulting compact vector representations of these entities hence make it easy to measure the relatedness between locations, time and words in tweets. LEGO comprises two working modes: cross-modal search (LEGO-CM) and location-similarity search (LEGO-LS), to answer these two types of queries accordingly. In LEGO-CM, we first build a graph of entities extracted from tweets in which each edge carries the weight of co-occurrences between two entities. The embeddings of graph nodes are then learned in the same latent space under the guidance of approximating stationary residing probabilities between nodes which are computed using personalized random walk procedures. In comparison, we supplement edges between locations in LEGO-LS to address their underlying spatial proximity and topic likeliness to support location-similarity search queries.

LOCAL NEWS AND EVENT DETECTION IN TWITTER

by

Hong Wei

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2020

Advisory Committee:
Professor Hanan Samet, Chair/Advisor
Professor David Mount
Professor Dinesh Manocha
Professor Leila De Floriani
Professor Udaya Shankar

Acknowledgments

I would like to express my deepest appreciation to my advisor, Prof. Hanan Samet, for his support, guidance, encouragement and patience throughout my Ph.D. study at the University of Maryland. Without him, I would not have been able to complete this research. I came to work with Hanan on a legendary project, TwitterStand, in his group in Spring 2015. My research problem was to try to detect as many small news as possible in that platform. After a while, we realized that we were suffering extreme data sparsity in Twitter which curb the conduct of further research. It was then only Hanan's persistent inquiries sent out to many people that finally helped us finally gain access to full Twitter data access through Microsoft Research. Even for a short time, such a full access to Twitter data greatly boosted my research. I am also extremely grateful to Hanan for his countless nights working on reviewing, editing and revising my research manuscripts. Often, he stayed up past midnight to finish editing my papers. There were also several times that I felt frustrated with my research progress. Henan has always been always there to listen to me, encourage me and more importantly, be patient with me. His advice has also been of great benefit to me in my personal life like job seeking and making a living. So I really had great pleasure of working with Prof. Hanan Samet.

I would like to extend my sincere thanks to Prof. David Mount, Prof. Dinesh Manocha, Prof. Udaya Shankar and Prof. Leila De Floriani for serving on my thesis committee and additionally Prof. Larry Davis for serving on my proposal committee. Many thanks for their valuable feedback and suggestions about my work. I would also like to thank Dr. Jagan Sankaranarayanan for the insightful discussions.

I would like to express my special gratitude to Prof. Leila De Floriani for her generous help in using her Apache Spark cluster. Thanks should also go to Dr. Sudipta Sengupta, Dr. Jin Li and Dr. John Krumm for providing access to the valuable Twitter data. I would like to acknowledge the help of UMIACS computing staff for their constant technical support.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Twitter, one of the most popular micro-blogging service, allows users to promptly and conveniently contribute content on a wide variety of subjects via tweets. A tweet is usually a short text message limited to at most 280 (140 before November, 2017) characters and is often posted to convey diverse information on various topics by Twitter users. Each user in Twitter, can subscribe to another user to receive the contents that the latter publishes. As a result, with people discussing what is happening outside in the real world by posting tweets, an invaluable amount of information on the real world news or events is hidden in Twitter. Extracting such information, especially local news and events, exerts a measure of influence over various applications such as situation awareness and disaster recovery. For example, people can acquire the latest information about such local activities in the town in which they are living such as sale promotions at local supermarkets and the opening of a farmers market, and thereby enhance their daily lives. It can also be helpful for commuting alarms by reporting real-time traffic jams or accidents. In such cases, after learning what is happening, commuters can actively make a decision to bypass the congested road segments or avoid the accident sites when planning the routes.

However, detecting and extracting local news and events for a local place is a challenging problem. First, most of the tweets are usually noisy in the sense that they contain

little information on local news and events. Therefore, it is important to identify a source of good quality tweets for performing news and event detection. Second, a local event is often described as an unusual activity that happens at specific time and place. This encourages the adoption of unusualness measures such as a burst in the number of tweets as a key factor in event detection. Twitter, however, is often plagued with misleading phenomena which also exhibit burstiness of tweets, such as people's inclination of tweeting about daily life activities and sometimes occasional global events as well as Twitter bots posting noisy advertising tweets. Isolating out the specific unusualness representing local events is therefore a challenging task. Third, the publicly accessible local tweets are very sparse. For example, the Twitter Sample API stream[1], which is open to public to access its real-time tweets, samples only around 1% of all tweets. Unfortunately, only 1% of the tweets contain embedded GPS coordinates (geotagged) which allow us to pinpoint the local place where they are posted from. This greatly limits researchers' opportunities to detect more and smaller local news and events because such news or events may only span a very limited number of tweets.

Apart from detecting local events at a location, it is also important to have descriptive profiles for locations of interest and to measure the similarities between locations if possible. These are usually composed of queries like "find which locations are mostly related to the given topics" and "find which locations are similar to a given location". Traditionally, people represent locations and topics in different forms simply because they are from different domains. For example, locations are usually specified as points or polygon in the geometric domain, while topics are textual words and character sequences in the

[1] https://dev.twitter.com/streaming/reference/get/statuses/sample

natural language domain. Under such representation forms, it is hard to perform cross-domain searches without building associative data structures and hard to perform location similarity searches with respect to semantic meanings too.

In this dissertation, we discuss several directions to mitigate the challenges in detecting and extracting local news and events, and meanwhile investigate applications of graph embedding techniques in uncovering correlations between location, time and topic in human's urban activities hidden in geotagged tweets. To be specific, we try to find answers to the following questions:

1. how to identify locally influential people on Twitter for potential news seeders,
2. how to recognize unusualness in tweet volume as signals of potential local events,
3. how to overcome the data sparsity of local tweets to enable detection more undergoing local news and events,
4. how to universally represent spatial, temporal and textual entities in the geotagged tweets in the same semantic space.

1.1 Measuring Spatial Influence of Twitter Users by Interactions

The three ways of interactions in Twitter–*retweet*, *reply*, and *mention*–comprise of a latent dynamic information flow network between users, which can be utilized to determine influential users. Chapter 2 focuses on determining which Twitter users have great influence on a query location Q in the sense that they are assumed to provide information that is of sufficient interest to prompt people at Q to interact with them. Note that

an influential Twitter user who is of great influence on Q may not be necessarily from Q. Therefore, we first define ***generalized influential Twitter users*** regardless of whether their location is known or not, meaning that such generalized influencers on Q can be either from inside Q, or outside Q, or even unknown. A more interesting subset of generalized influencers is the ones whose location is in Q, and termed as ***local influential Twitter users***. One potential application of finding local influencers (e.g., local news media) is to detect local events by tracking their tweets.

Using a large amount of data collected from Twitter, we first build a large-scale directed interaction graph of Twitter users and present an analysis of the geographical characteristics of the edges in this interaction graph and make several interesting observations. Based on these findings, we propose two versions of PageRank that measure spatial influence on the interaction graph: Edge-Locality PageRank (ELPR), and Source-Vertex-Locality PageRank (SVLPR), which takes into account the spatial locality of edges and the spatial locality of source vertices in edges, respectively. In addition, a Geographical PageRank (GPR) is also proposed trying to incorporate both of these two factors together. In the experimental evaluation, we examine the effectiveness of the proposed methods with respect to 3 different US cities "Boston, MA", "Bristol, CT" and "Seattle, WA". The results show that our algorithms outperform baseline approaches including the topological network metrics and the original PageRank. More importantly, we also explored the possibility of using local influential Twitter users as potential news seeders and showed that some types of users have high credibility in outputting tweets about local places.

1.2 Detecting Latest Local Events from Geotagged Tweet Streams

Geotagged tweet streams contain invaluable information about the real-world local events like sports games, protests and traffic accidents. Timely detecting and extracting such events may have various applications but yet unsolved challenges. In Chapter 3, we present DeLLe, a methodology for automatically **De**tecting **L**atest **L**ocal **E**vents from geotagged tweet streams. With the help of novel spatiotemporal tweet count prediction models, DeLLe first finds unusual locations which have aggregated unexpected number of tweets in the latest time period and thereby imply potential local events. Next, DeLLe calculates, for each such unusual location, a ranking score to identify the ones most likely to have ongoing local events by addressing the temporal burstiness, spatial burstiness and topical coherence. Furthermore, DeLLe infers an event candidate's spatiotemporal range by tracking its event-focus point, which essentially reflects the most recent representative occurrence site. Finally, DeLLe chooses the most influential tweets to summarize local events and thereby presents succinct but yet representative descriptions. We evaluate DeLLe on the city of Seattle, WA as well as a larger city of New York. The results show that the proposed method generally outperforms competitive baseline approaches.

1.3 Enhancing Local Live Tweet Stream to Detect News

Twitter captures invaluable information about real-world news, spanning a wide scale from large national/international stories like a presidential election to small local stories such as a local farmers market. Extracting more and smaller news for a local place

is a challenging problem and the focus of this chapter. The main challenge lies in the data sparsity of local tweets, which makes small local stories typically harder to detect compared to national stories in the sense that there may be just a handful of tweets about a local story. In Chapter 4, a system, called Firefly, is proposed that overcomes the data sparsity and captures thousands of local stories per day from a metropolitan area (e.g., Boston). The key idea lies in combining the enhancement of a local live tweet stream in Twitter, the identification of "locality-aware" keywords, and using these keywords to cluster tweets. Experiments show that the proposed system has significantly higher recall over a set of representative local news agencies, and at the same time, outperforms the baseline approach TwitterStand. More importantly, the results also demonstrate that our system, by utilizing the enhanced local live tweet stream, discovers much more local news than the methods working only on geotagged tweets, i.e., those with embedded GPS coordinate values.

1.4 Learning Embeddings of Spatial, Textual and Temporal Entities in Geotagged Tweets

With online social networks being extended to geographical space, location context plays a key role in many applications such as local event detection and location recommendation. Geotagged tweets in Twitter serve as an invaluable source to understand people's activities in urban space. Analyzing geotagged tweets to identify implicit contexts between location, time and text is an interesting problem. In Chapter5, we present LEGo, a methodology for **LE**earning embeddings of **Geo**tagged tweets with respect to enti-

ties such as locations, time units (hour-of-day and day-of-week) and textual words in tweets. The resulting compact vector representations of these entities hence make it easy to perform searches like "find which locations are mostly related to the given topics" and "find which locations are similar to a given location". LeGo comprises two working modes: cross-modal search (LeGo-CM) and location-similarity search (LeGo-LS), to answer these two types of queries accordingly. In LeGo-CM, we first build a graph of entities extracted from tweets in which each edge carries the weight of co-occurrences between two entities. The embeddings of graph nodes are then learned in the same latent space under the guidance of approximating stationary residing probabilities between nodes which are computed using personalized random walk procedures. In comparison, we supplement edges between locations in LeGo-LS to address their underlying spatial proximity and topic likeliness to support location-similarity search queries. We evaluate LeGo on datasets of New York City and Los Angeles, showing that the proposed method generally outperforms competitive baseline approaches.

Chapter 2: Measuring Spatial Influence of Twitter Users by Interactions

2.1 Introduction

Twitter, one of the most popular micro-blogging services, allows users to publish short messages, called *tweets*, on various subjects. In Twitter, each user, can subscribe to another user to receive the contents the latter publishes, through "following" the latter. In so doing, the former user becomes one of the latter's "followers", and the latter becomes one of the former's "friends". The definition of making a "friend" on Twitter is different from establishing a reciprocal "friend" relationship in other social network services like Facebook, because such a "following" operation in Twitter completes without requiring the user being followed to grant permission nor follow back the user who initiates the "following", which generates a directed follower-following relationship. With users as vertices, and their directed relations of following and being followed as edges, a social network in Twitter builds up.

Such a direct follower-following social graph, however, is not always available due to the difficulties imposed by the Twitter API rate limits in obtaining access to a complete social network link structures in current Twitter. On the other hand, Twitter offers a few more dynamic ways to interact with other people such as *retweet, reply, mention*, which have been utilized in some works to determine influential Twitter users such as [59, 16].

Different from the absence of follower-following relationship, interactions are embedded in the meta-data of tweets and need no further request to Twitter API once a tweet dataset is ready. Therefore, one of the popular strategies is to first rebuild a social network from interactions and then determine influential users in the interaction graph [57, 29, 120, 89, 14, 41, 53, 45, 135, 46].

However, few of the existing approaches to determining the influence of Twitter users on the social network gives credits to where they are from and furthermore how close they are. Therefore, in this chapter, we are focusing on answering the following question: ***Given a query location Q, which Twitter users have great influence on it?*** We refer Q to a circular region defined by a geographical center point l_q and a radius ϵ.

We consider a Twitter user to be spatially influential at Q if his authority has been endorsed by the local people from Q. We deem the interactions (*retweet, reply, mention*) one user initiates to another as his endorsement of the latter's authority. In essence, the more people from a location endorse a Twitter user, more spatially influential he becomes on that location. In this definition, we don't require a Twitter user to have to be from location Q (e.g., his home location falls within Q) to be considered influential there. In such a sense, the infuential Twitter users are termed as ***generalized influential Twitter users*** on Q. The more interesting subset of the influential Twitter users on Q is the ones who are also from location Q, termed as ***local influential Twitter users*** on Q.

Solving this problem is beneficial to many applications like targeted advertising, political campaign, trend analysis [80], and location-based recommendation [9]. In particular, finding local influential Twitter users also has the potential to discovering local news and events. For example, the Twitter accounts representing local news media usu-

ally cover and deliver information in their posted tweets regarding what is happening at a location and can be utilized as news seeders [102, 47] to help news detection [18, 101]. To test the viability of local influential Twitter users in such applications, we examined the tweets published by a set of top influential Twitter users in Boston for a week. The results show that more than half of the tweets are considered local by virtue of discussing content relevant to the local place, and the ratio of local tweets goes higher if we only consider specific group of users such as news person (e.g., News Media and Reporter) and sports person (e.g., Sport Player and Sport Team).

In this chapter, we first build a large-scale directed interaction graph. An intuitive solution to finding influential Twitter users then is to append a post-processing location filter step after finding influential people in general. For example, one can rely on the indegrees of vertices in the interaction graph or apply the PageRank schema to yield a ranking order for Twitter users regarding their influence, and then select the ones who fall within Q and identify them as influential Twitter users on Q.

Our proposed methods improve over this strategy by additionally considering spatial locality in the edges and its source vertices respectively. Specifically, by emphasizing on spatially local edges (applying a exponential distance-decay on the edges), i.e., whose two vertices have smaller geographical distances, our method Edge-Locality PageRank (ELPR) more effectively find ***local influential Twitter users*** than the network metrics like indegree and the original PageRank. By focusing on the edges whose source vertices are spatially local to the query location center l_q, our method Source-Vertex-Locality PageRank (SVLPR) doesn't rely on a post-processing location filter step and thereby also captures the Twitter users who are of great influence on but not necessarily from the loca-

tion Q. The experiments also show that SVLPR outperforms its indegree-based baseline approach. Moreover, our hybrid method Geographical PageRank (GPR) attempts to bring geographical distances among Twitter users into the process of finding spatially influential Twitter users, which improves over PageRank by taking into consideration both link structure and geographical distance while propagating influence among users.

The rest of this chapter is organized as follows. In Section 2.2, we review related work. In Section 2.3, we describe the dataset we are using, along with an interaction graph built from it. In Section 2.4, we first present our two methods to determine local and generalized influential Twitter users, respectively and additionally propose a hybrid method trying to combine them. Section 2.5 describes the experimental evaluation of our methods. Concluding remarks are drawn in Section 2.6.

2.2 Related Work

There has been much work on identifying the influential users in the social networks, Twitter in particular. A few of recent surveys Gayo-Avello [31], Kardara et al. [52] and Riquelme and González-Cantergiani [95] provide comprehensive summarizations on the different techniques regarding identifying influential Twitter users.

In the Twitter social graph, an intuitive way to measure a user's influence is by his number of followers, i.e., the indegree of the vertex representing this user. Although it is suggested that Twitter itself is also using the same strategy [119], the metric of indegree isn't always able to reflect the real happening information flowing patterns in Twitter [59, 16, 119, 64, 124] and therefore is limited in discovering influence patterns. For exam-

ple, Cha et al. [16] has reported the bias in identifying influential Twitter users by solely depending on indegree. One of the factors that may contribute to weakening the effect of indegree is the courtesy of following back. That is, a user being followed might follow back just for the sake of courtesy, and such a follower-following relationship does not carry the strong indication of information and influence flow [119]. In contrast, some practices of interaction such as mention and retweet in Twitter empower users to spread information beyond the capability of the existing follower-following network as the communication channels. That is, a user may retweet from or mention people who are not their immediate followers or people he is not following. This power allow users to collectively determines the importance of information by choosing to which information spread and thereby assume more duty in disseminating important information [16]. Consequently, such dynamic interactions are further exploited to determine influential Twitter users. For example, Kwak et al. [59], one of the earliest effort to quantitatively study the topological characteristics of Twitter's social network, have studied ranking users by the number of retweets and find that it is quite dissimilar with ranking users by the number of their followers. This indicates a gap in influence inferred from the number of followers and that from the popularity of one's tweets [59], and further demonstrates a difference between Twitter's static follower-following network structure and its dynamic interaction between users, e.g., retweeting. Furthermore, some derivatives of interactions have also been investigated like the normalized or averaged retweets and mentions by total tweets or total followers [16], which might yield a slightly different influence ranking result. Nevertheless, such statistical properties of interaction don't attribute credibility to the phenomenon that a user's influence might be propagated to distant users that are not directly connected

to (or interacting with him) on social networks.

On the other hand, there have been some works borrowing PageRank from ordering webpages in the connected World Wide Web [90] to ranking users in Twitter directed social network graph [59, 119, 124, 37, 107, 34]. PageRank improves over previous measures that are based directly on simple metrics in the sense that it assumes that by following a user, the followers are implicitly conferring some influence to him and then iteratively propagates a user's influence through the whole social graph. Thereafter, a lot of adaption on PageRank have been made to address more aspects of Twitter users in addition to the sole follower-following structures [119, 37, 107, 34, 124]. The above solutions, however, heavily rely on the follower-following network in Twitter, and thereby make the acquisition of necessary data extremely restricted to the Twitter API rate limits [1]. To avoid the limitations of Twitter API in obtaining the follower-following network structure and meantime capture the dynamic information flow between Twitter users, the interactions like *retweet*, *reply*, and *mention* have been utilized trying to build similar social graphs [57, 29, 120, 89, 14, 41, 53, 45, 135, 46]. With these graphs, the iterative influence propagation schema such as PageRank can be applied. For example, MultiRank [29] builds different graphs for different interactions such as *retweet* and *reply* respectively, in a given topic. Then the ranking of topical influential users is determined by coupling the PageRank scores in each of the graphs by performing a combined random surfer walk on the multi-relational network via assigning different weights to each graph.

Another strategy of finding influential users in social networks is by *influence maximization*, which is to select k users to maximize the expected number of users being

[1] https://dev.twitter.com/rest/public/rate-limiting

influenced [55]. Location-aware influence maximization methods are also proposed such as Bouros et al. [11] and Li et al. [67], to find top k influential users in a geographical region. Although they have a similar problem context to us regarding identifying local influential Twitter users, our work differs from them by addressing people's geographical proximity (distance) while propagating each other's influence through the social network. Moreover, our algorithms inherently bring ranking orders to all the users by running only once, which is beneficial over their work for queries with varying value of k.

With regard to incorporating geographical proximity between graph vertices into PageRank, the work of Chin and Wen [21] is the most related to ours. They solve a different problem to capture spatial concentration of population movement by running on a geospatial network, where each vertex represents a unique geographical place and edges form between places within reachability in a given travel time. In so doing, each vertex comes with geographical coordinates. This, however, does not always hold for the Twitter social graph as the location of some users might be unknown, as well as their geographical distances to others. We tackle this problem for such users by introducing geographical tendency which utilizes location distributions of their friends.

2.3 Build an Interaction Graph $\mathcal{G}$

2.3.1 Dataset

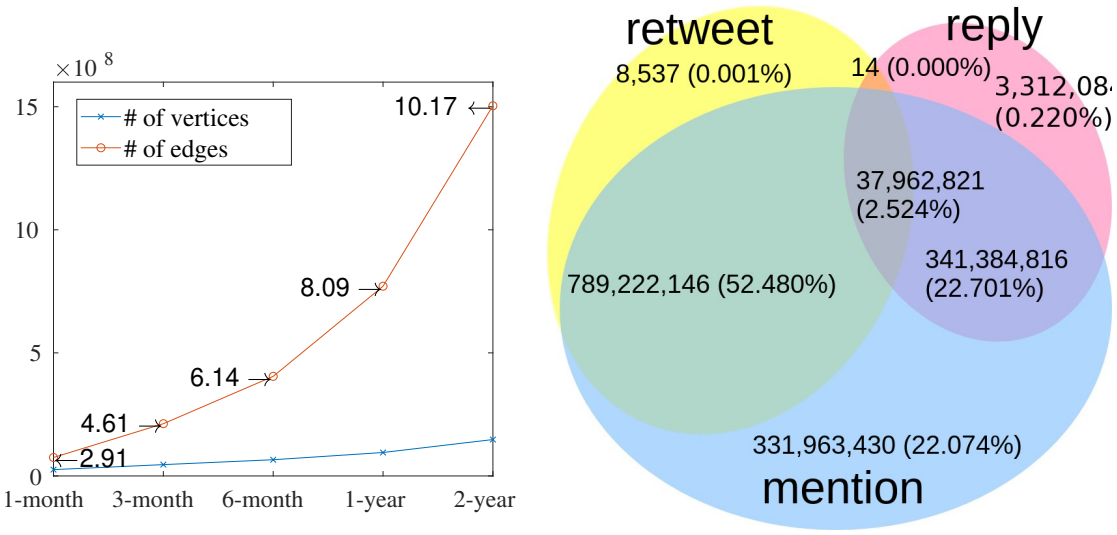
Like the directed follower-following relationships, each Twitter interaction between users has an underlying direction too, pointing from the user who actively initiates the action of *rewtweet*, *reply* or *mention* to the other one. Therefore, during building up the

interaction graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a directed edge e_{ij} from user v_i to user v_j is constructed and added to $\mathcal{E}$ if there exists at least one interaction pointing from v_i to v_j, both of whom are also added to $\mathcal{V}$ as vertices. For convenience, in a directed *edge* e_{ij} of $\mathcal{G}$, we call v_i one of v_j's *i-followers*, i.e., *interaction followers*, and vice versa, v_j one of v_i's i-friends, i.e., *interaction friends*.

Our dataset consists of 5,515,214,722 tweets collected between January 2015 and December 2016[2]. In these tweets, there are 1,097,055,845 *retweet* interactions, 587,550,806 *reply* interactions and 2,147,483,647 *mention* interactions. We therefore build an interaction graph $\mathcal{G}$ of 1,503,853,848 directed edges and 147,842,352 users as vertices. Note that in building $\mathcal{G}$, multiple interactions happening on a same directed edge will be aggregated together and don't create additional edges. $\mathcal{G}$ is relatively sparse in comparison to the one reported in [59]–a complete Twitter social network by July 2009, though $\mathcal{G}$ is on the similar scale in terms of the number of edges. For example, the ratio of number of edges over the number of vertices in $\mathcal{G}$ is 10.17 while the one in [59] has a ratio of 35.25 with a total number of 1,468,365,182 edges. Although collecting more tweets for a longer time will increase the ratio, such an increase is at a very slow speed as showen in Figure 2.1a.

Regarding the contribution to building edges in $\mathcal{G}$, as shown in the Venn diagram Figure 2.1b, *mention* is the most significant by covering 99.779% of edges, while *retweet* 55.005% and *reply* only 25.445%. The Venn diagram also shows that *mention* is covering most of the edges constructed from *retweet* and *reply*, indicating that most of the users who retweet or reply to each other also mention each other. This, however, is not the case

[2] https://dev.twitter.com/streaming/reference/get/statuses/sample



(a) (b)

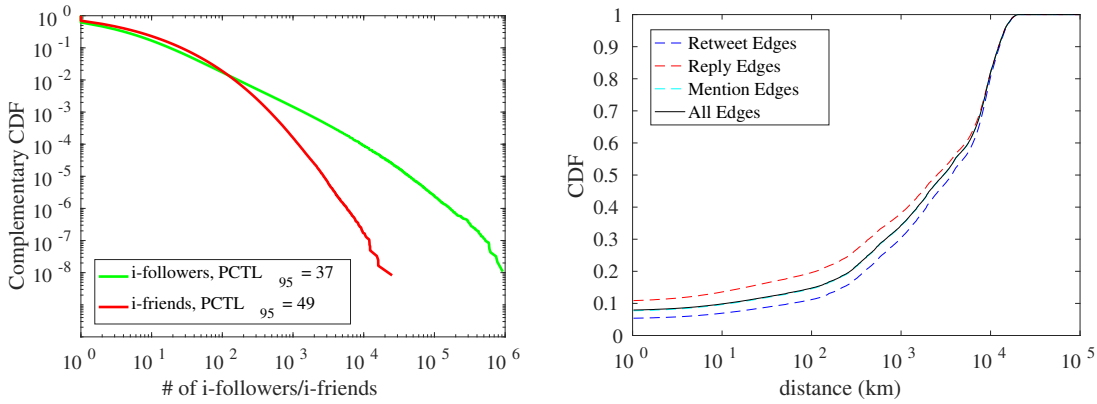
Figure 2.1: (a) The number of vertices and edges of interaction graphs built by using 1, 3, 6, 12 and 24-month of tweets. (b) Venn diagram of edges in $\mathcal{G}$ by *retweet*, *reply* and *mention*, respectively.

for *retweet* and *reply*, who only share 2.524% edges in common.

Regarding the distributions of indegree/outdegrees in the interaction graph $\mathcal{G}$, we plot the complementary cumulative distributions of the number of i-followers/i-friends each Twitter user has in Figure 2.2a, which shows a power-law pattern. Note that there is a huge gap between the maximum number of i-followers and the maximum number of i-friends a Twitter user has. This makes sense because a very popular Twitter user might receive interactions from millions of people but is very unlikely to actively send interactions to such a huge amount of people.

2.3.2 Twitter User Locations

In Twitter, there are two sources for knowledge of a user's location: the geographical coordinates in his GPS-tagged tweets and the home location in his profile – also



(a) (b)

Figure 2.2: (a) Distributions of the i-follower/i-friends a Twitter user has. (b) Distribution of the distances of edges.

termed as *profile-location*. The profile-location is often in the form of place names like "College Park, MD" and can be aligned with databases like GeoNames[3] to decode its geographical latitude/longitude coordinates [73, 72]. In order to assign a unique pair of latitude/longitude coordinates, for users who have multiple pairs of geographical coordinates, we compute the *L1-multivariate median* which essentially finds a point having the minimum sum of distances to a given set of points Z [24]:

$$\operatorname*{argmin}_{z'} \sum_{z \in Z} distance(z', z) \tag{2.1}$$

After discarding coordinates of (0.0, 0.0) that are likely caused by GPS malfunction, we then have 54,428,031 (36.8%) Twitter users having geographical coordinates, and 4,933,524 of them from GPS-tagged tweets. Correspondingly, 625,186,580 (41.6%) edges of $\mathcal{G}$ have both their vertices with geographical locations.

Next, we use Twitter users/lookup API to download the profile information for Twit-

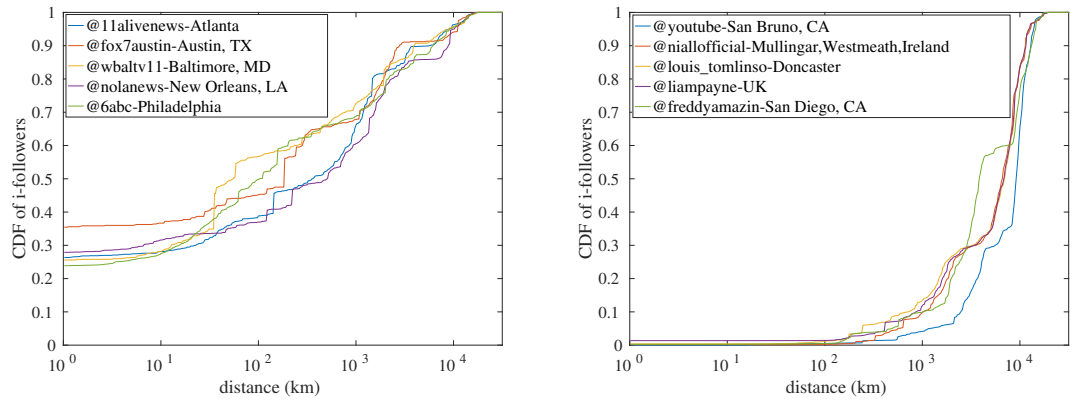
[3] https://www.geonames.org

ter users whose profile-location have not been exposed in our dataset. These users are usually the ones who had appeared in our dataset but were only being replied to or mentioned by others and thereby lacking the profile location information. After downloading profiles for these users, we get the locations of additional 12,018,353 users, making a total 66,446,384 (44.9%) users have geographical locations. This makes 756,737,542 (50.3%) edges in $\mathcal{G}$ have both of their vertices with geographical locations.

Furthermore, there have been methods proposed trying to estimate locations for Twitter users whose locations are unknown such as [24]. We therefore investigate the effect of utilizing such a geotagging procedure in Section 2.5.5.

Geographical Distribution of Edge Distances In this chapter, ***the distance of an edge in*** $\mathcal{G}$ refers to the geographical distance between its two Twitter users. For the edges both of whose two vertices have geographical coordinates to calculate a distance, we plot their distance distribution in Figure 2.2b, which shows that interactions happen over various distances and not always over shorter distances and should receive different geographical considerations.

We also examined the distributions of the distances from several local news agencies (who are considered as local influential Twitter users) to their i-followers as shown in Figure 2.3a. It shows that the i-followers of local news agencies are more aggregated around the place where these agencies are located, indicating their influence are more revealed at local places. This inspires us to look at only spatially local edges in $\mathcal{G}$ to find local influential Twitter users. In contrast, as shown in Figure 2.3b, the i-followers of the Twitter users who are considered influential by having largest indegrees are scattered over places, nationwide and even worldwide, indicating that their influence is not just fo-



(a) (b)

Figure 2.3: (a) CDFs of i-followers for 5 local news agencies over their geographical distance. (b) CDFs of i-followers for the top 5 users (who are selected by maximum indegrees in $\mathcal{G}$).

cusing on the place where they are from and may expanse to other places. Therefore, in determining the influential Twitter users for a query location Q, Twitter users from other places should also be considered.

2.4 Measuring Spatial Influence In $\mathcal{G}$

2.4.1 Observation and Motivation

The objective of measuring the spatial influence of Twitter users is to find, given a query location Q, which Twitter users have great influence on it. An intuitive solution to this problem is to append a post-processing location filter step after finding influential people in general. For example, one can use the indegrees of vertices in $\mathcal{G}$ or apply the original PageRank schema (i.e., without giving geographical considerations) to yield a ranking order for Twitter users regarding their influence, and then select the ones who fall within Q and identify them as influential Twitter users on Q.

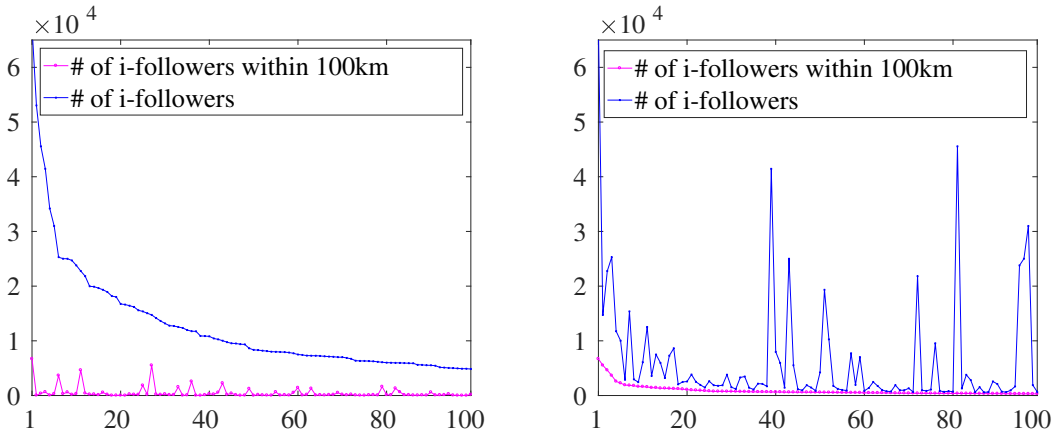


Figure 2.4: (a) Top 100 Twitter users in Boston sorted by the number of i-followers. (b) Top 100 Twitter users in Boston sorted by the number of i-followers within 100 km.

Such strategy, however, neglects a few important observations.

Regarding Local Influential Twitter Users: First, by utilizing a location filter, the above strategy assumes that, for a Twitter user in Q, his general influence equals to his local influence. This, however, is not necessarily true. For example, as shown in Figure 2.4a, a Twitter user from location Q who has many i-followers is not guaranteed to have many local people interacting with them, and *vice versa* in Figure 2.4b. A similar observation is also reported in [7] but on a scale of country-level.

Second, as shown in Figure 2.3a, for local influential Twitter users, their interaction followers are more aggregated around the local place, and the number of their followers decreases as distance increases, which implies that their influence is more revealed at local places and decays over longer distance. This motivates us to gives different geographical considerations to edges with different distances using a distance-decay function.

Regarding Generalized Influential Twitter Users: A user from a place (or even

without specifying his home location) other than the query location Q might have a considerable amount of i-followers from Q and thereby have a chance to exhibit non-negligible and even noticeable influence on Q. For example, even though the Twitter user "@Youtube" is from thousands of miles away from Boston, the number of his i-followers from Boston is larger than any other Twitter users that are in Boston. Another example is "@Patriots" who is a Boston-based Twitter user account, but we found in our dataset, thousands of people from Bristol, CT retweeting, replying and mentioning this user even though those two cities are almost 200km away. Applying a location filter to only keep the Twitter users who are within a limited geographical range to the query location center l_q is likely going to miss such users, and therefore is not suitable for determining the generalized influential Twitter users. This inspires us to alternatively measure a Twitter user's spatial influence on Q based on the spatial locality of his i-followers with respect to the query location center l_q and thereby capture Twitter users of great influence on Q without requiring them to have to be from Q.

In the following subsections, we first give a brief description of the PageRank algorithm. Next, we present our 2 instances of PageRank that address the above two observations, respectively. At last, a hybrid method is proposed trying to combine the 2 instances of PageRank together using distance-decay functions, along with a location query specific teleportation vector, which determines the initial influence values assigned to each vertex.

2.4.2 PageRank Overview

The mechanism behind PageRank can be briefly explained by an intuitive random surfer model on a given graph where this surfer visits a vertex with a certain probability and follows an outbound edge at random to visit next vertex. The influence of each vertex is then coded in the probability for the random surfer reaching that vertex, calculated as the sum of probabilities of the surfer following all possible edges towards to that vertex.

Additionally, PageRank defines a damping factor h which controls the probability that the random surfer, before starting visiting next vertex, chooses to follow an edge in the given graph to reach the next vertex or simply teleport to one which is not connected by edge with the previous vertex. This damping factor is used to avoid the random surfer being trapped in some disconnected components (if exist) in the directed graph and guarantees the convergence of PageRank. In summary, suppose we have a directed graph $G = (V, E)$ in which V is the vertex set and E is the directed edge set, the PageRank procedure can be iteratively defined as follows:

$$\mathbb{R}^{t+1} = (1-h) * \mathbf{\Pi} + h * \mathbb{R}^t \times \mathbb{M} \tag{2.2}$$

where $\mathbb{R}^t = \begin{bmatrix} r_1^t & r_2^t & \dots & r_N^t \end{bmatrix}$ is the ranking result after iterating t times, N is the number of vertices $N = |V|$, and each element r_i^t represents the PageRank score of the vertex v_i; h is the damping factor ranging from 0 to 1; $\mathbf{\Pi} = \begin{bmatrix} \pi_1 & \pi_2 & \dots & \pi_N \end{bmatrix}$ is a teleportation vector in which each element π_i denotes the probability that the surfer teleports to the vertex v_i from any other vertices; and $\mathbb{M}$ is the transition probability matrix which is

a $N \times N$ matrix with each element m_{ij} specifying the probability that the surfer transits to vertex v_j from vertex v_i by following an existing directed edge in the graph. In the typical PageRank algorithm, the teleportation probability to each vertex is identical by setting $\mathbf{\Pi} = \begin{bmatrix} \frac{1}{N} & \frac{1}{N} & \dots & \frac{1}{N} \end{bmatrix}$, and the transition probability m_{ij} is 0 if vertex v_i doesn't have a outbound edge to vertex v_j, and $m_{ij} = \frac{1}{|OUT_i|}$ if such an edge exists, where OUT_i denotes the set of vertices to which v_i has an outbound edge. For simplicity, we use the lowercase script out_j (or in_j) to denote the cardinality of the set OUT_j (or IN_j which denotes the set of vertices from whom vertex v_j has an inbound edge).

Transition Probability in Weighted Graph Generally, given a weighted graph, for example, $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ where $w_{ij} \in \mathcal{W}$ denotes the weight of an edge $e_{ij} \in \mathcal{E}$, the transition probability from vertex v_i to vertex v_j can be calculated defined as [121]:

$$m_{ij} = \frac{w_{ij}}{\sum\limits_{v_k \in OUT_i} w_{ik}} \tag{2.3}$$

2.4.3 Edge-Locality PageRank

Figure 3a shows that, as the representatives of local influential Twitter users, the local news agencies have more followers aggregated around the local place but less and less as geographical distance increases, indicating that their influence might decay over distance. This, therefore, inspires us that in determining local influential Twitter users, one Twitter user transfers more influence to another if they have a shorter geographical distance. We therefore propose to use a distance-decay function [21] to assign edges weights as follows and hence give more geographical considerations to edges who have

shorter distances, e.g,, those who are spatially local.

$$f^{EL} := \frac{1}{(d_{ij}+1)^{\kappa}} * \delta(e_{ij}) \tag{2.4}$$

where $\delta(e_{ij})$ is a binary checking function that outputs 1 if both of the two vertices v_i and v_j have geographical locations in $\mathcal{G}$, otherwise outputs 0. d_{ij} is the distance between vertex v_i and v_j when $\delta(e_{ij}) = 1$, otherwise, set to 0. Adding 1 to distance is to avoid zero-divisions. The parameter κ is the scale factor of distance decay [21] and determines the degree at which the power-law curve declines. In general, a larger κ yields a steeper curve and a more significant effect on distance decay.

With the above weight function f^{EL}, we calculate a Edge-Locality Transition Matrix $\mathbf{M^{EL}}$ using Equation 2.3. Along with the identical transportation vector Π in Equation 2.2, we now define Edge-Locality PageRank (**ELPR**) as follows:

$$\mathbb{R}^{t+1}_{ELPR} = (1-h) * \mathbf{\Pi} + h * \mathbb{R}^{t}_{ELPR} \times \mathbf{M^{EL}} \tag{2.5}$$

The ranking result yielded in *ELPR*, however, is not specific to the query location Q and thereby needs a location filtering post-process to find the Twitter users who are from Q.

Location Filtering Given a ranking list of Twitter users and a query location Q : (l_q, ϵ), an location filtering step is to output a Q-specific ranking list in which only the Twitter users at a distance of $\leq \epsilon$ to l_q are kept and their relative orders in the original ranking list are also reserved.

2.4.4 Source-Vertex-Locality PageRank

The preferences for spatially-local edges in the Edge-Locality PageRank (**ELPR**) proposed in previous section might miss some Twitter users who have been *retweeted*, *replied* or *mentioned* by a considerable amount of people from the query location Q, even though they are not from Q. To remedy this, in this section, we propose Source-Vertex-Locality PageRank (**SVLPR**) which addresses if a vertex is spatially local to the query location l_q, defined follows:

Definition 1. *A vertex $v_i \in \mathcal{V}$ is spatially local to a query location l_q if their distance(l_i, l_q) is within a threshold ϵ. Vertices that don't have a location l_i are not spatially local to l_q.*

We then propose the following source-specific weight function $f^{SVL} : e_{ij} \rightarrow w_{ij}$ in $\mathcal{G}$:

$$f^{SVL} := \delta'(v_i, l_q, \epsilon) \tag{2.6}$$

in which, $\delta'(v_i, l_q, \epsilon)$ is a binary spatial locality checking function that outputs 1 if the v_i is spatially local to l_q, otherwise outputs 0. In other words, Equation 2.6 essentially removes all edges $e_{i,j}$ where v_i is not in range ϵ of the query location l_q.

A Source-Vertex-Locality Transition Matrix $\mathbf{M^{SVL}}$ is then calculated using this weight function and Equation 2.3. Since $\mathbf{M^{SVL}}$ is already l_q-specific, we adopt identical transportation vector Π and define Source-Vertex-Locality PageRank (**SVLPR**) as

follows:

$$\mathbb{R}^{t+1}_{SVLPR} = (1-h) * \mathbf{\Pi} + h * \mathbb{R}^{t}_{SVLPR} \times \mathbf{M^{SVL}} \tag{2.7}$$

2.4.5 Geographical PageRank

Edge-Locality PageRank (*ELPR*) and Source-Vertex-Locality PageRank (*SVLPR*) find influential Twitter users on location Q by addressing two different geographical considerations of $\mathcal{G}$. The former emphasizes on the edges that are formed within a shorter spatial distance, while the latter focuses on the edges whose source vertices fall spatially-locally to the query location center l_q. In this section, we combine these two factors in a Geographical PageRank (**GPR**) algorithm. Specifically, concatenating these two factors in **GPR** is completed by the operation of multiplication via the weight function f^{GEO} defined as:

$$f^{GEO} := \begin{cases} \dfrac{1}{(d_{ij}+1)^{\kappa}} * \delta(v_i, l_q, \epsilon), & \text{both } v_i, v_j \text{ have locations,} \\ \dfrac{|IN_j^{l_i,\epsilon}|}{|IN_j|} * \delta(v_i, l_q, \epsilon), & \text{only } v_i \text{ has a location} \\ \dfrac{1}{(d_{max}+1)^{2\kappa}}, & \text{otherwise.} \end{cases} \tag{2.8}$$

where d_{ij} is the distance between vertex v_i and v_j, d_{max} is the maximum value of all d_{ij} and used to punish edges both of whose two vertices don't have available locations. And $\delta(v_i, l_q, \epsilon)$ has the same definition with the one in Equation 2.6.

Recall that IN_j is the set of vertices from whom vertex v_j has an inbound edge, i.e., the set of interaction followers of user v_j. We additionally define $IN_j^{l_i,\epsilon}$, a subset of

IN_j in which each vertex is within ϵ distance to the location l_i. When a vertex v_j doesn't have a location label, our intuition of using $\frac{|IN_i^{l_i,\epsilon}|}{|IN_i|}$ as its spatial influence to l_i is out of consideration for its likeliness of falling nearby to l_i by treating its interaction followers' locations as its potential location distribution.

From Equations 2.8 and 2.3, we then calculate a Geographical Transition Matrix $\mathbf{M^{GEO}}$. To give more preference to Twitter users who have shorter distance to the query location center l_q, we propose the following Q-Specific Teleportation Vector Π^Q to complement the Geographical Transition Matrix $\mathbf{M^{GEO}}$.

Query Location Q-Specific Teleportation Vector For a query location Q, we first define a vertex v_i's spatial relevance to Q as follows:

$$rel(l_q, v_i) = \begin{cases} \dfrac{1}{\Big(distance(l_i, l_q) + 1\Big)^{\kappa}}, & v_i \text{ has geo-coordinates } l_i \\ \dfrac{|IN_i^{l_q,\epsilon}|}{|IN_i|}, & \text{otherwise} \end{cases} \tag{2.9}$$

where $distance(l_i, l_q)$ calculates the geographical distance between the query location l_q and the location l_i of vertex v_i. The definition of $\frac{|IN_i^{l_q,\epsilon}|}{|IN_i|}$ is similar to the one defined in Equation 2.8 but now measures the likeness of an unknown-location vertex v_i falling in Q by treating v_i's interaction followers' locations as its potential location distribution. Now, we normalize the spatial relevance to get a vertex's teleportation probability $\pi_i = \frac{rel(l_q,v_i)}{\sum_{v_k \in \mathcal{V}} rel(l_q,v_k)}$ and use $\mathbf{\Pi^Q}$ to denote such a teleportation vector.

Combining the Geographical Transition Matrix $\mathbf{M^{GEO}}$ and the query location Q-Specific Teleportatoin Vector $\mathbf{\Pi}^Q$,we define the Geographical PageRank (**GPR**) as fol-

lows,

$$\mathbb{R}_{GPR}^{t+1} = (1-h) * \mathbf{\Pi}^{Q} + h * \mathbb{R}_{GPR}^{t} \times \mathbf{M}^{\mathbf{GEO}} \tag{2.10}$$

Like *SLVPR*, we don't apply a location filtering post-process on *GPR*, which will miss the generalized influential Twitter users on the query location Q. This distinguishes from *ELPR*, which is not query location specific and thereby runs only once for different Qs, although a location filter is needed.

2.5 Empirical Evaluation

In this section, we first describe the experimental settings including the related baselines approaches, the evaluation methods and default parameters settings. Next, we report the results of measuring spatial influence of Twitter users by different methods regarding 3 cities in the USA. Afterwards, we choose the city of Boston, MA to report the comparing results. Furthermore, we study the effects of the interaction's types, along with the effects of applying a geotagging procedure to estimate locations for unknown-location Twitter users, followed by a study on the sensitivity of the distance-decay factor κ in *ELPR* and *GPR*. Finally, we discuss the potential applications of using local influential Twitter users as news seeders with respect to local news (event) detection.

2.5.1 Experimental Settings

2.5.1.1 Baseline Approaches

Because the difference in types of influential Twitter users that *ELPR* and *SVLPR* are trying to find – the former finds *local influential Twitter users* who are not only having great influence on a location but also from there while the latter find *generalized influential Twitter users* and doesn't have a requirement regarding where they are from, we put them in two different control groups and list their related baseline approaches separately as follows. In addition, we also present the results of using the hybrid method *GPR* to investigate its effects of combing the two types of spatial locality defined in Section 2.4.3 and Section 2.4.4, respectively. Baseline approaches to *ELPR* (Edge Locality Group):

- *InD*: measures the influence by a user's In-Degree in $\mathcal{G}$, i.e., the number of i-followers a Twitter user has.
- *LocInD*: measures the influence of a user by the number of its i-followers who are within ϵ distance to this user.
- *PR*: i.e., PageRank, measures the influence by a user's score by running PageRank on $\mathcal{G}$.

Baseline approache to *SVLPR* (Source Vertex Locality Group):

- $iFol-l_q$: measures the influence of a user by the number of its i-followers who are within ϵ distance to l_q.

Since *InD*, *LocInD*, *PR* and *ELPR* are not Q-specific, a location filter is applied to only keep the Twitter users within ϵ distance to l_q.

2.5.1.2 Evaluation Methods

Choosing the city of Boston, MA, we study two aspects of the ranking algorithms: correlation and effectiveness.

1) Correlation. The correlation is measured by a modification of Kendall's τ [56] used in Kwak et al. [59]. This modification overcomes the the limit in the original Kendall's τ that rankings in comparison must have the same element and allows for comparing only top k elements in each rankings. The correlation ranges from 0 to 1, and a larger value indicates a stronger agreement. In this experiment, we only compare the top 100 in each algorithm's ranking result.

2) Effectiveness. It is very difficult to evaluate the effectiveness of rankings in lack of a ground-truth. To approach the effectiveness evaluation, for the methods in the group of "Edge Locality", we utilize a set of manually-collected local influential Twitter users in Boston, MA and compute the **average ranking order** in each of the methods; for the methods in the group of "Source Vertex Locality", we calculate **the number of verified Twitter accounts** in the top 100 influentials identified in each of the methods.

Average Ranking Order: We first manually collect 20 locally influential Twitter users accounts from 4 different categories in Boston metropolitan area and list them as follows:

News Agencies – "@wcvb", "@bostondotcom", "@cbsboston", "@7news", "@boston-

hearld";

Sports Team – "@redsox", "@celtics", "@nhlbruins", "@thebostonpride", "@boston-cannons";

Government – "@marty_walsh", "@cityofboston", "@bostonpolice", "@boston-fire",

"@masddot";

University – "@bu_tweets", "@harvard", "@mit", "@berkleecollege", "@north-eastern";

As describe in the following, the selection of the representative local influential Twitter users is completed by using an external authority, i.e., Google Search Engine, and such knowledge is not known a priori. More importantly, the experimental evaluation is not only to identify these local influential users but instead to compare the average ranking order of them.

Collecting Twitter users in the first 2 categories is completed by first typing the keywords in Google "Boston local news", and "Boston Sports team" to find top related websites and then locating their official Twitter accounts on the webpages. We didn't choose the news agency of "Fox 25 Boston" because it changes its Twitter account from "@fox25news" to "@boston25" in April 2017. The Twitter accounts in the category of *Government* are the official accounts of Boston Mayor, Boston Government, Boston Police Department, Boston Fire Department and Massachusetts Department of Transportation, respectively. And the Twitter accounts in the category of *University* are the official accounts of Boston University, Harvard University, Massachusetts Institute of Technology, Berklee College of Music and Northeastern University, respectively.

The order of Twitter users in a ranking starts from 0. The smaller order a Twitter user has, the more influential he is in that ranking. The average ranking order of a set of influential Twitter users in a ranking is the average of the orders of each influential Twitter in that ranking. ***Therefore, a smaller average ranking order indicates a better ranking algorithm***.

Number of Verified Accounts: In Twitter, verified accounts are the ones that have been examined to be authentic by Twitter itself and considered as high-quality Twitter users. The status of verification can be found in the Twitter user's profile information. We therefore propose to check the quality of a ranking algorithm by counting how many verified Twitter accounts in its top 100 elements. ***The more verified accounts that a ranking algorithm has in its top*** 100***, the higher is its quality.***

Note that in the evaluation, we also report the performance of the "Source Vertex Locality" methods regarding the metric of Average Ranking Order; and *vice versa.*, the performance of the "Edge Locality" methods regarding the metric of Number of Verified Accounts is also given.

2.5.1.3 Default Parameter Setting

The default parameters used in our methods and related baseline approaches are set as follows.

- l_q: the query location centers of "Boston, MA", "Bristol, CT" and "Seattle, WA" are set to 42.3584/-71.0598, 41.6812/-72.9407 and 47.6062/-122.332, respectively, using GeoNames database.

- ϵ: the radius ϵ in the query location Q (also the spatial locality threshold in Definition 1), is set to 100km, which is large enough for majority of the cities.

- h: the damping factor in PageRank is set to 0.85 for the algorithm *PR*, *ELPR*, *SVLPR* and *GPR*.

- κ: the distance-decay factor in *ELPR* and *GPR* are set to 4 in default. The sensitivity of κ is reported in Section 2.5.6.

- PageRank Iterations: 100 for *PR*, *ELPR*, *SVLPR* and *GPR*.

- Distance Unit: the distance is in the unit of ϵ, i.e., 100km. .

City	Edge Locality			
	InD	*LocInD*	*PR*	*ELPR*
Boston, MA	Patriots	Patriots	Patriots	Patriots
	CrazyFightz	OnlyInBOS	OITNB	BostonGlobe
	DrJillStein	BostonGlobe	JohnCena	OnlyInBOS
	Diaryforcrush	RedSox	BostonGlobe	RedSox
	TWICHISTE	stoolpresidente	RedSox	NHLBruins
Bristol, CT	SportsCenter	SportsCenter	SportsCenter	SportsCenter
	espn	espn	espn	espn
	ESPNNFL	SmackHighCT	ESPNNFL	SmackHighCT
	ESPNStatsInfo	ESPNNFL	ivoryella	MikeAndMike
	darrenrovell	ESPNStatsInfo	darrenrovell	ESPNStatsInfo
Seattle, WA	amazon	Seahawks	amazon	Seahawks
	OriginalFunko	Mariners	Starbucks	Mariners
	Starbucks	KING5Seattle	Seahawks	SoundersFC
	Seahawks	seattletimes	BillGates	seattletimes
	XSTROLOGY	SoundersFC	Microsoft	KING5Seattle

Table 2.1: The top 5 influential Twitter users identified for 3 different cities using *InD*, *LocInd*, *PR* and *ELPR*.

City	Source Vertex Locality		Hybrid
	$iFol-l_q$	$SVLPR$	GPR
Boston, MA	YouTube	YouTube	Patriots
	realDonaldTrump	realDonaldTrump	Youtube
	Patriots	Patriots	BostonGlobe
	GIRLposts	BostonGlobe	OnlyInBOS
	HillaryClinton	OnlyInBOS	RedSox
Bristol, CT	YouTube	YouTube	Youtube
	realDonaldTrump	realDonaldTrump	SportsCenter
	GIRLposts	GIRLposts	WSHHFANS
	SportsCenter	SportsCenter	realDonaldTrump
	SincerelyTumblr	CauseWereGuys	Patriots
Seattle, WA	YouTube	YouTube	Seahawks
	Seahawks	Seahawks	YouTube
	realDonaldTrump	realDonaldTrump	Mariners
	HillaryClinton	Mariners	seattletimes
	GIRLposts	DangeRussWilson	SoundersFC

Table 2.2: The top 5 influential Twitter users identified for 3 different cities using $iFol-l_q$, $SVPR$ and GPR.

2.5.2 Top 5 Influential Twitter Users for 3 U.S. Cities

In this section, we analyze and compare the top 5 Twitter users identified by our methods and the ones by related baseline approaches with regards to 3 cities "Boston, MA", "Bristol, CT" and "Seattle, WA". The results are listed in Table 2.1 and Table 2.2, in which the symbol "@" at the start of each Twitter username is omitted for compactness.

We notice that quite a few of the top 5 influential Twitter users listed in Table 2.1 and Table 2.2 are related to commercial accounts. This doesn't come as a surprise in the sense that such users usually have more interactions from other Twitter users due to their population and thus would rank at top positions. More examples of influential Twitter users from various walks of life (news media, reporters, sports team, sports player, politicians, musicians etc.) with respect to Boston can be found in the Appendix Table 8.1

and Table 8.2.

By listing only the top 5 influential Twitter users, Table 2.1 and Table 2.2 are able to show that in general, taking into geographical proximity into consideration, our proposed methods yield better results than the baseline approaches. Such a difference becomes more significant when the ranking orders (i.e., as the one listed in the table) are taken into account. We were surprised to observe such differences even for only the top 5 users. In the following, we describe the details of such a difference observed in different methods.

InD* vs. *LocInD: In general, *InD* might return noise Twitter users. For example, we do not consider the Twitter users "@Diaryforcrush" and "@TWICHISTE" for the city "Boston, MA" and the Twitter user "@XSTROLOGY" for the city of "Seattle, WA" identified by *InD* are of great influence on their cities because they have very few people from their cities to interact with them. Take "@TWICHISTE" for example, out of the $38,187$ i-followers he has, only 10 are within $100km$ of the center of Boston, MA. In contrast, the 5-th local influential Twitter user "@stoolpresidente" identified by *LocInD* only has $11,754$ i-followers, but $2,356$ of them are within $100km$ to the center of Boston, MA. Although "@Diaryforcrush" and "@XSTROLOGY" get the locations from their geotagged tweets, diffuse of such type of geographical information is not going to totally eliminating noisy users because of the existence of users like "@TWICHISTE" who indeed has a profile-location as "Boston, MA", and might also miss some important Twitter users like "@Patriots" and "@Mariners", neither of them giving valid profile-locations.

In contrast, by finding Twitter users who have the most interaction followers from the local area, *LocInD* gives high quality results. For example, most of the Twitter accounts identified by *LocInD* are officially accounts of either sports teams, or local news

agencies or reporters in each of the three cities, with an exception of "@SmackHighCT" in the city of "Bristol, CT", which is a branch account of a social platform SmackHigh. This account usually posts hilarious contents on high school lifestyle and receives lots of "retweets" from almost one thousand of people in "Bristol, CT".

PR* vs. *ELPR: Both *PR* and *ELPR* improve on their indegree counterparts *InD* and *LocInD* by not just considering how many i-followers (or local i-followers) a user has but also the influence of these i-followers. For example, in comparison with *InD*, the Twitter users in *PR* are all official and verified accounts. Similarly, "@NHLBruines" ranked in the top-5 in *ELPR* but not in *LocInD* because all the top-4 users in *ELPR* (or *LocInD*) are i-followers of "@NHLBruines" while only 2 of them are i-followers of "@stoolpresidente" even though "@NHLBruines' has less i-followers from Boston than "@stoolpresidente".

Taking the spatial locality of edges into consideration, *ELPR* generally outputs a different set of top 5 influentials in comparison with *PR* across the 3 cities because it focuses more on the interactions happened geographically within a city-level. Take the city of Boston for example, the top 5 influentials in *ELPR* have an average of 4204.2 people from Boston actively interacting with them, while the ones in in *PR* have only 2900.2. The numbers for the cities of Bristol and Seattle are 1286.8, 1423.8 and 1396.0 and 2371.6, respectively. This means that *ELPR* more effectively finds Twitter users that are locally influential.

iFol* $-$ l_q vs. *SVLPR: Comparing to previous algorithms, $iFol - l_q$ and *SVLPR* find Twitter users who are influential on a place but not necessarily from there. For example, either the profile-locations of "@YouTube", "@realDonaldTrump" or "@HillaryClinton" is specfified as the 3 cities. Another Twitter user "@GIRLposts" doesn't has a profile-

location. But this doesn't mean they don't have influence or negligible influence on the 3 cities. For example, for each of the 3 cities, "@YouTube" has the most number of i-followers from that city than any other accounts, even the ones who are at the city.

In comparison with $iFol - l_q$, the portion of the Twitter users who are from the query city identified by *SVLPR* slightly increases due to its additional consideration of link structures.

Furthermore, in these two methods, several Twitter users are found influential across all the three cities such as "@YouTube" and "@realDonaldTrump", indicating that the influence of these Twitter users are not limited to a local place and goes beyond their profile-locations. This corresponds to the distance distributions of such Twitter users to their i-followers as shown in Figure 2.3.

GPR: Taking both the spatial locality of edges and source vertices into consideration, *GPR* outputs a combination of the top influentials in *ELPR* and in *SVLPR*. The most interesting finding is that "@Patriots", the official Twitter account of a sport team based in Boston, ranked 5th in *GPR* regarding its influence on the city of Bristol, CT. This is because on one side, "@Patriots" has thousands of people from Bristol, CT to interact with it and on the other side, Boston,MA is at a moderate distance of 170km from Bristol, CT.

2.5.3 Correlation and Effectiveness

2.5.3.1 Correlation

The correlations between the algorithms are listed in Table 2.3, in which the highest correlation in each row is in bold font. It is clearly that indegree methods are more related to their PageRank counterparts, for example, *InD* vs. *PR*, *LocInD* vs. *ELPR* and $iFol-l_q$ vs. *SVLPR*. In contrast, our proposed methods have lower correlation with the existing metrics *InD* and *PR*, indicating they generate different ranking results to them. This implies identifying spatial influential Twitter users is not a simply procedure of first determining general influence in interaction graph $\mathcal{G}$ by *InD* and *PR* and then applying a location filter post-processing.

In addition, the methods *InD*, *LocInD*, *PR* and *ELPR* have lower correlations to the methods $iFol-l_q$ and *SVLPR* because the former group of methods require that a Twitter user who is influential on a location Q is also from that location, while the latter group of methods don't have such a requirement.

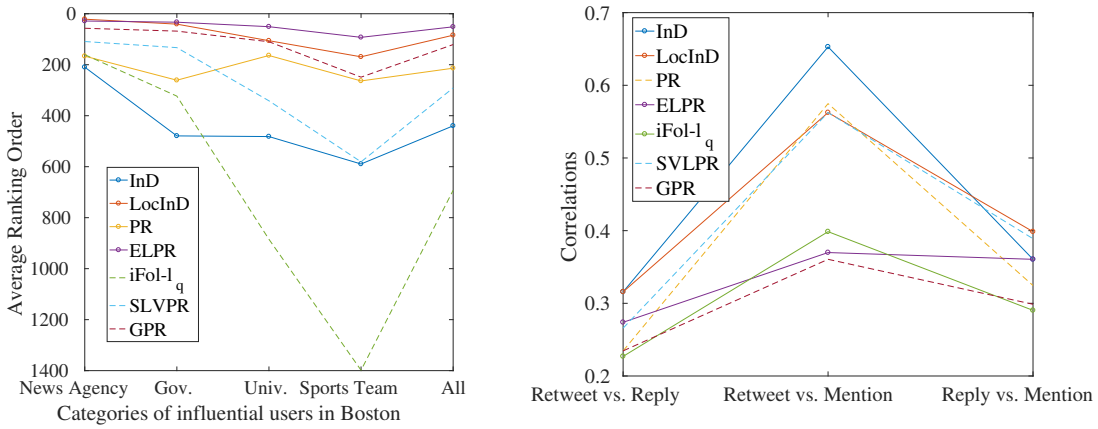
In default, our hybrid method *GPR* is slightly more correlated with *SVLPR* than *ELPR*, indicating that it emphasizes more on the spatial locality of source vertices than the spatial locality of edges and might have more Twitter users who are not from the query location Q in its ranking results as shown in Table 2.2.

We also compared with *GPR* after applying the location filter with the methods in *ELPR* group, and listed the results in the column with the header $GPR-l_q$. It shows that the hybrid method *GPR*, after removing the users who are not local the query location,

strongly correlates with *ELPR*.

	Edge Locality				Source Vertex Locality		Hybrid	
Corr.	*InD*	*LocInD*	*PR*	*ELPR*	$iFol-l_q$	*SVLPR*	*GPR*	$GPR-l_q$
InD	1.0	0.35	**0.60**	0.35	0.17	0.17	0.28	0.31
LocInD	0.35	1.0	0.36	**0.60**	0.30	0.38	0.30	0.54
PR	**0.60**	0.36	1.0	0.40	0.20	0.18	0.29	0.36
ELPR	0.35	**0.60**	0.40	1.0	0.30	0.19	0.50	0.54
$iFol-l_q$	0.17	0.30	0.20	0.30	1.0	**0.60**	0.53	0.17
SVLPR	0.17	0.38	0.18	0.19	**0.60**	1.0	0.56	0.15
GPR	0.28	0.30	0.29	0.50	0.53	**0.56**	1.0	0.23

Table 2.3: Correlation matrix between different algorithms.



(a) (b)

Figure 2.5: (a) The average ranking orders of 4 different categories of the local influentials in Boston, MA. (b) The correlations between different types of interactions.

2.5.3.2 Effectiveness

Average Ranking Order: Figure 2.5a shows the average ranking orders of the 4 categories of manually-collected locall influential Twitter users in Boston by different algorithms. Clearly, our method *ELPR* outperforms its baseline approaches *InD*, *LocInD* and *PR*. In addition, *LocInD* outperforms both *InD* and *PR*, justifying the benefits brought

by considering the spatial locality of edges in graph $\mathcal{G}$ in determining the spatial influence of Twitter users. Moreover, our method *SVLPR* that is aware of the spatial locality of source vertices to query location l_q, also achieves better performance than its baseline approach $iFol - l_q$ by additionally taking into account of link structures. At last, our hybrid method *GPR* considering both of the two types of spatial locality in *ELPR* and *SVLPR* has a moderate performance because it introduces popular users like "@YouTube" who are not in Boston, MA.

Number of Verified Accounts: Table 2.4 list how many verified accounts are there in the top 100 Twitter users identified by different methods. The results show that in the group of "Source Vertex Locality", our proposed method *SVLPR* is slightly better than its baseline approach $iFol - l_q$ because its additional awareness of link structures; and in the group of "Edge Locality", our proposed method *ELPR* clearly outperforms other related methods because in reality, most of the local influential accounts are official accounts of entities like organizations etc and such accounts usually are verified by Twitter. Our hybrid method, *GPR*, again achieves a moderate performance. This is because, comparing the *ELPR*, it also retrieves Twitter users that are pouplar among people but not necessarily are verified like "@WSHHFANS" because such Twitter users may not represent any organization entities in the real world.

Edge Locality				Source Vertex Locality		Hybrid
InD	*LocInD*	*PR*	*ELPR*	$iFol - l_q$	*SVLPR*	*GPR*
46	76	63	81	55	59	60

Table 2.4: Number of verified Twitter users in the top-100.

2.5.4 Different Types of Interactions

as shown in Figure 2.1b, the 3 types of interactions *retweet*, *reply* and *mention* contribute differently to building the edges in the interaction graph $\mathcal{G}$. To investigate how much difference in the influential Twitter users identified by different types of interactions, we run the algorithms on the graphs constructed from only using *retweet*, *reply* and *mention*, respectively and calculate the correlations for *retweet* vs. *reply*, *retweet* vs. *mention* and *reply* vs. *mention*. The results are plotted in Figure 2.5b, which shows that *retweet* vs. *mention* generally has stronger correlations than *retweet* vs. *mention* and *reply* vs. *mention*. This corresponds to Figure 2.1b where *retweet* and *mention* have more edges in common than the other two.

2.5.5 Effects of Geotagging Twitter users

In this section, we study the effects of applying a geotagging procedure to estimate locations for unknown-location Twitter users. We use relative changes (+/-) in Average Ranking Order and relative changes (+/-) in the Number of Verified Accounts to evaluate an algorithm's changes before and after geotgagging. The relative changes of Average Ranking Order is calculated with respect to all the 20 manually-collected Twitter users in Boston in Section 2.5.1.2.

We estimate location for unknown-location Twitter users by utilizing a Twitter user geotagging procedure [24], which is reported to have the state-of-the-art city-level accuracy. In essence, [24] assigns a location estimation to a Twitter user by using his reciprocal friends' locations as a set of points in Equation 2.1 to calculate a median point. After the

geotagging, we have 74,846,116 (50.6%) Twitter users assigned with geographical locations, and 1,084,772,048 (72.1%) edges whose two vertices both have geographical locations.

We then again run the different algorithms using the new set of location labels of vertices for $\mathcal{G}$, and list their results in Table 2.5. For the changes in Average Ranking Order, in the control group of "Edge Locality", methods *InD*, *LocInD* and *PR* get more affected by the geotagging procedure while *ELPR* receives less effects with the Average Ranking Order by only increasing 5.4. The method $iFol - l_q$ is much more susceptible to geotagging than *SVLPR* because $iFol - l_q$ only considers the the location of source vertices and would exhibit larger difference when more Twitter users are geotagged as in Q.

For the changes in the Number of Verified Accounts, all methods yield slight changes before and after geotagging, indicating geotagging unknown-location Twitter users has less effect on the verified official accounts. This is because in most cases such verified official accounts are likely to provide a profile-location, which lets us have their geographical location at hand before geotagging.

	Edge Locality				SV Locality		Hybrid
	InD	*LocInD*	*PR*	*ELPR*	$iFol - l_q$	*SVLPR*	*GPR*
Avg. Order	477.8	97.6	232.8	56.7	764.9	301.9	107.7
Avg. Order +/-	+38.1	+13.35	+19.15	+5.4	+72.4	+10.75	-13.25
Veri. Accts.	48	75	62	80	52	56	58
Veri. Accts. +/-	+2	+1	+1	-1	-3	-3	-2

Table 2.5: Effects of geotagging on different algorithms.

2.5.6 Sensitivity of Distance-Decay Parameter κ

To investigate the sensitivity of κ in *ELPR*, we compare its correlations to *InD* under different values of κ. Similarly, for the sensitivity of κ in *GPR*, we compare its correlation to *ELPR* and *SVLPR* respectively. The results are listed in Table 2.6.

Table 2.6 shows that until $\kappa = 2^3$, *ELPR* is becoming more similar to *InD* as the value of κ continues to increase to generate a more significant effect of distance-decay in $\mathbf{M}^{EL}$. This indicates that a larger κ makes *ELPR* more prefer Twitter users having shorter distance to the query location center l_q. The correlation drops at $\kappa = 2^3$ because *LocInD* relies on a location filter and doesn't geographically distinguish the Twitter users within $\epsilon = 100km$ to the query location l_q, while *ELPR* continues preferring to rank higher for those who have even shorter distance to l_q.

Table 2.6 also shows that *GPR* has more similarities to *SVLPR* across different κ than *ELPR*. In the meantime, as κ increases, *GPR* gives more weights on edges having shorter distances and thereby its correlation with *ELPR* increases. Such trade-off stabilizes after 2^2.

κ	2^{-3}	2^{-2}	2^{-1}	2^0	2^1	2^2	2^3
ELPR vs. *LocInD*	0.01	0.01	0.05	0.33	0.54	0.60	0.57
GPR vs. *ELPR*	0.02	0.02	0.02	0.11	0.39	0.50	0.50
GPR vs. *SVLPR*	0.46	0.46	0.47	0.52	0.57	0.56	0.56

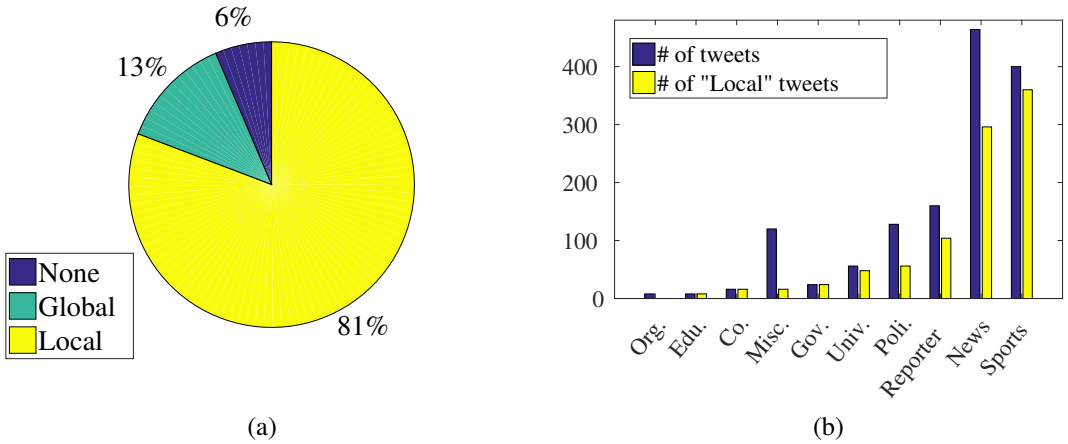
Table 2.6: Sensitivity of κ in R_{ELPR} and R_{GPR} regarding their correlation with R_{LocInD}, R_{ELPR} and R_{SVLPR}, respectively.

2.5.7 Application to News Detection

In this section, we explore the potential of local influential Twitter users in acting as news sources (e.g., news seeders [102, 32]) by examining how many of their tweets are about local news (events).

Using the dataset in Section 2.3, we collected 1,306 tweets posted by the top 70 Boston influential Twitter users identified by the ELPR method (which is given in Table 8.1 in Appendix 8 between Dec 01, 2016 and Dec 07, 2017, and manually categorize these tweets into "Local", "Global" and "None", indicating whether a tweet is about Boston's local news (events), or generally global news or neither of both. The mean and median number of tweets in each user are 18 and 6, respectively. The distribution is presented in Figure 2.6a, showing that 75% of the tweets are about news, and more importantly, 67.1% are considered local. This supports the viability of using local influential users as potential local news seeders.

However, not every local user tweets about a local location. For example, although "@HarvardBiz" is considered influential in Boston, his tweets are mostly reviews on business and technology etc, and are rarely about local news or events. Thus, to investigate which category of Twitter users (the category information of users is provided in Table 8.1 and Table 8.2 in *italic* font) are contributing "Local" tweets, for each category of users, we plot the number of their total tweets and the number of their "Local" tweets in Figure 2.6b, which shows that the users in the categories of *Reporters, News* and *Sports* are contributing most of the "Local" tweets and meanwhile maintain a high fraction of "Local" tweets in their own tweets. In addition, most of the tweets posted by users in



(a) (b)

Figure 2.6: (a) Ratio of "Local", "Global" and "None" tweets by top 70 Boston influential users in method ELPR. (b) Number of total tweets vs. number of "Local" tweets for different categories of users.

University, Government and *Education* are considered "Local", though they have fewer tweets. Therefore, these users might be considered as news seeders, and additional procedures such as classification or topic-sensitive ranking [119] might be exploited in the future to pick out such types of users to improve the quality of news seeders.

2.6 Conclusions

This chapter focused on finding spatial influential Twitter users on a query location Q based on the interactions sent out by the local people from Q. The experiments show that by making use of the spatial local edges, our proposed method Edge-Locality PageRank (ELPR) outperforms other related algorithms in finding local influential Twitter users. As local influential Twitter users don't include the ones who are from other places but still have great influence on Q, we furthermore present a method Source-Vertex-Locality PageRank (SVLPR) to find generalized influential Twitter users on the query location Q

without requiring them to be from location Q. A hybrid method Geographical PageRank (GPR) that takes into account both edge locality and source vertex locality to determine influential Twitter users is also presented. In addition, we also investigate the influence determined by using different types of interactions and also the effects of applying a geo-taggging procedure.

There are still many aspects of interactions to explore, such as the frequency and temporal properties [30]. The reciprocity of interactions is also another interesting factor. Also, it is an interesting topic to investigate the typical patterns of how user's influence evolves across regions. In addition, *SVLPR* or *GPR* can be modified to find local influential Twitter users by appending a location filter and therefore to compare with *InD* and *ELPR*. At last, as discussed in Section 2.5.7, topic-sensitive technologies like LDA might be explored further to identify local influential Twitter users that are in the topic of local news (events).

Chapter 3: Detecting Latest Local Events from Geotagged Tweet Streams

3.1 Introduction

With people posting what is happening outside in the real world, tweets in Twitter encapsulate invaluable information on real-world events as they break. For example, Twitter is nowadays becoming an increasingly important information channel to for people to learn about breaking news [58, 59]. Except for the basic facts including a text message and a publish timestamp, some tweets may also contain a geotag to indicate the posting location, and termed *geotagged tweets*. Geotagged tweets are particularly interesting in the sense that they provide the complement information about the place of interest, e.g., where the events occur. Such location information is crucial when profiling the occurrence of an event by filling in the three pieces of information: *where*, *when* and *what*. In this chapter, we aim to detect the latest local events from live geotagged tweet streams. A local event is defined as an unusual activity that appears at some specific time and place and also shows topical coherence. For instance, Figure 3.1 presents some exampling geotagged tweets about a soccer game held in the city of Seattle, WA. Timely discovering such local events has a wide range of applications. For example, people can acquire the latest information about such local activities in their living town., and thereby enhance their daily lives. It can also be helpful for commuting alarms by reporting real-time traffic

Figure 3.1: Examples of geotagged tweets about the soccer game of "Seattle Sounders" vs "D.C. United" at the stadium of "CenturyLink Field" at 7:30 PM, 2017-07-19. All the tweets were located at the stadium of "CenturyLink Field", i.e., the red grid cell in Figure 3.2a.

jams or accidents. In such cases, after learning what is happening, the commuters can actively make a decision to bypass the congested road segments or avoid the accident sites.

It is, however, challenging to detect local events from live geotagged tweet streams. First, detecting local events by capturing unusualness requires considering not only temporal historical patterns but also spatial circumstances. Some studies [84, 5, 62, 40] measure the burstiness, intensity of increment in the number of tweets at a place over a short time period, as signals of local events. But burstiness does not always imply the occurrence of a local event. For example, the burstiness of tweets at a shopping mall or a famous coffee bar in the morning is often expected and not unusual. Some work improves this measure to capture temporally routine patterns by gathering time-aware statistics [4]. However, without geographical consideration, occasional nation-wide events may also

accumulate a temporally unusual number of tweets at local places. For example, on presidential election nights, one may observe suddenly more tweets all over the places. Second, a local event, as it develops, may receive follow-up updates on its content and may also migrate geographically. For example, when a crime happens at a place, people expect to receive updates as investigation progresses. Another example is that a demonstration protest may follow a route moving from one place to another. Therefore, it is desirable to dynamically and timely monitor and track the development of an ongoing event, and report its latest updates.

In this chapter, we propose DeLLe to discover, track and describe local events from live geotagged tweets. The contribution of DeLLe lies in its four modules: *seeker*, *ranker*, *expander* and *summarizer*.

Seeker finds unusual locations which exhibit spatiotemporal unusualness with respect to the number of tweets and therefore potentially correspond to local events. For this purpose, seeker employs a novel prediction-based anomaly detection strategy. In particular, seeker first exploits convolutional LSTMs (ConvLSTM [103]) to predict the expected number of tweets in the future, which accounts for not only historical patterns but also neighboring locations. Next, seeker compares the predicted value with the actual number of tweets to determine the existence of unusualness. Unlike previous studies [58, 5] which claim anomalies only based on the local time series data of a location, we also consider the horizontal situation in other places simultaneously to mitigate the effects of global events.

Ranker suppresses the possibly noisy candidates of local events. In practice, not all spatiotemporal burstinesses necessarily correspond to an actual local event. We therefore

bring order to the candidates with a ranking procedure by considering temporal burstiness, spatial burstines and topical coherence, and thereby select the top ones likely to be corresponding to the occurrence of local events.

Expander tracks and updates the movement of an ongoing local event in both space and time using event-focus and content similarity. An event focus records its most important site of occurrence at certain time. While the content similarity between the tweets in two nearby locations is used as a measurement to check whether an ongoing local event moves to nearby locations or keeps bubbling up at the same place. In so doing, this module is dynamically monitoring the impact range of a local event.

Summarizer generates an abstract for a detected local event by selecting its most influential tweets. For human consumption, an event should be presented in a succinct description [58] but yet with up-to-date and key information. It is therefore important to choose representative tweets to summarize the detected local events. This module builds tweet authority graphs based on their textual similarities and subsequently runs random walk procedures to select the most influential tweets for events.

3.2 Related Work

There has been a plural of works on detecting local events using tweets in Twitter. Atefeh [8] and Abdelhaq [2] provide two excellent surveys. In general, existing methods focusing on geotagged tweets can be classified into two strategies: *model dimension extension* and *geographical space tessellation*. Model dimension extension treats location as additional variables to existing models. For example, some studies treat location as

latent variables in their generative topic model [131, 44, 140, 117]. Location distance between tweets can also be incorporated to measure similarities [134, 130, 112, 10] during clustering.

Geographical space tessellation divides space into small and disjoint cells for aggregating geotagged tweets. The motivation is that a local event usually has a limited spatial impact and would fall in the same or nearby cell(s). The grid is the simplest yet most commonly used way of tessellation [104, 3, 5, 2, 50, 113, 49, 141], although other structures have also been explored including hierarchical triangular meshes [58], pyramid structures [81, 80], Voronoi tessellations [66] and k-d tree [76].

After aggregating tweets to tessellation cells, a simple way for event detection is to examine whether the number of the aggregated tweets or the arriving rate exceeds a certain threshold [84, 4]. This, however, is easily plagued by tweet distribution heterogeneity both temporally and spatially. Thus, various anomaly detection methods have been explored. The core idea is to use previous history of data to build a baseline (or make a prediction) and then compare with the actual value to check for significant discrepancies [58, 5, 62]. For example, TwitInfo [83] uses the weighted average of historical tweet counts to compute the expected frequency of tweets. But sole historical data often neglect the effects exerted by nearby geographical regions. Krumm and Horvitz [58] therefore include features like tweet counts from adjacent regions in their anomaly detection method. Our method is different from the above methods in two senses: 1) our prediction model captures both spatial dependencies and temporal patterns [116]; 2) when claiming an anomaly, we account for not only the history of a location itself but also the situation at other places in the meantime to mitigate the effect of unexpected global events.

We found that the most related work to our task are EvenTweet [5], Eyewitness [58], GeoBurst [134] and TrioVecEvent [131]. EvenTweet detect events by identifying and clustering temporal bursty keywords. However, using words instead of tweets as clustering elements, this method may group semantically irrelevant words together and in the meantime not sit well with event summarization. Eyewitness [58] discretizes space and time and finds tweet volume spikes as potential local events by comparing the predicted value with the observed value. However, it needs to perform an exhaustive sweep through different space and time pieces and thereby is not easy to modify for online processing. GeoBurst [134] generates candidate events by identifying pivot tweets based on geographical and semantic similarities and ranks them using spatiotemporal burstiness to filter out noisy ones. TrioVecEvent learns multimodal embeddings of tweets to address the information on location, time and text during clustering and is reported to achieve much better accuracy than its baseline approaches. However, neither of these two methods actively performs event detection on a given tweet stream unless a query time window is specified.

Due to the sparsity of geotagged tweets (1%), some methods try to acquire more local tweets by tracking local people [118, 114]. The location information in these methods, however, are usually in a very coarse resolution (e.g., city-level) and rarely used when grouping tweets together. Some methods try to first detect an event and then estimate its location afterwards, e.g., TwitterStand [102]. These methods are different from our focus as we instead try to extract local events from geotagged tweet streams.

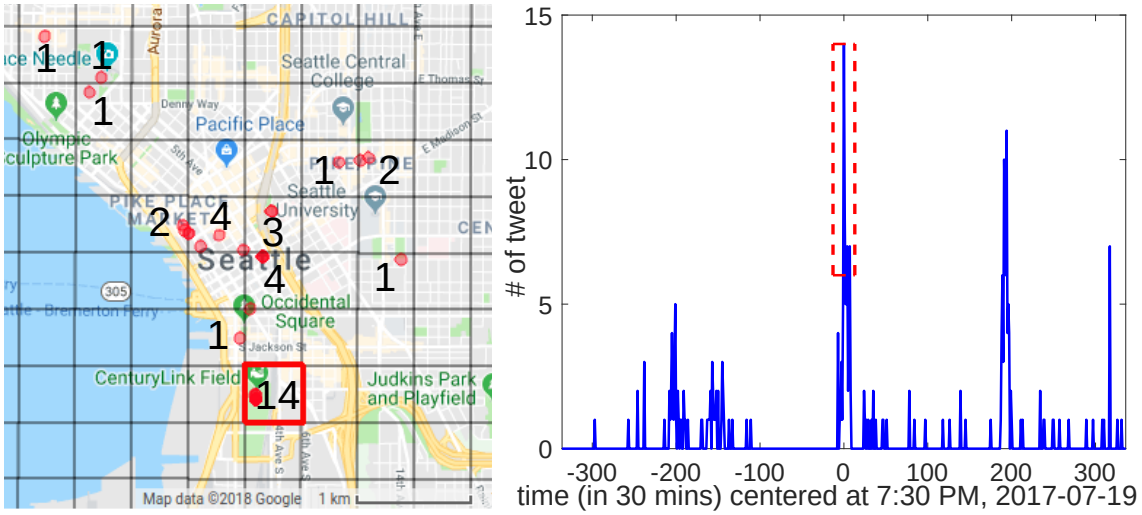
3.3 Preliminaries

3.3.1 Problem

Given a geotagged tweet stream, our goal is to identify the latest local events. Formally, suppose that t is the current (latest) time point and Δt is a short time interval, we define $\mathcal{D}_t$ to be the geotagged tweet stream up to t, and $\mathcal{D}_{t-\Delta t \to t}$ be the geotagged tweet stream from $t - \Delta t$ to t. In other words, $\mathcal{D}_{t-\Delta t \to t}$ essentially represents the latest geotagged tweets with respect to Δt. For simplicity, a geotagged tweet d can be seen as a tuple $\langle time_d, loc_d, txt_d, user_d \rangle$ in which $time_d$ is the publication time, loc_d is the geographical location (i.e., a pair of lat/long coordinates), txt_d refers to the textual content and $user_d$ is the user posting this tweet. The latest local event detection problem is then to extract from $\mathcal{D}_{t-\Delta t \to t}$ all possible local events, where each event is a cluster of geographically, temporally and semantically close tweets.

Typically, the occurrence of a local event often brings about an unusually considerable amount of relevant tweets at the happening location for a certain time period. For example, a soccer game was set to start at 7:30 PM at the stadium "CenturyLink Field" near the center of the city of Seattle. Many tweets with keywords "Sounders", "soundersfc" and "CenturyLink Field" etc (shown in Figure 3.1) was posted at that location during the game. Figure 3.2 shows that an unusual amount of such tweets were observed both geographically and temporally.

Motivated by the above observation, we propose a prediction-based method for detecting the latest local events, called DeLLe. The key idea of DeLLe is to first detect



(a) (b)

Figure 3.2: The soccer game in Figure 3.1 brings about an anomalous amount of tweets both spatially and temporally. (a) Spatial distribution of the tweets around the stadium at 7:30 PM - 8:00 PM. The stadium lies in the red square. Each red dot is a tweet, and the number in a grid cell refers to its number of tweets while an empty cell means no tweets. (b) Temporal distribution of the tweets at the game stadium. The tweets are aggregated every 30 minutes.

spatiotemporal unusualness as possible candidates of local events and then select the ones that most likely corresponding to local events.

3.3.2 System Overview

Figure 3.3 demonstrates DeLLe's overall design. DeLLe can work in two modes: batch mode and online mode. The major difference is that the batch mode exploits a disjoint discretization in the time dimension while the online mode utilizes a continuous sliding time window and correspondingly a set of updating modifications for online processing. We will detail these modifications in Section 3.5.

We utilize a uniform grid to tessellate the spatial region into squares of size $\Delta l \times \Delta l$, where Δl is the side length of the square. Although more complex tessellation struc-

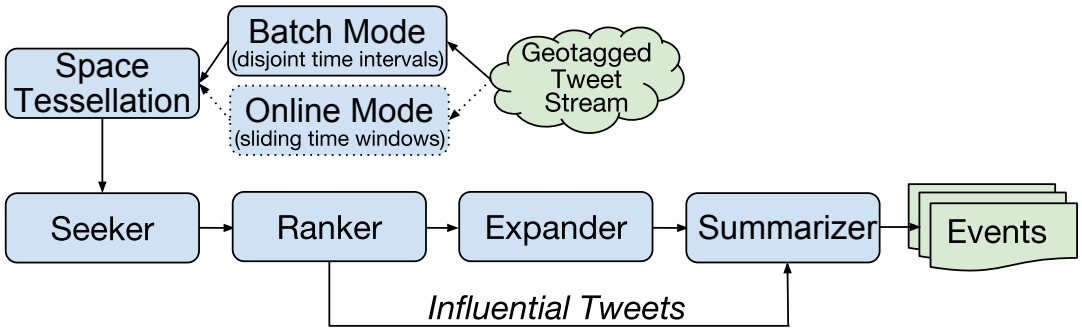


Figure 3.3: System overview.

tures [51, 58, 80] have been explored, grid tessellation is the simplest yet most commonly used way of subdividing geographical space [104, 5, 50, 113, 49]. More importantly, it enables us to exploit state-of-the-art spatiotemporal tweet count prediction model [116] by treating each grid cell as a pixel and therefore the whole grid as image-like data. In reality, local events may not fall neatly on the grid cell boundaries. Therefore, we propose a module of *expander* to connect nearby cells which share similar content to alleviate that issue.

After discretizing space, the tweets are subsequently fed into a pipeline of four modules: *seeker*, *ranker*, *expander* and *summarizer*. *Seeker* finds spatiotemporal unusualness in the number of tweets as potential candidates of local events. *Ranker* selects which set of unusualness found by the *seeker* are most likely to be local events. Afterwards, the *expander* tries to infer a local event's span in both time and space under the metric of semantical similarity. At last, the *summarizer* generates an abstract for a detected local event by selecting the latest top influential tweets.

3.4 The Batch Mode

In this section, we present the workflow of DeLLe in its batch mode. In order to detect the local events from $\mathcal{D}_{t-\Delta t \to t}$, we discretizes the geotagged tweet stream into a set of disjoint intervals, i.e.,$\{\cdots [t-2\Delta t, t-\Delta t), [t-\Delta t, t)\}$. In the following, we explain how to do tweet count prediction, unusualness detection, event expansion, and summarization for this time series of tweets.

3.4.1 Seeker

After tessellating the space into an $M \times N$ grid and discretizing time into periods of length Δt, the task of seeker is to identify grid cells that show an unusual aggregation of tweets in latest geotagged tweet stream $\mathcal{D}_{t-\Delta t \to t}$ or $\mathcal{D}_T$ where T denotes the last time interval of length Δt.

3.4.1.1 Tweet Count Prediction

The goal of tweet count prediction is to use previously historical tweet count data in a local region to forecast the number of tweets to appear in the next time step [116]. On an $M \times N$ grid map, the tweet count values in the grid cells at time step τ can be written in a tensor $\mathbf{X}_\tau \in \mathbb{R}^{M \times N}$ where $\mathbf{X}_\tau(m,n)$ is the tweet count in the grid cell (m,n) at time step τ. Therefore, the prediction problem is formulated as follows:

Definition 2. *The tweet count prediction problem $\mathcal{P}$ is to generate a prediction $\mathbf{Y}_T$, which is an estimation of $\mathbf{X}_T$, given a list of historical observations $\{\mathbf{X}_\tau | \tau = 0, \cdots, T-1\}$.*

Related Approaches As time goes by, the tweet counts in a region may be formulated as time series data, which enables the exploitation of the techniques like historical average and autoregressive integrated moving average (ARIMA) [42]. For example, TwitInfo [83] uses the weighted average of historical tweet counts to compute the expected frequency of tweets. Lin et al. [74] proposed a space-time autoregressive integrated moving average (STARIMA) model to predict urban traffic flow volume. Moreover, Chae et al. [17] adopt a similar model to seasonal ARIMA and decomposes the time series into the sum of a seasonal part, a trend part, and a remainder part, to check whether there exists an unusual volume of tweets.

Time series analysis based techniques, however, often neglect the effects exerted by nearby geographical regions when making predictions on a specific local region. Therefore, in their work on finding anomalies, Krumm and Horvitz [58] build a gradient boosting regression function that estimates the number of tweets on a region based on a list of features including the time of the day, the day of the week, and the tweet counts from neighboring regions.

With the recent advances in deep learning, a few recent studies have focused on introducing deep neural networks into modeling spatiotemporal data [15, 106]. For example, Shi et al. [103] propose a novel *convolutional LSTM* (*ConvLSTM*) network for precipitation on radar echo spatiotemporal data, which enables the capture of both spatial and temporal correlation simultaneously by combining a convolution network and a recurrent LSTM network. Such a combination is done by innovatively replacing the matrix multiplication operations used in LSTM with convolution operations. This is different from Spatiotemporal Recurrent Convolutional Networks (SRCN) proposed in [126]

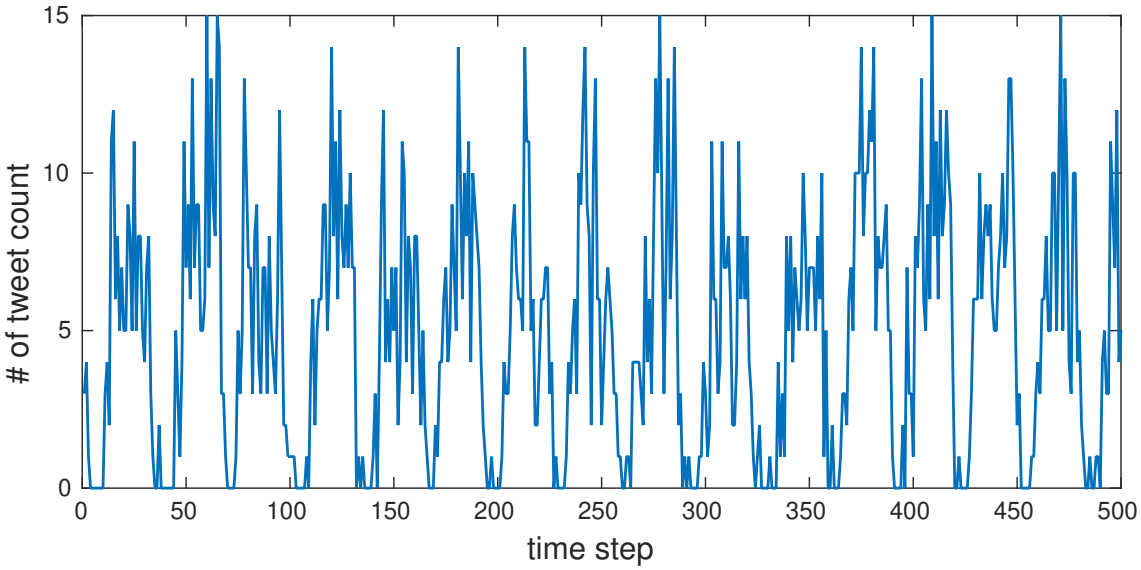
which simply stack additional LSTM layers after convolutional layers.

Focusing on citywide crowd prediction, Zhang et al. [136] first partition historical spatiotemporal sequences into three subsets *closeness*, *period* and *trend*, which correspond to recent, near and distant history. Each subset is then fed into a Deep Convolution Neural Network to yield a prediction, and such predictions are then fused together along with external features such as week-of-day to produce the final forecast. Moreover, their subsequent work [137] further introduces the residual network [38] to capture citywide spatial dependence and gives better accuracy. Our method is different from them in the sense that we utilize ConvLSTM layers instead of regular convolution layers to build up our model, which shows effectiveness in our dataset.

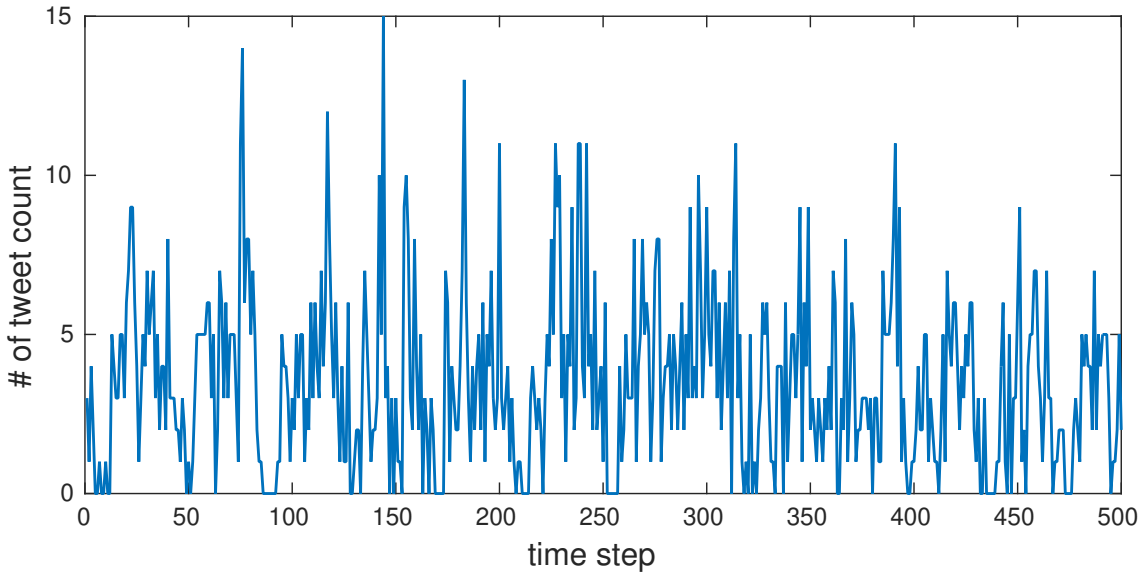
Our Prediction Model Making high-quality predictions of tweet count in a region is challenging due to complex spatial and temporal dependencies. For instance, if there are consistent bursty tweets at a coffee shop (e.g., Starbucks) in the morning, it should not be mistakenly reported as unusual. The advances in deep learning have motivated a few recent studies to introduce deep neural networks into modeling spatiotemporal data for better capturing spatial and temporal dependences [136, 137].

In this chapter, we presented a residual Convolutional LSTM (ConvLSTM [103]) based prediction model [116], which exhibited state-of-the-art accuracy. In the following, we give a brief introduction to this tweet count prediction model [116]. Figure 3.5 illustrates the structure of the neural network model. Zhang et al. [137, 136] pointed out that making predictions on spatiotemporal data relies on not only the observations of *recent time* but also those in *near history* and *distant history*, and model these temporal dependencies as temporal *closeness*, *period* and *trend*. A similar observation on tweet count

data is also found [116]. For example, Figure 3.4 draws the tweet counts in a region



(a)



(b)

Figure 3.4: Temporal pattern. (a) Seattle City; (b) NYC. Time step is in the unit of 30 minutes, starting from 18:30 on 2016-06-15.

for 500 time steps in the city of Seattle and NYC, respectively. The results show that our data indeed have certain temporal periodical pattern. As a result, in order to predict an expected tweet count Y_t at time step t, we break the historical observations to extract the

Δl	the side length of a grid cell
t, Δt	current time, length of time interval
τ, T	time step, the latest time step
$\mathcal{D}_t$, $\mathcal{D}_T$	tweet stream up to t, tweet stream during T
$\mathbf{X}_\tau$, $\mathbf{Y}_\tau$	tweet count at τ, prediction of tweet count at τ
(m,n)	a grid cell in an $M \times N$ grid map
c, p, q	closeness, period, trend
$\mathbf{E}_T$, $\mathcal{E}_T$	prediction error, a list of prediction errors up to T
$k_{\Delta \mathbf{E}'_T}$	unusualness threshold
$\mathcal{Y}_T(m,n)$	a list of historical predictions on (m,n)
$\mathcal{W}_T(m,n)$	the set of keywords in (m,n) at T
$SDD^w_T(m,n)$	spatial density distribution of word w in (m,n) at T
$TS(d',d'')$	topical similarity between tweet d' and d''
$TB_T(m,n)$	temporal burstiness of cell (m,n) at T
$SB_T(m,n)$	spatial burstiness of cell (m,n) at T
$TC_T(m,n)$	topical coherence of cell (m,n) at T
$TEC_T(m,n)$	the set of Temporal Expansion Cells of cell (m,n)
$SEC_T(m,n)$	the set of Spatial Expansion Cells of cell (m,n)
k	the number of tweets for event summarization and topical coherence

Table 3.1: List of main notations in Section 3.4

closeness, *period* and *trend* dependence sequences X^c_t, X^p_t and X^q_t.

To be specific, the *closeness* sequence is a list of l_c continuous tweet count values right before the current time step and is denoted by $X^c_\tau = \begin{bmatrix} X_{\tau-l_c} & X_{\tau-(l_c-1)} & \cdots & X_{\tau-1} \end{bmatrix}$. The *period* sequence is a l_p-long list of historical tweet count values periodically sampled every p interval: $X^p_\tau = \begin{bmatrix} X_{\tau-p\cdot l_p} & X_{\tau-p\cdot(l_p-1)} & \cdots & X_{\tau-p\cdot 1} \end{bmatrix}$. Similarly, the *trend* sequence is a l_q-long list of historical tweet count values periodically sampled but every q interval: $X^q_\tau = \begin{bmatrix} X_{\tau-q\cdot l_q} & X_{\tau-q\cdot(l_q-1)} & \cdots & X_{\tau-1\cdot q} \end{bmatrix}$. In practice, p is set to a duration of one-day to capture daily periodicity and q to one-week to reveal weekly trend. Each of X^c_τ, X^p_τ and X^q_τ is separately fed into three designated neural networks which shares the same structure but with different weights, to generate separate predictions Y^c_τ, Y^p_τ and Y^q_τ, respectively. In the end, a parametric-matrix-based fusion process is applied to combine the 3 prediction

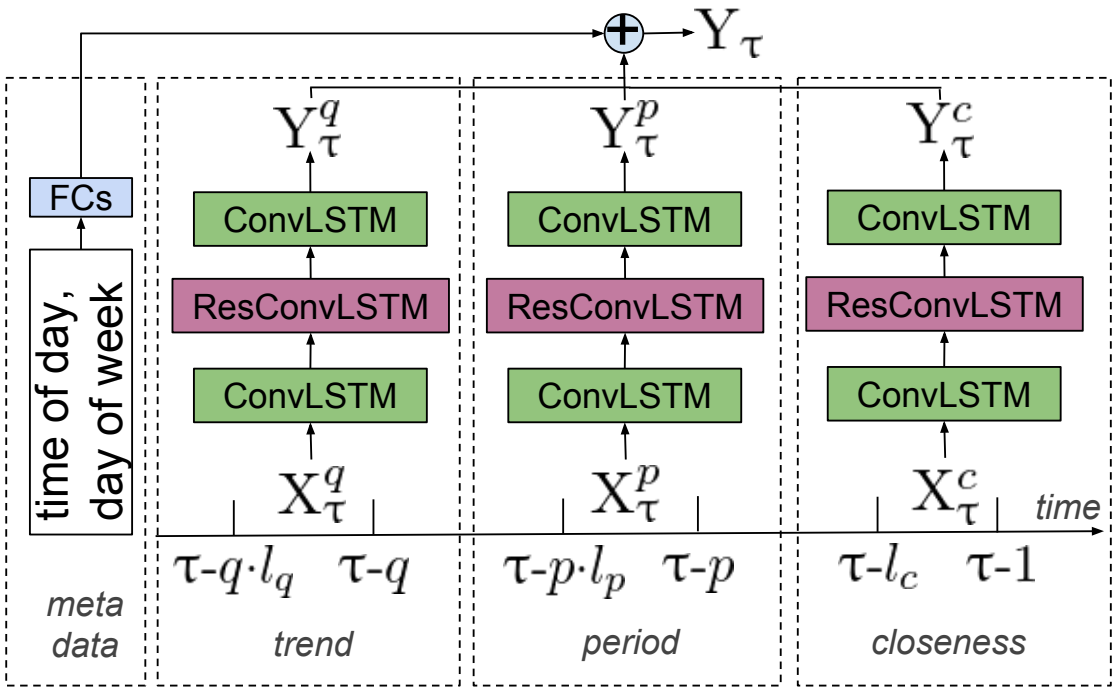


Figure 3.5: Tweet count prediction model. ResConvLSTM: Residual ConvLSTM block; FCs: Fully-Connected Layers, i.e. Dense layers.

results, together with the meta-feature of time, to generate the final prediction. Each of the three dependence sequences is then fed into a designated network with the same structure but different weights to get the three predictions Y_t^c, Y_t^p and Y_t^q, respectively.

In the following, we will briefly explain the key stacking blocks in Figure 3.5, including the ConvLSTM layer, the ResConvLSTM block, the FC layer and temporal properties fusion.

1. *ConvLSTM layer*.

   The Long Short-Term Memory (LSTM) network, one of the well-known recurrent neural networks, has achieved great success in many applications such as sequence modeling and especially sequence prediction [33, 43, 105]. Despite its strong ability in modeling temporal dependences of sequences, LSTM ignores spatial informa-

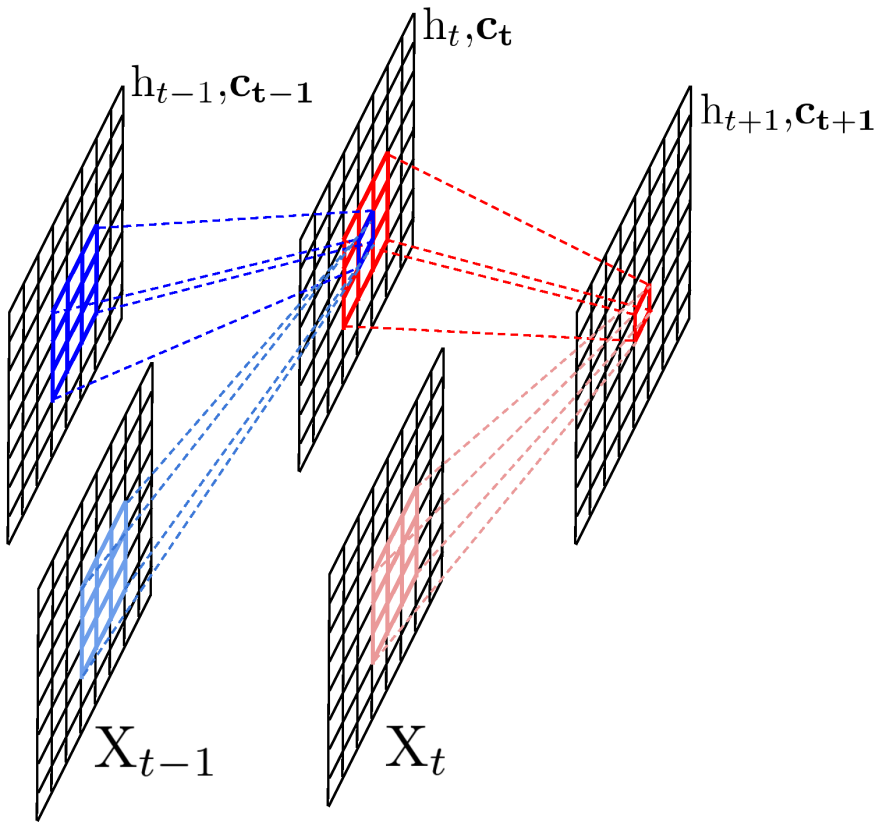


Figure 3.6: The inner structure of ConvLSTM. The LSTM matrix multiplication is replaced with convolution.

tion when the sequence data is multi-dimensional. To overcome this drawback, Shi et al. [103] proposed the Convolutional LSTM (ConvLSTM) which innovatively uses a convolution operator in the state-to-state and input-to-state transitions (see

Figure 3.6). The key equations in ConvLSTM are shown as follows:

$$
\begin{aligned}
\mathbf{i_t} &= \sigma(\mathbf{W}_{xi} * \mathbf{X}_t + \mathbf{W}_{hi} * \mathbf{h}_{t-1} + \mathbf{W}_{ci} \circ \mathbf{c}_{t-1} + \mathbf{b}_i) \\
\mathbf{f_t} &= \sigma(\mathbf{W}_{xf} * \mathbf{X}_t + \mathbf{W}_{hf} * \mathbf{h}_{t-1} + \mathbf{W}_{cf} \circ \mathbf{c}_{t-1} + \mathbf{b}_f) \\
\mathbf{c_t} &= \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ tanh(\mathbf{W}_{xc} * \mathbf{X}_t + \mathbf{W}_{hc} * \mathbf{h}_{t-1} + \mathbf{b}_c) \\
\mathbf{o_t} &= \sigma(\mathbf{W}_{xo} * \mathbf{X}_t + \mathbf{W}_{ho} * \mathbf{h}_{t-1} + \mathbf{W}_{co} \circ \mathbf{c}_t + \mathbf{b}_o) \\
\mathbf{h_t} &= \mathbf{o}_t \circ tanh(\mathbf{c}_t)
\end{aligned}
\tag{3.1}
$$

where t iterates from 1 to $T-1$. The variables $\mathbf{X}_t$, $\mathbf{c}_t$, $\mathbf{h}_t$, $\mathbf{i}_t$, $\mathbf{f}_t$, and $\mathbf{o}_t$ are tensors to represent values of the inputs, cell outputs, hidden states, input gates, forget gates and output gates. σ is a logistic sigmoid function. The operator $\circ$ denotes the Hadamard product, i.e., element-wise product of matrix. And $*$ denotes the convolution operator instead of matrix multiplication, which is a key difference from FC-LSTM [33]. At last, $\mathbf{W}_*$ and $\mathbf{b}_*$ are weight and bias matrices parameters which need to be learned during training.

2. *ResConvLSTM block*.

It is well known that deeper networks can model more complex functions and thus are more expressive. However, networks that work well in practice usually cannot be very deep. This is due to the vanishing gradient problem. To avoid this vanishing gradient problem and make the design of a deeper network possible, [38] proposed skip connections which directly link the output of lower layers to the input of higher layers. This shortcut has proven to be effective to alleviate the vanishing gradient problem in the training process and achieved significantly better performance in

many applications. Recently, [69] has shown that skip connections can also help to prevent the loss function from being chaotic, leading to a more convex loss function, and thus, making it easy to find a good local minimum. Essentially, a residual building block can be defined as:

$$\mathbf{Y} = \mathcal{F}(\mathbf{X}) + \mathbf{X}, \tag{3.2}$$

where $\mathbf{X}$ and $\mathbf{Y}$ are the input and output tensors of the residual block. The function $\mathcal{F}$ represents several convolutional or ConvLSTM layers [39, 137, 138]. We use the ConvLSTM [103] to assemble the residual block, which is illustrated in Figure 3.7. This is a key difference from ST-ResNet [137] which uses a regular convolutional layer instead as shown in Figure 3.7.

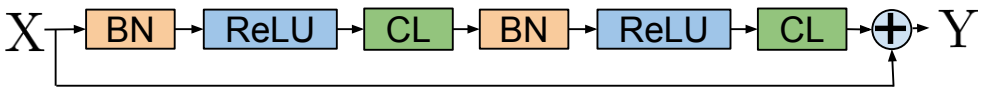


Figure 3.7: ResConvLSTM block. BN: Batch Normalization; ReLU: Rectifier Linear Unit; CL: ConvLSTM

3. *FC layer*.

   To help capture the regular time-varying changes, meta-data features such as time-of-day, day-of-week are also hooked in the model by stacking two fully-connected layers. The first is an embedding layer for features and the second maps from low to high dimensions to make the output have the same shape as the target [137].

4. Temporal Properties Fusion

   Zhang et al. [137, 136] pointed out that in spatiotemporal data sequences, mak-

ing predictions on the future observations does not only rely on the observations of *recent time* but also depends on those in *near history* and *distant history*. Such temporal dependencies are modeled as temporal *closeness*, *period* and *trend*. More specifically, the temporal *closeness* dependence sequence is a l_c-long list of consecutive observations before the current time step and can be denoted by $X_t^c = \begin{bmatrix} X_{t-l_c} & X_{t-(l_c-1)} & \cdots & X_{t-1} \end{bmatrix}$. The temporal *period* dependence sequence is a l_p-long list of historical observations which are chosen with a time interval p: $X_t^p = \begin{bmatrix} X_{t-p\cdot l_p} & X_{t-p\cdot(l_p-1)} & \cdots & X_{t-p\cdot 1} \end{bmatrix}$. Similarly, the temporal *trend* dependence sequence is a defined as a l_q-long list of historical observations which are also periodically chosen but with the time interval q: $X_t^q = \begin{bmatrix} X_{t-q\cdot l_q} & X_{t-q\cdot(l_q-1)} & \cdots & X_{t-1\cdot q} \end{bmatrix}$. In practice, p is set to a period of one-day to capture daily periodicity and q is set to one-week to reveal weekly trend.

Each of X_t^c, X_t^p and X_t^q are separately fed into three designated neural networks, which have the same structure but different weights, to generate observation predictions Y_t^c, Y_t^p and Y_t^q, respectively. At last, a parametric-matrix-based fusion is adopted to combine the three outputs Y_t^c, Y_t^p and Y_t^q to yield the final prediction $\mathbf{Y_t}$ [137] using the following equation:

$$\mathbf{Y_t} = \mathbf{W^c} \circ \mathbf{Y_t^c} + \mathbf{W^p} \circ \mathbf{Y_t^p} + \mathbf{W^q} \circ \mathbf{Y_t^q} \tag{3.3}$$

where $\mathbf{W}^*$ are weight matrices that balance different components. Additionally, features such as the time of the day and the day of the week can also be incorporated into $\mathbf{Y_t}$ using fully-connected layers.

Except for the output ConvLSTM layer which has only 1 hidden state, all ConvLSTM layers are configured to have 32 hidden states. Since we only focus on predicting the expected spatiotemporal tweet count for the next time step, we set the output ConvLSTM layer to return one prediction sequence.

We define the size of the filter in our ConvLSTM to be 3×3. This is because the spatial correlation of tweet count data is quite local, i.e., the number of tweets in a grid is correlated with the ones in the nearby grids instead of grids farther away. For example,

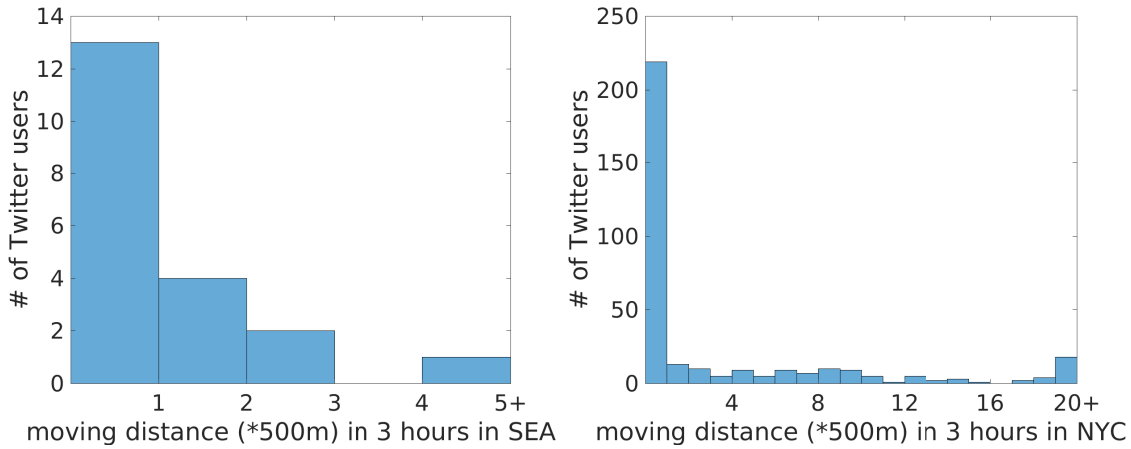


Figure 3.8: Histogram of moving distance of Twitter users. We only consider Twitter users who have 2 or more geotagged tweets in the 3-hour time period starting from 18:30 on 2016-06-15. The moving distance of a user is calculated as the largest distance between the GPS coordinates in his geotagged tweets.

Figure 3.8 shows the histogram of moving distance of Twitter users during a time period of 3 hours in the city of Seattle and NYC, respectively. We notice that the majority of Twitter users travel less than 500 meters, i.e. less than the size of a grid cell.

Comparing with ST-ResNet [137], we replace its regular convolutional layers with ConvLSTM, as the latter is more powerful in capturing temporal dependence. Moreover, we stack only one residual block, instead of multiple blocks, because we empirically

notice that adding more layers to our model cannot improve the performance of the model and sometimes results in over fitting. This also corresponds to the fact that Twitter users in our dataset usually have shorter moving distances.

3.4.1.2 From Prediction To Unusualness

We define the prediction error to be $\mathbf{E}_T = \mathbf{Y}_T - \mathbf{X}_T$, where $\mathbf{X}_T$ is the latest tweet count on a spatial $M \times N$ grid and $\mathbf{Y}_T$ is the prediction of $\mathbf{X}_T$. $\mathbf{E}_T(m,n)$ indicates the prediction error of the grid cell (m,n). Intuitively, a significant negative $\mathbf{E}_T(m,n)$ indicates a local event as there were many more tweets than usual. Following [58], we define precision of our prediction model to be $\sigma_{\mathcal{E}_T}$, where $\sigma_{\mathcal{E}_T}(m,n)$ is the standard deviation of the grid cell (m,n) w.r.t. its history of prediction errors $\mathcal{E}_T(m,n) = \{\cdots \mathbf{E}_{T-1}(m,n), \mathbf{E}_T\}$. To account for the precision of the prediction model, we re-define the prediction error as:

$$\mathbf{E'}_T = \mathbf{E}_T \oslash \sigma_{\mathcal{E}_T} \tag{3.4}$$

where $\oslash$ denotes the element-wise division operation.

To detect unusual grid cells using $\mathbf{E'}_T$, we utilize an image restoration framework called Deep Image Prior [108]. Our intuition is that the unusualness in $\mathbf{E'}_T$ is like spike noise in an image, and Deep Image Prior can be used to denoise corrupted images without prior knowledge of training data. Suppose that $\mathbf{E''}_T$ is the restored image of $\mathbf{E'}_T$, and $\Delta\mathbf{E'}_T = \mathbf{E''}_T - \mathbf{E'}_T$, we claim a grid cell (m,n) is unusual if

$$|\Delta\mathbf{E'}_T(m,n) - \mu_{\Delta\mathbf{E'}_T}| \geq k_{\Delta\mathbf{E'}_T} \cdot \sigma_{\Delta\mathbf{E'}_T} \tag{3.5}$$

where $\mu_{\Delta\mathbf{E}'_T}$ and $\sigma_{\Delta\mathbf{E}'_T}$ are the mean and standard deviation of grid cells in $\Delta\mathbf{E}'_T$, respectively. $k_{\Delta\mathbf{E}'_T}$ is a predefined threshold for determining the unusualness of a grid cell. Different from [58], our approach accounts for both history of a grid cell and information of other cells on the whole region when detecting unusualness in a location. This is important in differentiating global events which might cause an unusual number of tweets on a local grid cell.

3.4.2 Ranker

The *seeker* module described above outputs a set of unusual grid cells. In this section, we make a ranking of these unusual locations to identify the top ones that are most likely corresponding to the occurrence of local events, by addressing temporal burstiness, spatial burstiness and topical coherence.

3.4.2.1 Temporal Burstiness

For a grid cell (m,n), suppose that $\mathcal{Y}_T(m,n)$ represents a history of estimations on its number of tweets up to the time step T, and is defined as:

$$\mathcal{Y}_T(m,n) = \{\cdots \mathbf{Y}_{T-1}(m,n), \mathbf{Y}_T(m,n)\} \tag{3.6}$$

Then we use z-score to quantify the grid cell (m,n)'s temporal burstiness [134] at T, denoted as $TB_T(m,n)$ and defined as:

$$TB_T(m,n) = \frac{\mathbf{X}_T(m,n) - \mu_{\mathcal{Y}_T(m,n)}}{\sigma_{\mathcal{Y}_T(m,n)}} \tag{3.7}$$

where $\mu_{\mathcal{Y}_T(m,n)}$ and $\sigma_{\mathcal{Y}_T(m,n)}$ are the mean and standard deviation of $\mathcal{Y}_T(m,n)$, respectively. Recall that $\mathbf{X}_T(m,n)$ is the actual number of tweets in grid cell (m,n) at time step T.

3.4.2.2 Spatial Burstiness

Given a grid cell (m,n), the spatial burstiness is measured by the spatial density distribution of keywords of the tweets in (m,n). The intuition is that a low spatial density distribution means that the keyword is widely spread over space and a high distribution means that the keyword occurs only at a few locations. Therefore, the keywords in local events should have higher spatial density distribution to be spatially bursty.

Suppose that $\mathcal{D}_T(m,n)$ is the tweet set in grid cell (m,n) at T, and $\mathcal{W}_T(m,n)$ is the set of keywords (e.g., after removing stop words) in (m,n), i.e., $\mathcal{W}_T(m,n) = \{w \mid w \in txt_d \text{ and } d \in \mathcal{D}_T(m,n)\}$. Let $SDD_T^w(m,n)$ be the spatial density distribution of keyword w in grid cell (m,n) at T, i.e.,

$$SDD_T^w(m,n) = \frac{\#\text{ of } w \text{ in grid cell } (m,n)}{\sum\limits_{(m',n')\in M\times N} \#\text{ of } w \text{ in grid cell } (m',n')} \tag{3.8}$$

We now define the spatial burstiness of grid cell (m,n) as:

$$SB_T(m,n) = \sum_{w\in\mathcal{W}_T(m,n)} SDD_T^w(m,n) \tag{3.9}$$

3.4.2.3 Topical Coherence

The topical coherence is to capture the semantical similarity of tweets in a grid cell. In other words, the tweets posted on the same event should be discussing similar content and probably using similar vocabularies. Twee2Vec [28] learns the vector-space representations of tweets using a character-based bi-directional recurrent neural network model, and has been demonstrated to have good performance in the application of clustering semantically similar tweets together [109]. To measure the topical similarity between tweets, we use Tweet2Vec implementation[1] to encode a textual tweet in character sequence to a vector embedding with a default dimension size of 500.

Let $TS(d', d'')$ be the topical similarity between tweets d' and d''. To measure the topical coherence of the tweets in cell (m,n), we construct a graph, called the *Tweet Influence Graph*.

Definition 3. *(Tweet Influence Graph). The tweet influence graph on the grid cell* (m,n) *at* T*, is an undirected graph* $G_T = (V_T, E_T)$ *where* V_T *is the set of all tweets in* $\mathcal{D}_T(m,n)$*,* E_T *is the set of edges between tweets, and the weight of an edge between* d' *and* d'' *is their topical similarity* $TS(d', d'')$*.*

We now employ PageRank [90], a random walk procedure, on the tweet influence graph to bring orders to the influence of tweets in $\mathcal{D}_T(m,n)$ and thus identify the top k tweets with the most influence, denote by $\mathcal{D}_T^k(m,n)$. The *topical coherence* is thus defined

[1] https://github.com/vendi12/tweet2vec_clustering

as:

$$TC_T(m,n) = \frac{\sum_{d' \in \mathcal{D}_T^k(m,n), d'' \in \mathcal{D}_T^k(m,n)} TS(d', d'')}{k^2} \tag{3.10}$$

The rationale is that if the tweets in $\mathcal{D}_T(m,n)$ are about the same local event, then the most topically influential tweets should have higher topical similarity between each other. One may point out that such a topical coherence measurement would suppress a grid cell having multiple topically unrelated ongoing events. We argue that such a case is very rare with a fine space and time discretization.

3.4.2.4 Ranking Function

As the final step, we now define the ranking score of the grid cell (m,n) by aggregating its *temporal burstiness*, *spatial burstiness* and *topical coherence*, after rescaling them to $[0,1]$ with respect to other grid cells:

$$\mathcal{R}_T(m,n) = TB'_T(m,n) \cdot SB'_T(m,n) \cdot TC'_T(m,n) \tag{3.11}$$

where $TB'_T(m,n) = (TB_T(m,n) - TB_T^{min})/(TB_T^{max} - TB_T^{min})$ with TB_T^{max} and TB_T^{min} being the maximum and minimum of topical burstiness among all grid cells at T. Spatial burstiness and topical coherence are rescaled in the same way, receptively.

3.4.3 Expander

Suppose that we choose the top K unusual grid cells after ranking at T, and claim that they are the candidates most likely to be local events in $\mathcal{D}_T$. In reality, different local events might have different spatial and temporal ranges, e.g., spanning over a larger region than the grid cell size $\Delta l * \Delta l$ or for a longer duration than the time discretization interval Δt. We therefore, in this section, try to infer the spatiotemporal range of these local event candidates.

The basic idea of expander is to connect (spatially or temporally) nearby grid cells if they share similar content. As presented in Algorithm 1, the expansion consists of two parts: temporal expansion and spatial expansion. The temporal expansion checks whether the occurrence of previous event candidates continues to the present, and updates them if so. The spatial expansion examines whether nearby grid cells are relevant to the same event.

During the expansion, for each event candidate, we maintain a grid cell as its *event-focus* grid cell. The event-focus grid cells are initially set to be the most unusual cells (i.e., the top ranking cells in Equation 3.11). As time proceeds, the event-focus grid cell of an event might stay at the same grid cell (e.g., a sit-down protest), or move to another one (e.g., a demonstration protest), or simply no longer exists (e.g., the ending of an event). Meanwhile, new event-focus grid cells might join as well if new events happen. Note that during the spatial expansion, several event-focus grid cells might exist adjacently and need to be merged if they are about the same content. For a given cell (m,n), we denote by $SEC_T(m,n)$ (Spatial Expansion Cells) the cells for it to examine

for spatial expansion at T, and similarly $TEC_T(m,n)$ (Temporal Expansion Cells) for temporal expansion. $SEC_T(m,n)$ and $TEC_T(m,n)$ are defined as follows.

$$\begin{aligned} SEC_T(m,n) &= \{(m \pm i, n \pm j)_T \mid i,j \in \{-1,0,1\}\} \setminus \{(m,n)_T\} \\ TEC_T(m,n) &= SEC_T(m,n) \cup \{(m,n)_T\} \end{aligned} \tag{3.12}$$

The spatial range for expansion is currently set to adjacent cells incident at an edge or vertex and can extend further if necessary.

Whether or not two adjacent grid cells are connected depends on their content similarity. We treat each grid cell containing its tweets as a document and thus build a term-document matrix. In this matrix, each row represents a token (non-stop words) in tweets, each column represents a document, i.e., a grid cell, and each element can refer to the token frequency (or TF-IDF) per document. Next, the content similarity between two grid cells can be calculated by their corresponding column vectors under the metric of cosine similarity. More advanced document similarity techniques such as Latent Semantic Analysis (LSA) [61] may further be applied on the term-document matrix to measure the similarities between documents at a lower rank. It is, however, usually a time-consuming process due to the introduction of Singular Value Decomposition (SVD). For approximation as well as efficiency, we limit each column vector to contain the information on its most frequent k_{cs} tokens during cosine similarity calculation, where k_{cs} is a predefined value.

Algorithm 1: Expander

Input: C_{T-1} — the set of *event-focus* grid cells at time $T-1$, C'_T — the top K ranking grid cells at time T (i.e., the newly identified *event-focus* grid cells), ε_{cs} — content similarity threshold.

Output: The updated *event-focus* grid cells C_T at time T

/* Temporal Expansion */

foreach *event-focus grid cell* $(m,n)_{T-1} \in C_{T-1}$ **do**

  // $TCC_T(m,n)$ are the grid cells at time T

  // that are temporally connected to $(m,n)_{T-1}$

  $TCC_T(m,n) \longleftarrow \{\, c \mid c \in TEC_T(m,n),$ $\varepsilon_{cs} \leq \texttt{ContentSimilarity}(c, (m,n)_{T-1})\}$;

  if $TCC_T(m,n) \neq$ **then**

    $c' = \underset{c \in TCC_T(m,n)}{\text{argmax}}\ \texttt{ContentSimilarity}(c, (m,n)_{T-1})$;

    // c' is now a new *event-focus* grid cell

    // transited from $(m,n)_{T-1}$

    $C'_T \longleftarrow C'_T \bigcup \{c'\}$;

/* Spatial Expansion */

foreach *event-focus grid cell* $(m,n)_T \in C'_T$ **do**

  // $SCC_T(m,n)$ are the grid cells at time T

  // that are spatially connected to $(m,n)_T$

  $SCC_T(m,n) \longleftarrow \{\, c \mid c \in SEC_T(m,n),$ $\varepsilon_{cs} \leq \texttt{ContentSimilarity}(c, (m,n)_T)\}$;

  // If $(m,n)_T$ is spatially connected to other

  // *event-focus* grid cells, merge them.

  if $SCC_T(m,n) \bigcap C'_T \neq$ **then**

    $Temp = (SCC_T(m,n) \bigcap C'_T) \bigcup \{(m,n)_T\}$;

    $c' = \underset{c \in Temp}{\text{argmax}}\ \texttt{TopicalCoherence}(c)$;

    $C'_T \longleftarrow (C'_T \setminus Temp) \bigcup \{c'\}$;

return C'_T

3.4.4 Summarizer

The module of *summarizer* selects the most representative tweets from a cluster of tweets in an event, and thereby produce a succinct description. When summarizing an event across several time steps, the tweets at the latest time step T are preferred to earlier ones in order to reflect the newest dynamic updates on events.

The general idea of event summarization expects that the tweets associated with

the event demonstrate a meaningful description of the event for human consumption [58]. For this purpose, we exploit the most influential tweets in the grid cells. As discussed in Section 3.4.2.3, $\mathcal{D}_T^k(m,n)$ consists of the top k tweets with the most influence at the grid cell (m,n). The summarization works as follows. First, if an event is limited to one grid cell, then its top k tweets are the summarization set of tweets. i.e., $\mathcal{D}_T^k(m,n)$. Second, if an event impacts several grid cells, then we look at the top grid cells with the largest topical coherence scores defined in Equation 3.10 to select which tweets to form the summarization. To be specific, suppose that an event e's spatial impact at T consists of a set of grid cells, denoted by SI_T^e. The subset of SI_T^e used for summarization is defined as:

$$SISum_T^e = \{(m^i,n^i)|\ i = 1 \cdots\ k'\} \tag{3.13}$$

where $1 \leq k' \leq k$, specifying that the summarization tweets are only from the top-k' grid cells with the largest topical coherence scores in SI_T^e. The topical coherence score in each grid cell weighs how many tweets it will contribute to the summarization. For example, let $TC_T(m^i,n^i)$ be the i-th largest topical coherence score, then the number of tweets its grid cell (m^i,n^i) should contribute is:

$$k^i = round(\frac{TC_T(m^i,n^i)}{\sum_1^{k'} TC_T(m^i,n^i)}) * k \tag{3.14}$$

Such k^i tweets come from the top k influential tweets in (m^i,n^i), denoted by $\mathcal{D}_T^{k^i}(m^i,n^i)$.

Therefore, the summarization tweet set is:

$$SumTweets_T^e = \bigcup_i^{k'} \mathcal{D}_T^{k^i}(m^i, n^i) \tag{3.15}$$

3.5 Online Modifications

In this section, we present the modifications that allow DeLLe to process tweets in an approximately online way. The major modification is to utilize a continuously moving sliding window instead of disjoint intervals of time. For example, suppose that the current time is t, the window length is Δt, and the current sliding window is at $[t-2\Delta t, t-\Delta t)$. Then the next sliding window to consider in the online processing is at $[t-2\Delta t+\Delta s, t-\Delta t+\Delta s)$, instead of $[t-\Delta t, t)$ as in the batch mode. Δs denotes the moving step length in the sliding window. In what follows, we describe the changes to the modules in the batch mode needed to enable online processing.

Seeker In the online processing, with a small moving step Δs, two consecutive sliding windows mostly overlap each other and might present little difference. Consequently, if the prediction model takes the previous consecutive windows as the input, it probably generates a prediction very similar to the current sliding window and thus fails to detect anomalous aggregation of tweets. Therefore, to make predictions in the online processing, we still use the data in the previously disjoint time interval as the input. For example, the last time interval in the *closeness* sequence used for predicting the tweet count at $[t-\Delta t, t)$ is $[t-2\Delta t, t-\Delta t)$, instead of $[t-\Delta t-\Delta s, t-\Delta s)$, which is the last sliding time window.

Ranker As the sliding window proceeds, the tweets in the grid cells may also

change, as well as the scoring factors in the ranking Equation 3.11. Recalculating some scores like *spatial burstiness* and *topical coherence* from scratch can be very time-consuming. Therefore, we leverage historical results to update the changes caused by inserting new tweets as well as deleting old tweets. For example, in updating the *spatial burstiness* scores, the system maintains a keyword list which specifies the frequency of a keyword's appearance in each grid cell. Thus, only simple addition or subtraction is necessary for updating frequencies of words. The more complex changes come from updating the scores of *topical coherence* as the tweet influence graph may evolve when inserting new tweets or deleting obsolete tweets. To handle such changes, we exploit OSP [125], a fast random walk algorithm on dynamic graphs using Offset Score Propagation. The core idea of OSP is to first calculate an offset seed vector based on the adjacency difference between old and new graphs.Next, such a seed vector is propagated across the new graph, resulting in offset scores. Finally, OSP adds up the old and offset random walking scores to get the final scores.

Expander The most time-consuming part in this module is calculating the content similarities between grid cells using their most frequent keywords, which may change as news tweets come in or old tweets go away. For a fast implementation, each grid cell maintains a local heap to track the top frequent keywords in it.

Summarizer The summarizer is easy to modify for online processing because the topical coherence of each grid cell and its most influential tweets have already been calculated in the modified ranker module. Therefore, the essential task is to, for each event, maintain a list of top-k' grid cells with the largest topical coherence scores, by using a priority queues.

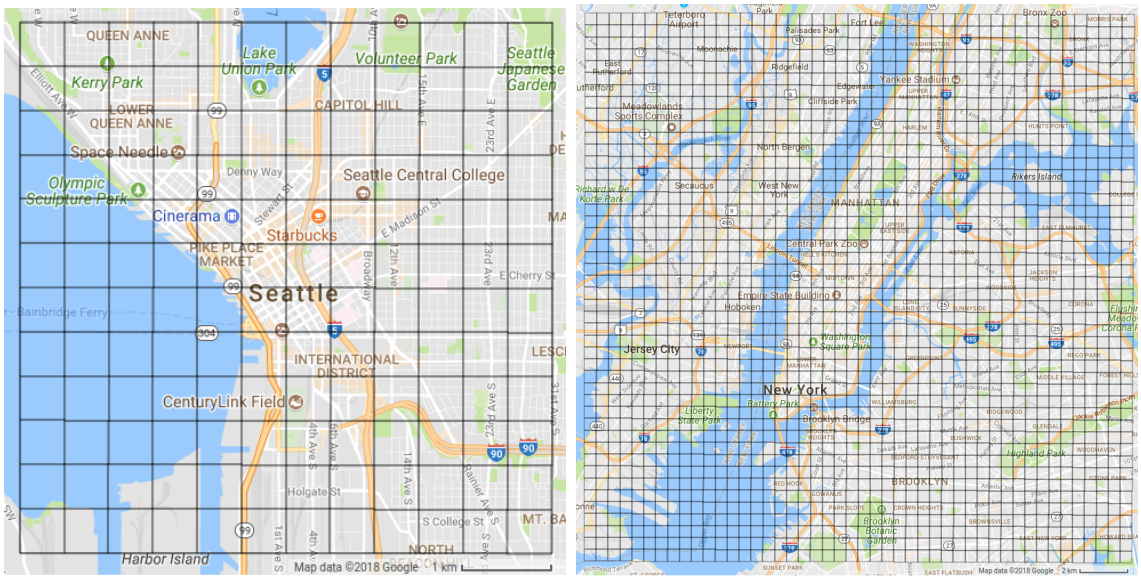
3.6 Evaluation on Tweet Count Prediction

All the experiments in this study are completed on an Nvidia GPU Quadro P6000 and the models are built using Keras [22] libraries with TensorFlow [1] as the backend.

3.6.1 Datasets

We use two sets of geotagged tweets collected from 2015-07-09 to 2017-09-30 in two cities: Seattle, WA (SEA) and New York City (NYC) to carry out all our experiments. The total number of tweets in each dataset is $1,025,181$ and $10,084,839$, respectively. Geotagged tweets are those that contain a pair of longitude and latitude coordinates values which indicate their location. These geotagged tweets are then aggregated into grid cells, which are $500m \times 500m$ squares spanning from [47.579784, -122.373135] to [47.633604, -122.293062] in SEA, and from [40.647984, -74.111093] to [40.853945, -73.837472] in NYC, which correspond to their metropolitan areas, respectively. The two grid maps are shown in Figure 3.9, respectively. In this study, we define the interval of a time step to be 30 minutes, an empirical trade-off between the prediction promptness and accuracy. For example, the task of prediction prefers shorter temporal intervals as it gives more timely results. Shorter temporal intervals, however, might be too small to aggregate enough tweets for making high-quality prediction due to the sparsity of tweets.

Removing Spam Tweets We identify two types of tweets as spam: (1) The tweets whose geographical coordinate values are the same as one of the city centers. Because such tweets are likely posted by accounts who simply give out a nominal location address (e.g., “Seattle, WA” and “New York City”) which are then automatically geodecoded by



(a) (b)

Figure 3.9: (a) 12×12 grid map in Seattle. (b) 46×46 grid map in NYC.

the Twitter location service to city centers. Such accounts send out geo-targeted tweets spams such as "@tmj_sea_legal1" and they are very unlikely to be present exactly at the city centers. We removed 224,335 and 0 tweets for Seattle and NYC in this step. (2) The tweets that are posted by suspicious Twitter users who behave more like bots, e.g., publishing more than 5 tweets at exactly the same location and 3 or more of such tweets are sent out only in 1 minute. We removed 204,800 and 44,389 tweets for Seattle and NYC datasets in this step. After filtering out spam tweets, we now have 756,457 and 9,880,039 tweets in the Seattle and NYC datasets, respectively.

Normalization The values of the tweet count are scaled to $[-1, 1]$ using Min-Max normalization [137]. Consequently, a tanh activation function is applied to the output for a faster convergence [65, 137]. To compare with the groundtruth, the predicted values are scaled back to normal ranges.

Training We split the data in each of the two cities into the training and the testing

dataset, where the testing dataset contains the last 28 days of the observation sequences and the rest of the data belong to the training dataset. In so doing, we have 18,624 training samples and 1,344 testing samples for the city of Seattle, and 26,304 training and 1,344 testing samples for New York City. The discrepancy between the numbers of training samples are due to occasional missing data on some days for each of the two cities. Following [137], our training procedure contains two steps. (1) To find a good initialization of our model, We first train our model using 90% of the training data and reserve the rest 10% as validation data. During this step, we apply early-stopping based on the validation loss. (2) After that, we continue to train our model on all the training data for another fixed number of epochs (e.g. 100 epochs). The loss function used in the training process is the Mean Squared Error.

By default, the periodicity and trend interval p and q are set to one day and one week, respectively. The lengths of the dependence sequences are set to $l_c = 3$, $l_p = 1$ and $l_q = 1$.

3.6.2 Baseline Approaches

We choose the following seven methods as the baseline approaches:

- *ZERO*: a naive baseline approach which simply yields predictions of 0s for all tweet count.

- *ARIMA*: Auto Regressive Integrated Moving Average (ARIMA) model is a time series analysis model for understanding the time series data or predicting future points in the series [42].

- *SARIMA*: Seasonal ARIMA, which additionally considers possible seasonal effects.

- *Eyewitness*: Eyewitness [58] uses gradient boosting regressors to train a regression function by considering features such as the time of the day, the day of the week and tweet counts from neighboring regions.

- *ST-ResNet*: ST-ResNet [137] is the currently state-of-the-art method used in spatiotemporal data prediction which is a strong baseline. Different from the proposed method, it uses regular convolution layers instead of convolutional LSTM layers. By default, ST-ResNet uses one residual block, which achieves the best results on our dataset. The effects of stacking multiple residual blocks will be further explored in Section 3.6.4.4.

- *ConvLSTM* $\times$ 3: a baseline approach that simply stacks three layers of ConvLSTM in order to contrast the effectiveness of a residual block over a ConvLSTM layer. It replaces the Residual ConvLSTM block with a ConvLSTM layer in Figure 3.5.

- *ConvLSTM* $\times$ 4: a baseline approach that stacks four layers of ConvLSTM in order to contrast the effectiveness of the skip connection in the residual block. We define this model by simply removing the skip connections in our proposed model.

3.6.3 Evaluation Metric

The results are measured by the Root Mean Square Error (RMSE):

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - X_i)^2} \tag{3.16}$$

where n is number of testing cases, and Y_i and X_i are the prediction and groundtruth values, respectively.

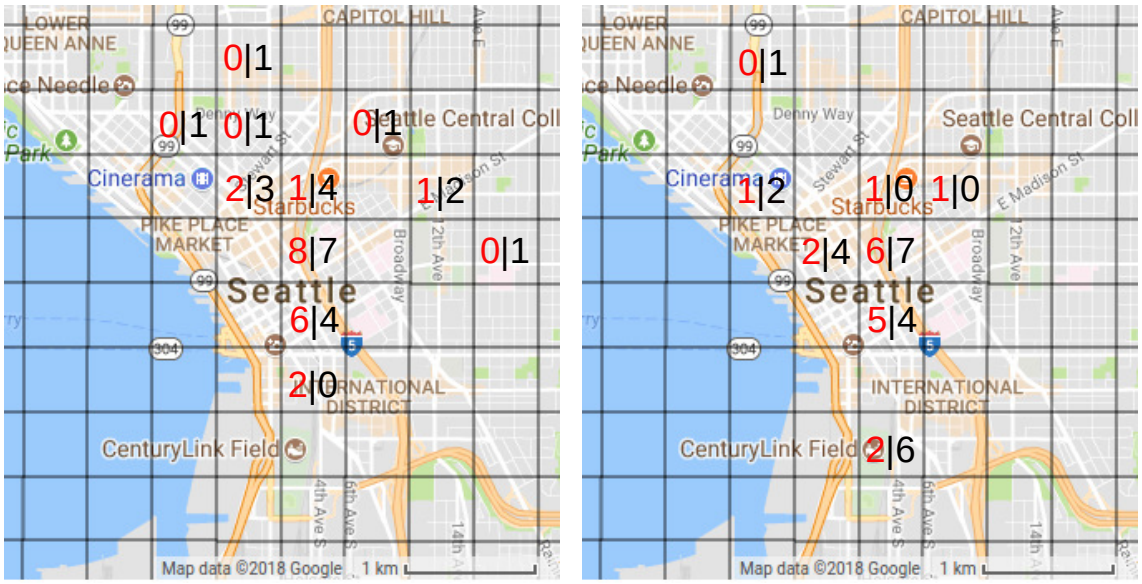
3.6.4 Experimental Results

We start with an illustration of two predication examples, followed by a comparison between our proposed method and the six baselines mentioned in Section 3.6.2. Then we study the effectiveness of temporal dependence sequences and the effect of deeper neural networks.

Figure 3.10 presents the prediction results using our model for the two tweet count distribution examples. The denotation in each grid cell is in the form of "prediction|groundtruth", referring to the prediction *vs.* groundtruth number of tweet count. The numbers in red are predictions. No denotation in a cell means a correct match with the groundtruth. The results show that both of the predictions are generally good matches to the groundtruth by being able to capture the overall distribution of tweets as well as yielding only a slight difference for grid cells that have larger values of the tweet count. The error is mostly caused from predicting empty tweets for grid cells which have only one tweet. Such a situation is relatively arbitrary in the sense that the occurrence of such a tweet can be sporadic, which makes it hard to predict.

3.6.4.1 Compare with Baselines

Table 3.2 shows the results of seven baselines and the proposed method on two cities: Seattle and New York City. Simply generating prediction of 0s (ZERO) for every



(a) (b)

Figure 3.10: (a) Prediction example of tweet count distribution around the Seattle city center at 17:00-17:30 on 2016-07-16. (b) Prediction Example of Tweet Count Distribution around around the Seattle city center at 17:30-18:00 on 2016-07-16. (The denotation in each grid cell is in the form of "prediction | groundtruth", referring to the prediction *vs.* groundtruth number of tweets. The numbers in red are predictions. No denotation in a cell means a correct match with the groundtruth.)

grid cell performs much worse than all other methods. We notice that ST-ResNet outperforms all the other methods except the proposed one, showing its effectiveness. Using ConvLSTM achieves comparative results to ST-ResNet. We believe that this is because of the ability of ConvLSTM to model the spatial and temporal information well. The proposed method outperforms all the baselines and achieves state-of-the-art results. It achieves significantly better accuracies than both ConvLSTM $\times$ 3 and ConvLSTM $\times$ 4, which illustrates the effectiveness of the skip connections. As mentioned in [69], the loss functions of deeper networks are more likely to be chaotic, while adding skip connections can prevent this leading to a more convex loss function which is easier to train.

Method	Seattle	NYC
ZERO	0.6353	1.2054
ARIMA	0.5117	0.5301
SARIMA	0.5242	0.5340
Eyewitness	0.4580	0.5332
ST-ResNet	0.4344	0.5166
ConvLSTM × 3	0.4659	0.5232
ConvLSTM × 4	0.4557	0.5278
Our Model	**0.4164**	**0.4879**

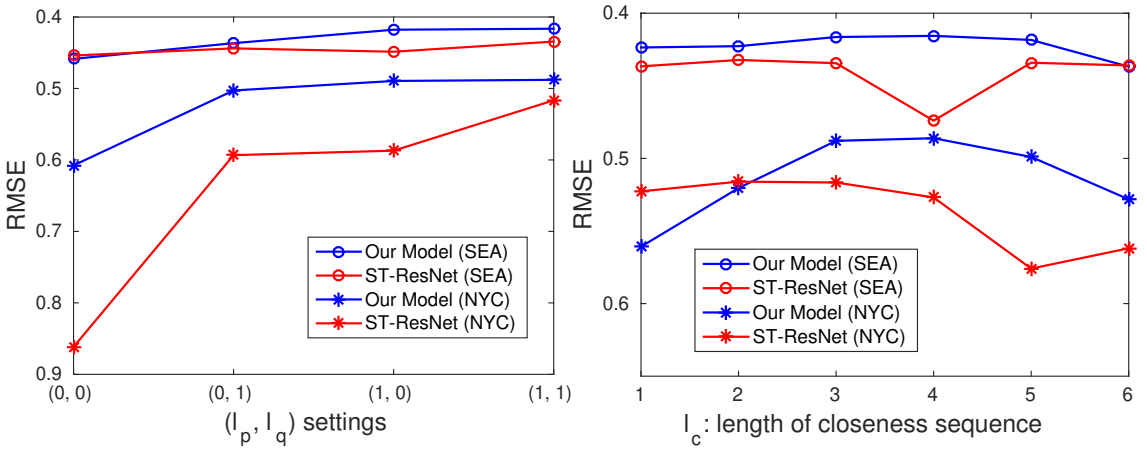
Table 3.2: Comparison results (RMSE) on city of Seattle and NYC.

3.6.4.2 Effects of *period* and *trend* Dependence

We now investigate the performance of our model with and without utilizing *period* and *trend* information. We set the corresponding length variables l_q (l_q) to 0 or 1 to indicate whether the model is configured to use such information. The results are presented in Figure 3.11a. It shows that only using *closeness* information may perform even worse than the baselines and justifies the exploitation of *period* and *trend* dependence sequences. Nevertheless, in this study, we found that longer (> 2) *period* and *trend* dependence sequences do not always yield better accuracy.

3.6.4.3 Effects of Length of *closeness* Dependence Sequences

In this subsection, we study whether a longer *closeness* dependence sequence can help achieve better performance in method ST-ResNet and in our model. The results are illustrated in Figure 3.11b. It can be seen that both models are able to achieve slightly better accuracy when the length begins to increase, but the performance saturates or becomes worse after l_c reaches 4. One possible reason is that the tweets that happened a



(a) (b)

Figure 3.11: (a) Effects of using *period* and *trend* dependence or not. (b) Effects of length of *closeness* sequences. Note that the higher the curve, the smaller the RMSE value.

longer time ago may not provide much information for predicting the tweet at the current time. Meanwhile, our model has higher gains than ST-ResNet because recurrent structure is more powerful in capturing temporal information. Moreover, we notice that ST-ResNet is more sensitive to tweets posted a longer time ago as the performance drops dramatically when $l_c = 4$ for Seattle and $l_c = 5$ for New York City.

3.6.4.4 Effects of Building Deeper Networks

In general, we found no significant gains by stacking more residual ConvLSTM blocks in our method ResConvLSTM. Take the city of Seattle for example, Figure 3.12 illustrates the results of stacking $\{0, 1, 2, 4\}$ residual blocks using RMSE metrics. It shows that two or more layers can not guarantee to achieve better results, although the performance deteriorates if no residual block is used at all. The situation is similar when it comes to stacking more residual convolutional blocks in baseline approach ST-ResNet.

We believe this is due to the following two reasons: (1) As discussed in [69], deeper networks usually have a more chaotic loss function, making them difficult to train. (2) Deeper networks are more likely to suffer from over fitting.

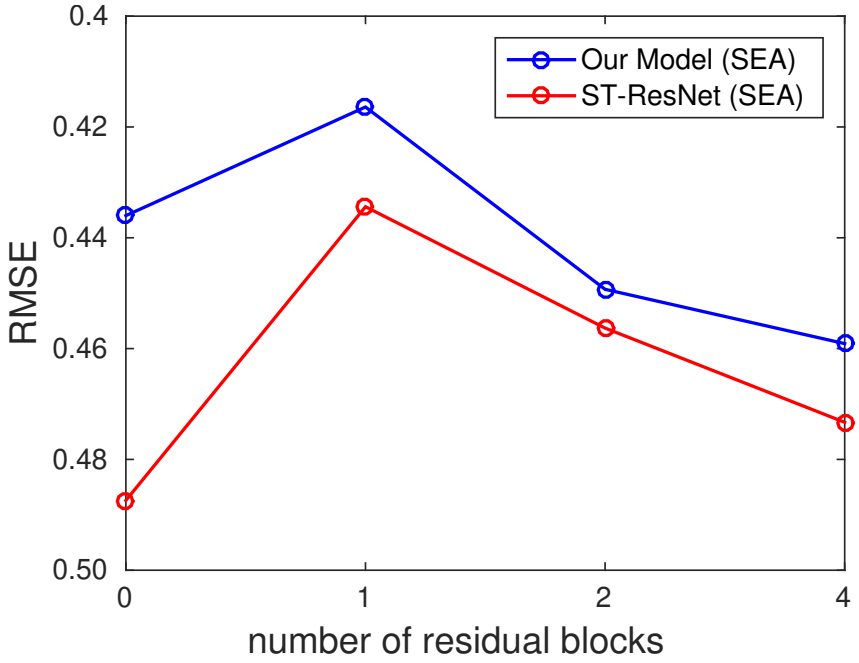


Figure 3.12: Results of stacking more residual blocks in the city of Seattle.

3.7 Evaluation on Local Event Detection

DeLLe is implemented in Python and evaluated on a computer with an Intel Xeon E5 CPU, an Nvidia Quadro P6000 GPU and 64GB memory.

3.7.1 Experimental Settings

3.7.1.1 Datasets

The evaluation is performed on two sets of geotagged tweets collected in Section 3.6.1 for two cities: Seattle, WA (SEA) and New York City (NYC) [116]. Their

geographical regions are two bounding boxes spanning from [47.579784, -122.373135] to [47.633604, -122.293062] in SEA, and from [40.647984, -74.111093] to [40.853945, -73.837472] in NYC as illustrated in Figure 3.9. We take the data from 2017-06-23 to 2017-07-23 for testing and local event detection.

3.7.1.2 Baseline Approaches

We compare with the baseline approaches described below:

- EvenTweet [5] first identifies temporal bursty keywords and spatial local keywords and then clusters them to find local events.
- Eyewitness [58] finds tweet volume spikes in discretized time and space as potential local events by comparing the actual number of tweets with the predicted value using a regression model.
- GeoBurst [134] first generates candidate events by seeking pivot tweets based on geographical and semantic similarities and then ranks them with spatiotemporal burstiness to remove noisy ones.
- TrioVecEvent [131] first learns multimodal embeddings of tweets on the domains of location, time, and text and then uses a Bayesian mixture clustering model to find event candidates.

3.7.1.3 Parameter Settings

We run DeLLe in its batch mode by default and will evaluate its difference from the online processing in Section 3.5. The major parameters in DeLLe are set as follows. For space and time, we set the side length of grid cells Δl = 500m and the length of time interval Δt = 30 minutes (by dividing a natural integral hour into two intervals) since such values provide fine enough resolution for local event detection as well as yield good performance for tweet count prediction [116]. As a result, we have a 12×12 grid map in SEA and 46×46 in NYC. For the moving step length in sliding windows, we set $\Delta s = 5$ minutes, which is long enough for the online processing latency in our system. In the seeker module, we set the length of *closeness*, *period* and *trend* to $l_c = 3$, $l_p = 1$ and $l_q = 1$ as in Section 3.6 [116] because such a setting achieves the best prediction accuracy. We set the threshold for determining the unusualness of a grid cell $k_{\Delta \mathbf{E}'_T} = 3$, a commonly used value for anomaly detection. As for the PageRank procedure to calculate topical coherence in the ranker module, we use the default damping factor 0.8 and run 20 iterations in all cases. After tuning, in calculating content similarity between grid cells for expansion, we set the number of frequent tokens k_{cs} to 5 and the content similarity threshold ε_{cs} to 0.7. We set $k = 5$ as the number of the most influential tweets used for calculating the topical coherence as well as the number of tweets used to summarize a local event [134, 58]. In each time interval, we select at most $K = 5$ unusual grid cells as the local event candidates. Because not every time interval does necessarily have K local events happening, we apply a simple heuristic for suppressing the negative candidates. It removes grid cells having too few users (i.e., less than 5) or having a topical coherence score less than

0.8, which is a suggested lower bound for tweet clustering using Tweet2Vec [28, 109]. For fairness, we also similarly filter out the event candidates with less than 5 users for the baseline approaches as well in the evaluation.

EvenTweet takes the same space partition as in DeLLe and similarly selects the top K local event candidates. Since each event in EvenTweet is a cluster of keywords instead of tweets, we use the implementation in [134] to retrieve the top k representative tweets.

Eyewitness exhaustively sweeps through a set of different space and time discretization and is unsuitable for processing live tweet streams. We ease its settings by using the same space and time discretizations in the batch mode of DeLLe. To select the top K local event candidates, we rank them by the prediction error divided by the standard deviation of the error of its regression function, which has shown to be an important feature in classifying events to be postivie or negative [58]. After that, each local event is represented by by choosing $k = 5$ tweets with the highest frequency words.

For GeoBurst and TrioVecEvent, we adopt their default parameter settings and implementations in [134] and [131], respectively. Since both methods require an input time window to query the occurrence of local events, we set it as a list of disjoint Δt-size windows like the time discretization in DeLLe's batch mode and choose the top K candidates for comparison. Note that TrioVecEvent also classifies an event candidate to be true or false, we therefore use the spatial deviation (i.e., lat/long deviations, which are the two most important features in their classifier) to rank local events.

(a) Seattle, WA (b) New York City

Figure 3.13: Examples of true local events. The left is in Seattle, WA, and the right is in New York City, respectively. (a) A baseball game of Yankees-Mariners at Safeco Field (2017-07-20 7:00 PM). (b) NYC Pride March traversing down Fifth Avenue (2017-06-25 10:30 AM).

(a) Seatlle, WA (b) New York City

Figure 3.14: Examples of false local events. The left is in Seattle, WA, and the right is in New York City, respectively. (a) People talking about food near the Space Needle (2017-07-22 4:30 PM). (b) People waiting for 4th of July fireworks at East River (2017-07-04 5:00 PM).

3.7.2 Illustrative Cases

We select several positive and negative examples of local event detection and present them in Figure 3.13 and Figure 3.14, respectively. Each example is described by 5 representative tweets with locations plotted as red circles in the accompanying maps. Ahead of each tweet is its publisher's username. Note that multiple tweets may reside at the same location causing overlapping and dark red circles.

Figure 3.13 illustrates two positive local events reported in DeLLe. Figure 3.13a is about a baseball game between the Yankees and the Mariners held at the Safeco Field in Seattle. Figure 3.13b is about NYC Pride March 2017 traversing southward down

Fifth Avenue in New York City. Those two events are very demonstrative as examples of local events because they have exhibited the necessary properties DeLLe wants to capture: spatiotemporal unusualness regarding the number of tweets at a local place and topical coherence regarding the content of aggregated tweets. The tweets selected to describe the events are also representative to convey the necessary information. It is worth mentioning that the tweets in Figure 3.13a fall closely to the common border of two neighboring grid cells. The expander module in DeLLe effectively captures this case by connecting spatiotemporally adjacent grid cells sharing similar content. These two examples also appeared in the baseline approaches.

Figure 3.14 presents two cases of negative local events in EvenTweet and Eyewitness, respectively. Figure 3.14a refers to an activity about people talking food near the Space Needle in Seattle, WA. EvenTweet reported this activity as a local event since it finds some spatiotemporal bursty keywords like "Bite". This is because, when the day comes around dinner time, that area seems to be a popular place for people to eat and thereby aggregates tweets with similar keywords about food. Likewise, GeoBurst also falsely reported a related geo-topic cluster because it groups together tweets mentioning similar keywords on the topic and located geographically closely. Although such an activity may attract enough tweets at a high rate at certain time (e.g., dinner time in the example), it usually follows a periodic daily pattern and does not reflect any unusual event. Neither Eyewitness or our method DeLLe reported this activity because both of them take routine patterns into consideration. Similarly, TrioVecEvent classified it as a non-local event too. This is because its multimodal embedding model also addresses the effect of time in tweets and unveils typical words in different regions and time periods.

Figure 3.14b is an example of negative local event reported in Eyewitness. It is about people waiting for the 4th of July fireworks show at East River Ferry Dock in New York City. This is more like a national event in the United States because fireworks show on Independence Day may happen at different places in a nationwide scale. When it comes close to the evening, one may expect that tweets about fireworks suddenly increase all over the country. Both GeoBurst and TrioVecEvent reported this nationwide event too. This is because such an event is also geographically compact and more importantly, semantically coherent. In contrast, we do not find the occurring grid cell of this event ranked in the top unusual grid cell candidates in our method DeLLe. There are three reasons behind this. First, the likelihood of unusualness in this grid cell was not high considering that other places were experiencing similar burstiness in tweet volume. Second, the spatial burstiness was not strong either because similar keywords were being used everywhere. Third, the topical coherence in this grid cell deteriorated due to the presence of lots of other tweets like "@511NY Cleared: Incident on #ServiceBus at Midtown". EvenTweet did not report this event either because the keyword like "fireworks" and "july4th" appeared adequately in other regions too and thus was considered not to be local to this event's occurring site.

3.7.3 Quantitative Analysis

3.7.3.1 Effectiveness

We first evaluate the different local event detection methods using *precision*, *recall* and *f-score*. For precision, we recruited 3 volunteers to individually judge the detected

Method	Seattle, WA				NYC			
	#	**P**	**R**	**F**	#	**P**	**R**	**F**
EvenTweet	354	0.391	0.390	0.390	1665	0.146	0.131	0.138
Eyewitness	273	0.769	0.593	0.670	1204	0.614	0.398	0.483
GeoBurst	354	0.517	0.517	0.517	1665	0.203	0.182	0.192
TrioVecEvent	240	0.858	0.582	0.694	1214	0.704	0.461	0.557
DeLLe	269	**0.862**	**0.655**	**0.745**	1128	**0.741**	**0.450**	**0.560**

Table 3.3: Comparison results using Precision, Recall and F-Score.

events and collect the results using the strategy of majority votes. The instructions given to the judges are summarized as follows:

Each candidate of local event has a set of 5 tweets, accompanied by a mini map showing their location. Local events may come from the city of Seattle, WA or New York City. Please read the tweets and answer if they are talking about the same local event. Local events can be about traffic accidents, sports games, parade, protests, gatherings and crimes etc. National or global events such as national holidays are not considered to be local. Daily life activities such as buying coffee in the morning and going out to eat dinner in the evening are not local events either. In addition, if the presented tweets are about totally different topics, then it is not an local event. If you can't determine whether the candidate is about a local event, label it as negative candidate too.

In lack of groundtruth on the set of events happened in the real world, we build a pseudo groundtruth by assembling a set of distinct true positive local events reported in different methods to calculate the recall and f-score. The comparison results are listed in Table 3.3. It shows that DeLLe outperforms baseline approaches in most cases. In particular, a significant improvement is observed over EvenTweet and GeoBurst. DeLLe also achieves comparatively better results to Eyewitness, showing the effectiveness of its un-

usualness detection and consideration of topical coherence. We notice that TRIOVECEVENT outperforms all other methods except for the proposed one, showing its effectiveness of multimodal embedding of location, time and text information in tweets.

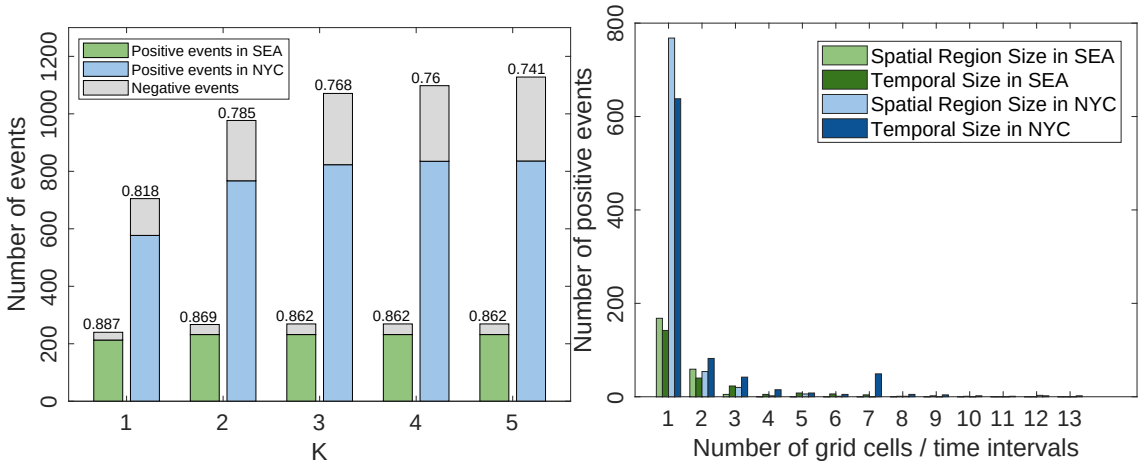


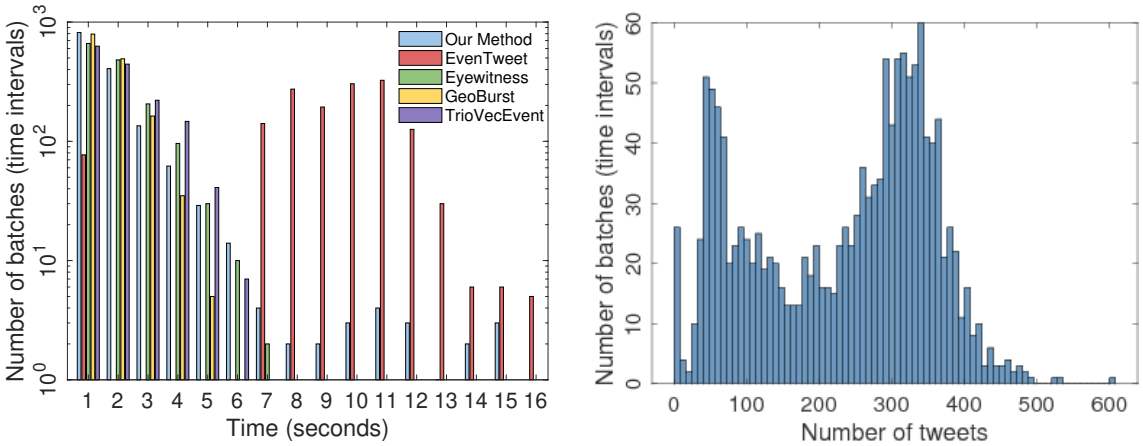
Figure 3.15: (a) Precision with different K values. (b) Temporal span and spatial region size of positive local events in DELLE.

To evaluate the sensitivity of K, Figure 3.15a illustrates the number of positive local events out of the total detected ones in DELLE when K varies. The decimal number above each bar represents the precision. In general, the precision decreases as K increases because a larger K likely outputs more negative local events, even thought it may also give more positive ones. We notice that the precision and the number of positive local events nearly maintain the same in SEA after $K = 2$ and NYC after $K = 3$.

Figure 3.15b plots the distributions of positive local events in DELLE regarding the temporal span (i.e., number of time intervals) and spatial region size (i.e., number of grid cells). The results show that the majority of the events fall within one single time interval and one grid cell. This validates our settings in the time and space discretization.

3.7.3.2 Efficiency

To investigate the efficiency, after each time interval ends, we record the time spent in processing the tweets aggregated during that interval for the 5 different methods. The results are reported on the NYC dataset as it contains relatively more tweets. The total number of time intervals is 1,488.



(a) (b)

Figure 3.16: Distributions on the numbers of time intervals over their processing times in (a), and over their number of tweets in (b).

Figure 3.16a presents the distributions of time intervals over their processing time in different methods. To have an idea of the number of tweets in each time interval, we plot its histogram in Figure 3.16b. Among the three methods that exploit space partition strategy (i.e., EvenTweet, Eyewitness and DeLLe), Eyewitness in general is the most efficient method because it does not require sophisticated tweet text processing except when summarizing its detected event. DeLLe has achieved similar efficiency with Eyewitness in majority cases, even though a few of the cases sometimes take as long as 15 seconds. The major overhead lies in computing topical coherence in the ranker module as well as

content similarity in the expander module. These steps, however, are only necessary when an unusual grid cell appears. Simply running the seeker module to identity potential local event candidates is very fast and takes 0.06 seconds on the average. EvenTweet is less efficient than the other methods due to its calculation of spatial entropy to identify spatially local keywords and then performing clustering. Although GeoBurst and TrioVecEvent have excellent efficiency as well, their implementations [134, 131] require certain preprocessing steps on the tweets like extracting keywords and keyword co-occurrence relation, which would take considerably more time.

3.7.4 Online Modifications

The batch mode of DeLLe divides the temporal dimension into disjoint time intervals, i.e., $\{\cdots[t-2\Delta t, t-\Delta t), [t-\Delta t, t)\}$. In practice, some local events may fall across these interval boundaries. We made online modifications in Section 3.5 for handling this issue. In this section, we investigate the effectiveness and efficiency of these modifications on the NYC dataset.

Effectiveness is evaluated by example how many local events detected in the batch mode are also detected in the online processing and meanwhile how many local events the batch mode misses. For comparison, we here claim that two local event candidates refer to the same occurrence if i) their content similarity is greater than 0.7; ii) their time centroids (i.e., the average publish time of the tweets) are within $2\Delta t$ (i.e., one hour); iii) they come from the same grid cell. Figure 3.17a shows the Venn diagram of different sets of local event candidates generated in batch and online mode. For comparison, we also

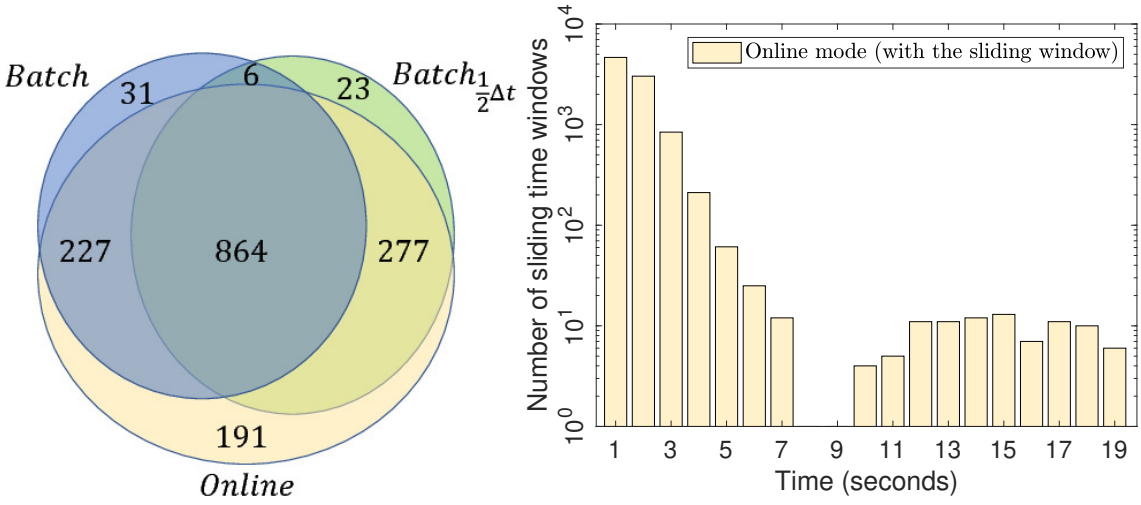


Figure 3.17: (a) Venn diagram on the sets of events in batch mode and online mode. (b) Distribution of time spent in online mode.

include one variant of the batch mode, *called* $Batch_{\frac{1}{2}\Delta t}$, which offsets the disjoint time intervals by $\frac{1}{2}\Delta t$, i.e, $\{\cdots[t-\frac{5}{2}\Delta t, t-\frac{3}{2}\Delta t), [t-\frac{3}{2}\Delta t, t-\frac{1}{2}\Delta t)\}$. The Venn diagram shows that the online mode, with help of the flexible sliding time window, has chances to screen different interval settings on the temporal dimension and indeed discovers more local events. We also found that $Batch_{\frac{1}{2}\Delta t}$ has slightly more events than the original batch mode (i.e., dividing an integral hour into two time intervals). This is reasonable in the sense that although the latter time division fits more with the habits of people for planning events, people are likely to post tweets before an event starts when they have chances.

For evaluating the efficiency, we similarly record the processing time for each step in the sliding moving window. The results are presented in Figure 3.17b. Generally, the online processing shows a similar trend with the batch mode except for more cases falling after 10 seconds. After analyzing, we found that the major overhead lies in the frequent invocation of the expansion procedure to connect temporally adjacent cells that

are semantically similar. Even so, the worst case takes less than 20 seconds in general and is likely acceptable for many applications.

3.8 Conclusions

In this chapter, we presented DeLLe for detecting latest local events in geotagged tweet streams. In essence, DeLLe first identifies spatiotemporal unusualness using a novel prediction-based anomaly detection approach, and subsequently ranks them to identify potential local events, by addressing both spatiotemporal burstiness and topical coherence. Afterwards, DeLLe monitors the impact range for an ongoing local event in space and time by tracking its movement with content similarity, and meanwhile selects influential tweets for summarization. The evaluation results on two selected cities show that DeLLe outperforms competitive baselines in most cases, showing the effectiveness of the proposed method.

The human evaluation yields a groundtruth of local events, and therefore enables the exploration of learning to classify spatiotemporal unusualness into true/false local events using features like burstiness and topical coherence. We leave this for our future work.

Chapter 4: Enhancing Local Live Tweet Stream to Detect News

4.1 Introduction

The popularity of Twitter arises from its capability of letting users promptly and conveniently contribute tweets on a wide variety of subjects such as news, stories, ideas, and opinions. As a result, with people discussing what is happening outside in the real world by posting tweets, an invaluable amount of information on the real world news is hidden in Twitter. Therefore, many researchers have devoted remarkable efforts to discover this knowledge. For example, TwitterStand [102] is a news tweet processing system that aggregates tweets from a sparsely sampled tweet source to detect news. This is not a problem for major news stories since there are more than enough tweets to capture them.

However this approach is too brute-force for smaller-scale local news where every single tweet matters because such types of news may only span a very limited number of tweets. Figure 4.1 shows a news story about the "Westborough Education Foundation" that happened at around 6:30 PM on Oct 24, 2016 at Westborough, MA. We only found 6 tweets (8 if retweets are included) about this news by the time we captured the screenshot, and none of them is geotagged, i.e., containing a pair of geographical lat/lon coordinate values. No access to full tweets in Twitter makes data sparsity pervasive in Twitter's

publicly accessible tweets, and further compromises the possibility of collecting all 6 tweets about this news. The challenge in capturing such news lies in being able to find these tweets, cluster them into a news story, and then subsequently displaying it on a map.

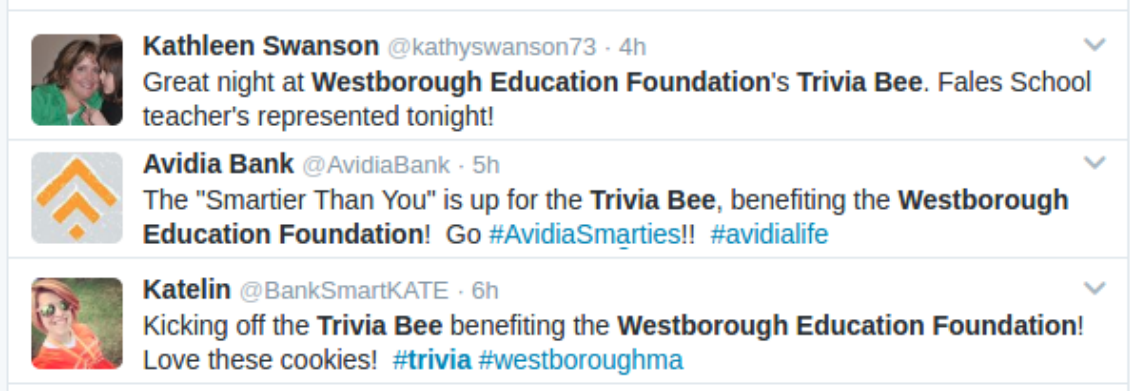


Figure 4.1: A local news in Westborough, MA on Oct 24th, 2016.

In this chapter, we are interested in detecting news (a set of tweets) that are being discussed by local people from a given place (e.g., Boston city), and meanwhile emphasizing on finding local news. The term "local news" refers to a news event that happens at or is of great interest to the given place. For instance, the news story in Figure 4.1 may only be of interest to the local community and not much further beyond. Local news can sometimes escalate to be of national/international interest such as when it is dramatic (e.g., Boston Marathon bombing in May 2013). We want to capture both these types. Other national and international stories that are discussed by local people (e.g., a presidential election) are also in by providing a local perspective to larger news stories. Our focus is primarily the former two classes of stories, and later in our experiments we evaluate how well we do with and without considering these national and international news stories.

Identifying the news stories that are of great interest to a place requires a combina-

tion of approaches. It requires first finding users that reside and tweet about our place of interest. To find such users, we implement an efficient online social network-based Twitter user geotagging approach, which is to approximate the location of a Twitter user by examining the publicly-known locations of his social friends (neighbors). The publicly-known location, termed the *profile-location*, is provided in a Twitter user's profile, but is only available for around 20% (in our case, 32%) of Twitter users [24]. This makes the procedure of geotagging Twitter users indispensable in our system. With the help of this scheme and its efficiency, our system, Firefly, keeps trying to find as many as possible active Twitter users from a given area and putting their posting statuses (tweets) to a local live tweet stream to largely increase its number of local tweets.

Next, there is a larger problem of clustering these local tweets so that news can be captured. For example, some features like bursty words [60, 84] or TF-IDF [102, 114] that are commonly used to group tweets together might not work well with small local news because such news span over a very limited number of tweets, and thus words in them hardly bring about burstiness or yield distinguishing TF-IDF scores. Another category of methods that only exploit geotagged tweets such as [58, 134] would simply miss the news example in Figure 4.1 because few of its tweets are geotagged.

In this chapter, we utilize an idea of "locality-aware keywords" to capture the changes in word-usage patterns caused by a news of limited local interest from the perspective of individual people. Essentially, the locality-aware keywords in each tweet are a set of words that are used only recently by this tweet's publisher and also at the same time only appear in a limited number of other Twitter users' tweets. Such locality-aware keywords correspond to the aspects of a local news being "novel" as its nature of being new,

as well as having a small spread span among Twitter users. Take the one in Figure 4.1 for example, “Westborough”, “Education”, “Foundation”, “Trivia” and “Bee” are considered as locality-aware because they are new words used by this set of people.

To capture news from the enhanced local live tweet stream, we keep identifying and updating locality-aware keywords from tweets that are in the latest 6-hour sliding time window (The choice of 6-hour window is in recognition that television media usually has four times of locally-oriented news broadcast in one day and thus is an appropriate lifetime of local news), and group tweets together that share at least a number of locality-aware keywords to form news clusters. Finally, in our system’s UI, a Twitter timeline is created to post the news we detect from an area in real time. We also estimate the geographic focus of detected news (tweets clusters) to display them on maps.

The main contribution of this chapter is summarized as follows:

- We implement an efficient online Twitter user geotagging procedure on Apache Spark, which takes less than 3 seconds to geotag Twitter users appearing in 1000 tweets. Such efficiency is essential to maintaining the liveness of the enhanced local tweet stream and furthermore the timeliness in news detection.

- Our enhanced local live tweet stream easily covers up a typical metropolitan area. For example, in Boston, we are tracking 176K Twitter users, which is considered sufficient since Boston has a population of 646K[1] and that one-fifth of the USA population are active Twitter users[2].

- The design of locality-aware keywords emphasizes the word usage characteristics

[1] http://www.census.gov/popest/about/terms.html
[2] https://www.statista.com/statistics/274564/monthly-active-twitter-users-in-the-united-states/

of small, local news from the view of Twitter users who are discussing them (e.g., only a small number of people talk about them and they use words they didn't use before).

- We evaluate our system against a set of representative local news agencies as well as a few baseline approaches. The results show that we achieve the highest news coverage and at the same time, outperform the baseline approaches. More importantly, our method detects hundreds of more local news in comparison with the methods that solely utilize the existing Twitter's publicly available tweet stream.

The rest of this chapter is organized as follows. Section 4.2 summarizes the related work. Section 4.3 details the design and implementation of our system. Section 4.4 describes the experimental evaluation of our methods. Section 4.5 contains concluding remarks as well as directions for future work.

4.2 Related Work

There is a large body of related work that deals with extracting useful patterns (e.g., news, events) from social media, Twitter in particular. Two recent surveys Atefeh and Khreich [8], and Abdelhaq [2] provide an excellent description of different techniques. We review some of the related work that deals specifically with the problem of detecting local events. There are two broad categories of methods for taking location into consideration when performing detection tasks, namely: *location-anchored* and *event-anchored*. The essential difference is whether event or location is the primary clustering key. For example, event-anchored methods first detect an event and then determine its location,

while location-anchored methods examine if an event happens at a certain location.

Location-Anchored Methods: Among the location-anchored methods are two popular approaches: *model dimension extension* and *geographical space tessellation*. Model dimension extension treats geographical information as an additional variable to the existing models. For example, in calculating similarity between documents while performing a clustering algorithm, geographical distance between tweets can be incorporated in the clustering algorithm [70] to form potential events [134, 131, 112, 10]. Hong et al. [44] and Zhou and Chen [140] and Wei et al. [117] treat geographical regions as latent variables in their generative topic model.

Geographical space tessellation fills the map with small, non-overlapping cells. The motivation here is that local news or events, which usually have an limited geographical area impact, should fall in the same or nearby cell(s). *Grid* tessellation is the simplest yet most commonly used way of subdividing the geographical space into small equal-sized cells [104, 5, 50, 113, 49]. In reality however, the geographical distribution of social media documents is not homogeneous, frequently requiring the consideration of adjacent cells in the analysis. To alleviate this issue, a few strategies are proposed including resizing the cells, connecting nearby cells if they share similar features, or utilizing an adaptive hierarchical tessellation structure [51]. For example, Krumm and Horvitz [58] et al. discretize the space with a hierarchical triangular mesh. Magdy et al. [81, 80] describe a system called Mercury for querying top-k spatio-temporal queries on microblogs in real-time using a pyramid structure.

After tessellation, the social media documents or features are aggregated into small cells according to their inferred geographical information. Next, an intuitive way to de-

tect the existence of any anomaly at a specific location is to count aggregated documents or other feature entities like keywords to see if their number exceeds a certain threshold. Counting, however, is easily plagued by distribution heterogeneity both temporally and spatially. Therefore, various anomaly detection techniques have been explored. For example, Xu et al. [122] employ a probabilistic model that recovers spatio-temporal signals using a Poisson point process estimation to deal with sample bias and data sparsity problems. Others exploit the usages of a discrepancy paradigm which compares between previous data (to build up a baseline) and the newly observed data [58, 5, 62, 40].

Nevertheless, such methods depend heavily on the availability of social media documents containing geographical information. Such geographical information, however, is very rare in Twitter, with geotagged tweets accounting for less than 1% [113, 12, 76]. Some works have proposed to estimate a geographical location for a non-geotagged tweet. The intuitive approach towards this problem is to geotag nominal locations (place names) embedded in the content of a microblog to get its possible longitude/latitude coordinates by aligning against existing gazetteer databases or services, e.g. GeoNames[3] [2, 113, 110, 98]. While another set of works try to assign a geographical location to a non-geotagged tweet by its poster's location [12, 83, 93] , which might be initially estimated through a social network based procedure [24, 71, 123, 27, 99] or tweets content-based methods [19, 20, 82, 35, 25, 68].

Event-anchored Methods: This class of methods, after identifying events, leverages an additional step of spatial analysis to determine the locations where they are happening. For example, TwitterStand [102], after clustering tweets to identify events,

[3] http://geonames.org/

estimates each news cluster's geographical focus by making use of both geographical information in the content of the tweet and by the source location of the users. This geographic focus is computed as a whole by ranking the geographic locations in the cluster. One basic measure of relevance used in their ranking is the frequency of occurrence of each geographic location in the cluster. The reasoning is that if a geographic location is important to the event at hand, the it would be mentioned in many tweets and linked articles belonging to the cluster. In addition, they also give a higher relevance score to groups of locations that are mutually proximate by considering that geographic locations that are nearby to each other lend evidence to each other. To infer and track the location of detected earthquake or typhoon events, Sakaki et al. [100] resort to Kalman filtering and particle filtering by treating each Twitter user as a sensor.

Even though all event-related documents are exploited (not just the ones with location information) in event-anchored methods, their data sources still suffer from sparsity to detect small, local events. For example, TwitterStand's data source, which then claimed to sample around 10% of all tweets but now only 1%, is still too small for small-scale events that might only span $3 \sim 5$ tweets in total.

Therefore, realizing it is the local data sparsity that undermines the opportunities for researchers to discover small-scale events in Twitter, our system proposes to enhance the public local live tweet stream for an area by i) identifying as many Twitter users as possible that are from that area and then ii) tracking the tweets that they publish in real-time. Weng and Lee [118] similarly track a number of users in Singapore to detect news but only at a small scale, i.e., 1K Twitter users. In contrast, we identify and track 176K users in Boston. Our work is also different from Albakour et al. [6], which directly

chooses several areas in London to collect tweet data, and tries to detect events for each of these areas separately. Their method doesn't solve the problem of local data sparsity by using Twitter's Streaming API, i.e., statuses/filter with parameter "locations", in our experiment, is still very sparse and thus makes a very limited contribution to local news detection.

4.3 System

In this section, we present the design and implementation of our event detection system, Firefly, as illustrated in Figure 4.2. Including the User Interface, Firefly consists of 5 major modules, which are described below sequentially.

4.3.1 Online Twitter User Geotagging via Spark

The goal of this module is to keep estimating the geographical locations for more Twitter users, and thus to maintain a large pool of geotagged Twitter users. In so doing, for a given geographical area like the Boston Metropolitan area, our system can easily retrieve a large body of Twitter users in it. Tracking tweets posted by these users significantly enhances our local live tweet stream.

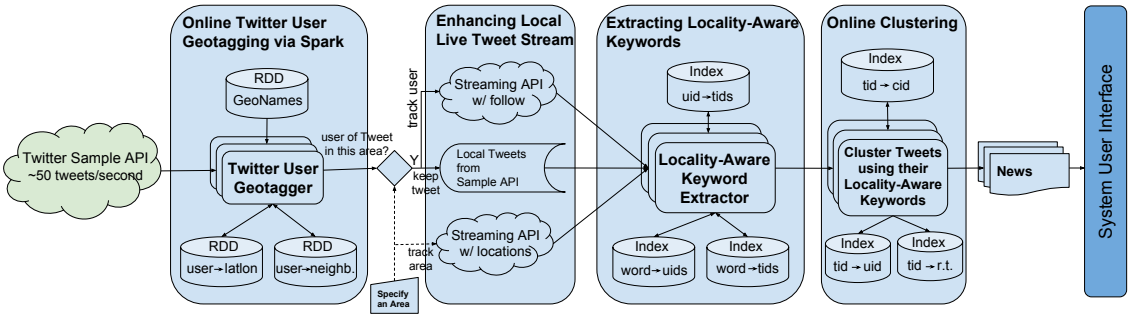


Figure 4.2: System architecture of Firefly.

The motivation behind geotagging Twitter users is that the profile-location information for specifying where a Twitter user comes from is only sparsely available in public data. Therefore, inspired by studies [**Takhteyev2012Geography**, 88] that online social friendships are often formed over short geographic distances, a social network-based Twitter user geotagging method is proposed in [24], which approximates a user's location by examining the publicly-known locations of his online friends (neighbors). This method is reported to have the state-of-the-art city-level accuracy when geotagging a large-scale body of Twitter users and, more importantly, doesn't require sophisticated natural language processing in comparison with tweets content-based methods [19, 20, 82, 35, 25], thus making it more suitable for online geotagging.

To be specific, the social network-based geotagging problem is addressed from the point of view of solving an optimization problem, i.e., inferring user locations is solved by finding

$$min_{\boldsymbol{f}} \|\nabla \boldsymbol{f}\| \text{ s.t. } f_i = l_i, \ \forall i \in L \tag{4.1}$$

where $\boldsymbol{f} = (f_1, f_2, f_3 ... f_n)$ represents location estimation for each user $1...n$, and L denotes the set of users who opt to make their locations l_i public. The total variation is formulated as $\|\nabla \boldsymbol{f}\| = \sum_{ij} w_{ij} * d(f_i, f_j)$, where $d(\cdot, \cdot)$ measures geographical distance.and w_{ij} weighs the friendship between user i to user j, which essentially reflects how many times user i reciprocally interacts with j such as retweeting, mentioning etc. Note that, an edge between i and j in the graph is bidirectional and only formed if both i and j have actively initiated at least one interaction with each other, and we use reciprocal neighbors

or friends to term such edges.

The above minimization problem could be solved by calculating, for each user, the *L1-multivariate median* from his reciprocal neighbors' locations. The value of *L1-multivariate median* [111], which acts as a user's estimated (geotagged) location and is denoted by l^{L1mm}, essentially finds a point that minimizes the sum of its distances to the users' reciprocal neighbors. For a user j, its *L1-multivariate median* l_j^{L1mm} is mathematically defined as,

$$l_j^{L1mm} = \underset{l}{\operatorname{argmin}} \sum_{l_i \in L_j} w_{i,j} * d(l, l_i) \tag{4.2}$$

where, L_j contains the locations of j's reciprocal neighbors. In the implementation, Equation 4.2 can be solved through a coordinate descent procedure.

Upon completing the calculation of location estimate, for a user j, how far l_j^{L1mm} deviates from his reciprocal neighbors determines whether he accepts l_j^{L1mm}. This deviation, called *Geographical Dispersion*, is defined as,

$$GD(L_j) = median_i \; w_{i,j} * d(l_j^{L1mm}, \; l_i) \quad \text{s.t. } l_i \in L_j \tag{4.3}$$

For example, user j will accept his estimated location if $GD(L_j)$ is less than a given threshold, γ. In our experiments, we set $\gamma = 100$ km, which is suggested as a suitable trade-off between geotagging coverage and accuracy for the city-level scenarios [24].

One drawback of [24] lies in indiscriminately utilizing all available location information from reciprocal friends to calculate a candidate location estimation in Equation 4.2, while some of them might be noisy points as discussed in [123]. For example,

as illustrated in Figure 4.3 (where each circle represents a reciprocal friend and the number in each circle denotes the weight to that friend), a user from Boston has 9 reciprocal friends with available location information, 4 of them (red circles) are relatively far away from Boston and can be seen as noisy points or outliers because incorporating them into Equation 4.2 is likely to yield a location estimation that does not satisfy the geographical dispersion constraints γ, and thereby fails to geotag this Twitter user.

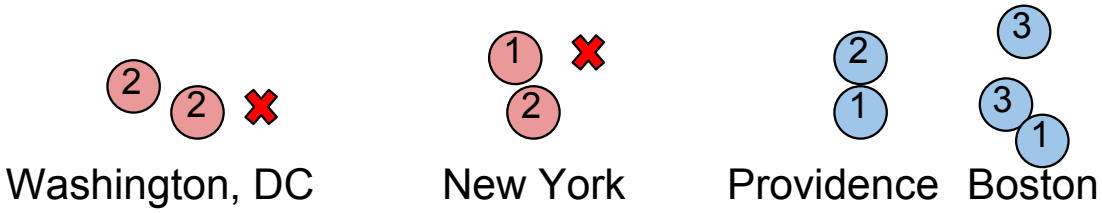


Figure 4.3: An illustration of outliers in the locations of reciprocal friends.

Inspired by the observation in [123] that the location of a friend is usually more reliable if a user has multiple friends from that or nearby location, we propose a single-linkage-clustering based outlier removal procedure to get rid of potential noisy points. As presented in Algorithm 2, this procedure works as follows. Take the locations of a user j's reciprocal neighbors, L_j, as the input, we first perform the Single Linkage Clustering with geographical dispersion γ being the distance threshold. During the clustering, two location points in L_j that are within γ are grouped into the same cluster; and two clusters are merged if a pair of points from each of them are within γ. Next, we select the cluster with maximum sum of weights and use it as new L_j in Equation 4.2 to calculate the location estimation.

Another improvement over [24] is a minimum size constraint for L'_j because too few location information might be considered as weak evidence [27]. In other words, we

Algorithm 2: Outlier Removal

Input: The locations of user j's reciprocal neighbors –L_j; distance threshold –γ; cluster size threshold–λ

Output: A list of locations after removing outliers – L'_j

1: A set of clusters $C = \{C_1, C_2, C_3...\} \longleftarrow$ Single Linkage Clustering on L_j with γ;
2: $L'_j = \underset{C_k \in C}{\text{argmax}} \sum_{l_i \in C_k} w_{i,j}$
3: return L'_j if $|L'_j| \geq \lambda$; else $\emptyset$

refuse to calculate l_j^{L1mm} for user j if $|L'_j|$ is less than a given threshold λ. The experimental results show that such a constraint for λ might effectively improve the accuracy of geotagging in the sparse social networks where users have only a few reciprocal friends, especially the ones with valid locations.

Publicly-Known Locations of Twitter Users In Twitter, there are two sources to know a user's location: profile-location or the GPS coordinates embedded in his tweets. The profile-location is often in the form of place names like "College Park, MD" and can be aligned with databases like GeoNames to decode its geographical latitude/longitude coordinates. In order to assign a unique pair of latitude/longitude coordinates, for a user having multiple GPS points available in his tweets, we compute the *L1-multivariate median* for these points and similarly check the geographical dispersion to decide whether to use this median or not. At last, for a Twitter user who has a valid profile-location as well as a valid *L1-multivariate median* calculated from his tweets, we opt to use his profile-location if this location is within γ of the median; otherwise, his two sources of location information seem to be conflicting with each other and thus wouldn't be utilized. Algorithm 3 outlines our online Twitter user geotagging procedure, which utilizes a streaming computing platform Spark Stream by maintaining 4 RDD variables [127, 26, 128, 129]. Resilient Distributed Dataset (RDD), is a distributed memory abstraction which gives

Spark the ability to perform fast in-memory map-reduce operations. IndexedRDD extends *key-value* RDD by enforcing key uniqueness and pre-indexing the entries for efficient look-up operations. In practice, RDD could be seen as a table in the database. The IndexedRDD variable for GeoNames, location→latlon, is to align the profile-location, e.g., "Boston, MA", to decode its latitude/longitude coordinates, e.g., [42.3584, -71.0598]. The RDD variable, location→user keeps a reversed index from a user to his profile-location to perform join operation in Spark. The RDD variable, user→twGPS, stores for each user, the GPS coordinates embedded in his tweets. The RDD variable, user→neighb., stores the neighborships between users. Finally, the IndexedRDD variable, user→latlon, caches the geotagged user to retrieve users in a given area.

To quickly start our online geotagging procedure, i.e., fill in the RDD variables, we boost our algorithm with one year of tweets data collected from the Twitter Sample API statuses/sample. We discretize this live tweet stream into 23-second intervals using DStream in Spark to perform the online Twitter user geotag. For an incoming user, we first look-up his geographical coordinates in user→latlon; if this fails, then we try to align his profile-location (if provided) to GeoNames; otherwise, we retrieve a list of his reciprocal neighbors' locations to estimate his location.

4.3.2 Enhancing Local Live Tweet Stream

Given a geographical area, this module tries to collect as many tweets as possible from three sources: two of Twitter's statuses/filter Streaming API – "follow" and "loca-

Algorithm 3: Online Twitter User Geotagging via Spark

Input: Twitter's Public Live Tweet Stream – G; 1 year of tweets collected from Twitter Sample API – T

Output: Geotagged Twitter Users

1: **Boosting Phase**:

a. Load location→latlon from GeoNames; and extract location→user, user→neighb., and user→twGPS in T;

b. user→latlon ⟵ location→user join location→latlon;

c. Update user→latlon with users whose lat/lon can be calculated upon user→twGPS using Equation 4.2 and 4.3.

2: **Online Geotagging**:

a. Init a Spark DStream D in G w/ a 23s time window;

b. Update user→neighb. and user→twGPS with D;

c. **for each** user u in D who is not in user→latlon and fails to align profile-location in location→latlon and fails to calculate a lat/lon in user→twGPS **then do**

i). get u's reciprocal neighbors' coordinates L_u by joining u, user→neighb. and user→latlon;

ii). $L'_u \leftarrow$ Outlier-Removal(L_u)

iii). calculate l_u^{L1mm} by L'_u **if** $|L'_u| \geq \lambda$;

iv). u accepts l_u^{L1mm} **if** $GD(L_u) \leq \gamma$;

end for

tions"[4], and tweets filtered from another Twitter Sample API statuses/sample [5], which returns a small random sample (usually 1%) of all public tweets. The Statuses/filter "follow" real-time returns the postings of a list of specified Twitter users (5,000 at most) as they publish tweets; while "locations" tracks the tweets falling in a geographical area either according to tweet's embedded GPS coordinates or place names.

After specifying an area A, our system first retrieves a set of Twitter users who fall inside A using IndexedRDD variable user→latlon built in Section 4.3.1, and collects their live tweets via statuses/filter "follow". Our experiments in Section 4.4.2 show that doing so dramatically increases the number of local tweets and thereby boosts the number of detected local news in our system. Meanwhile, statuses/filter "locations" is also initiated

[4] https://dev.twitter.com/streaming/reference/post/status/filter

[5] https://dev.twitter.com/streaming/reference/get/statuses/sample

to collect tweets with embedded GPS coordinates or place names falling inside A. Finally, we also keep one's tweets captured from Twitter Sample API if he is from A. Note that as the system runs, we also keep following the newly found Twitter users belonging to A to track their real-time tweets.

4.3.3 Extracting Locality-Aware Keywords

"Hot" news or events in Twitter often cause, temporally or spatially, noticeable changes (e.g., word usage and increase in the number of related-tweets) in Twitter, thereby encouraging the exploitation of anomaly detection techniques such as the discrepancy paradigm [62, 5, 58] which makes a comparison between previous data (to build up a baseline) and the newly observed data to discover anomalies. These techniques are often addressed only from the perspective of detecting anomalies in the entire set of tweets (e.g., a set of tweets collected or aggregated together either geospatially or temporally), and in so doing might miss small-scale local news. Again, the data sparsity might make the problem worse. For example, to detect the news in Figure 4.1 is like finding a needle in a haystack from tweets because such a story, with only 6 tweets, hardly affects the word usage pattern in that evening at Westborough, MA.

However, if we look at the news story in Figure 4.1 from the view of individual people involved, such a small news poses noticeable changes in their word-usage pattern. For example, "Westborough Education Foundation Trivia Bee" are recently used words for 3 the Twitter users in that afternoon.

Therefore, given the sparsity of local news tweets, we utilize the following obser-

vations to capture such news. First, instead of looking for bursty or frequently used words with respect to a corpus of tweets from different Twitter users, we focus on the newly-used words with respect to the tweets from a single Twitter user. In other words, for a Twitter user, we are only interested in the words recently used by him. Such newly-used words correspond to the aspect of local news being "novel" as its nature of being news. Second, to reflect the aspect of local news being discussed by a limited number of people, we look for the words that are only used by a limited number of Twitter users, instead of the ones intensively used by people. Therefore, for a given tweet, we identify the words exhibiting the above two properties and call them *locality-aware keywords* in the sense that they are aware of the characteristics of local news. For example, consider the tweets in Figure 4.1 where "Westborough", "Education", "Foundation", "Trivia" and "Bee" are considered as locality-aware because they are new words used by this set of people.

Inspired by this, we recognize a word (only non-stopwords) in a tweet to be locality-aware by looking at 3 measures: how many times this tweet's publisher uses it, how many other users are using it and how many tweets contain it. To ensure the local news we detect are up to date, all these measures are computed in the latest 6-hour sliding time window from the enhanced local live tweet stream. If we treat a user's tweet as a sentence, then all his tweets in time order form a document, and all the tweets in the latest time window consist of a corpus. This is different from the idea of TF-IDF used in [102, 114] which treat each single tweet as a document.

We term the above 3 measures as *term frequency*, *document frequency* and *corpus frequency*, i.e., *TF*, *DF* and *CF*, respectively. Here we assume that a word appears at most once in a tweet (or counts only once if more), which is reasonable given the 140-

Algorithm 4: Online Extracting Locality-Aware Keywords and Online Clustering to Detect News

Input: the latest 6-hour sliding window in enhanced local live tweet stream – S; the locality-aware constraints – R_{TF}, R_{DF} and R_{CF}; the threshold values m, n and r

Output: news, i.e., clusters of tweets

1: load hash variables uid→tids, word→uid, word→tids, tid→uid, tid→cid, tid→r.t. in last time window;

2: **while** true **do**

a. pull a tweet from S, get its non-stopword tokens W, tweet id t, and user id u;

b. Locality-Aware Keywords $W_L \longleftarrow \emptyset$

c. **Extracting Locality-Aware Keywords Procedure:**
for each word $w \in W$ **do**

i). calculate TF_w, TF'_w, DF_w, CF_w, CF'_w from uid→tids, word→uid, and word→tids;

ii). $W_L \longleftarrow W_L \cup \{w\}$ **if** TF_w, TF'_w, DF_w, CF_w and CF'_w meet with the constraints of R_{TF}, R_{DF} and R_{CF};

iii). update word→uids, word→tids by inserting w and its corresponding u and t;

end for

d. update uid→tids by inserting u and t;

e. update tid→r.t. by inserting t and retweet number;

f. **Online Clustering Procedure:**
for each $W_L^m \in$ subsets of W_L with size m **do**

i). retrieve Q – the ids of tweets containing all words in W_L^m, from word→tids;

ii). retrieve the user set U in Q using tid→uid;

iii). **continue if** $|U| < n$;

iv). extract the largest group of tweets C from Q with the same cluster id c (a null c means that the tweets in C haven't formed a cluster yet);

v). calculate RT_C, which is the sum of retweet number of each tweet in C, using tid→r.t..

vi). **if** $|C| \geq \lceil \frac{|Q|}{2} \rceil$ **and** $|U| \geq \lceil r * RT_C \rceil$ **then**
$c \longleftarrow$ generate a new id **if** c is null;
assign t to cluster c in hash tid→cid;
report t as a *news* tweet to system UI;
break;
end if

end for

g. Remove obsolete tweets from uid→tids, word→tids, tid→uid, tid→cid and update word→uids;

end while

character limit. For a given word w in the tweet posted by user u, these measures are computed as: $TF_w = |T_u \cap T_w|$, $DF_w = |U_w|$, and $CF_w = |T_w|$. T_u denotes the tweets of user u, T_w denotes the tweets containing word w, U_w denotes the users who recently used the word w. Our heuristic is that, in order for a word w to be locality-aware, it should have a smaller TF_w (i.e. how many times it has been used recently by a Twitter user), which indicates that word w might be newly used by this user and thereby captures a news's "novelty"; DF_w (i.e., how many Twitter users have been using word w recently) should have a limited range (like $[3, \frac{|U_S|}{20}]$ specified in parameter settings in Section 4.4.4.1, where U_S is the set of Twitter users), to reflect the local news's characteristic of having a limited spread among people; and also CF_w should be small to avoid commonly used words like "day" and "people" etc. In our implementation, to account for the heterogeneity of the rates of publishing tweets for different users and for the number of tweets collected at different times and different places, we also use the relative frequencies of TF_w and CF_w, i.e., $TF'_w = \frac{|T_u \cap T_w|}{|T_u|}$, $CF'_w = \frac{|T_w|}{|T_S|}$, where T_S represents all current tweets. The constraints for TF, TF', DF, CF and CF' — denoted by R_{TF}, R_{DF} and R_{CF} — are discussed in Section 4.4.4.1.

4.3.4 Online Clustering to Detect News

As presented in Algorithm 4, we take into account the following two aspects to group tweets together. First, the tweets need to share at least a number m of locality-aware keywords to be grouped together. Second, at least n different Twitter users must exist in a cluster. Existing methods usually neglect the importance of these two aspects.

For example, GeoBurst [134] measures the semantic similarity between two tweets by performing random walks on their keyword co-occurrence graph to calculate the average probability that one tweet reaches another. However, without requiring a minimum number of keywords in a tweet, two tweets containing and sharing very few keywords could be mistakenly considered semantically coherent even if they are not on the same topic. In addition, TwitterStand [102] groups tweets together as long as they are similar enough in the TF-IDF vector space and in so doing, might form noisy clusters out of a single Twitter user's repeated tweets.

Therefore, in our method, to cluster an incoming tweet, we first retrieve a set of tweets sharing at least m locality-aware keywords. If these tweets were contributed by less than n Twitter users, or the majority of the tweets don't locate in the same cluster, then we don't group this new tweet and try another set of m locality-aware words. We also require that a news spreads among more local people. In Twitter, the spread extent of a tweet is provided by its retweet number, i.e., how many other Twitter users retweet it. We now define, for a given news cluster C, its spread extent RT_C to be the sum of the retweet number of each tweet in it. And the local spread ratio $spread_{local}$ is computed by $\frac{|U|}{RT}$, where U is the users contributing to C. In our experiments, we set $spread_{local} \geq r = 0.4$ to account for the local tweets that we might not capture.

The details of calculating the above measures are presented in Algorithm 4. Generally, Firefly uses a one-shot process, meaning that once a tweet is added to a cluster, it remains there forever. We will never revisit or recluster the tweet, which is desirable for real-time detection of news from a local live tweet stream. We don't incorporate additional care of the aspects from geographical dimension or temporal dimension as they are

implicitly reflected in the procedure enhancing the local live tweet stream and its 6-hour sliding window.

4.3.5 System User Interface

As shown in Figure 4.4, our user interface consists of two sources: a Twitter Timeline[6] and a Google Map based Web Application [102]. The Twitter Timeline allows a user to view a list of tweets collected for various purposes, such as real-time monitoring of a Twitter user's updates or searching for the latest tweets on a specific topic. Therefore, in order to demonstrate the latest news that we detect in real-time, a Twitter Timeline [7] is created via Twitter Collections API, which is very convenient for other Twitter users to view and even subscribe to. Note that the Collections API only allows for a user to retain a few thousands of tweets and automatically delete the oldest ones if it has too many tweets.

To display the events that we detect on the Google map-based web application, we utilize a procedure to estimate the geographical focus for a news cluster in [102]. This procedure, by making use of both the geographical information in tweet content and the source location of the users in an event cluster, computes a geographical focus as a whole by ranking the geographic locations mentioned in the cluster. After geotagging an event cluster, the Google map-based web application displays a marker for this event at its geographical coordinates.

[6] https://support.twitter.com/articles/164083
[7] https://twitter.com/bostonnewslocal/timelines/878280225074950144

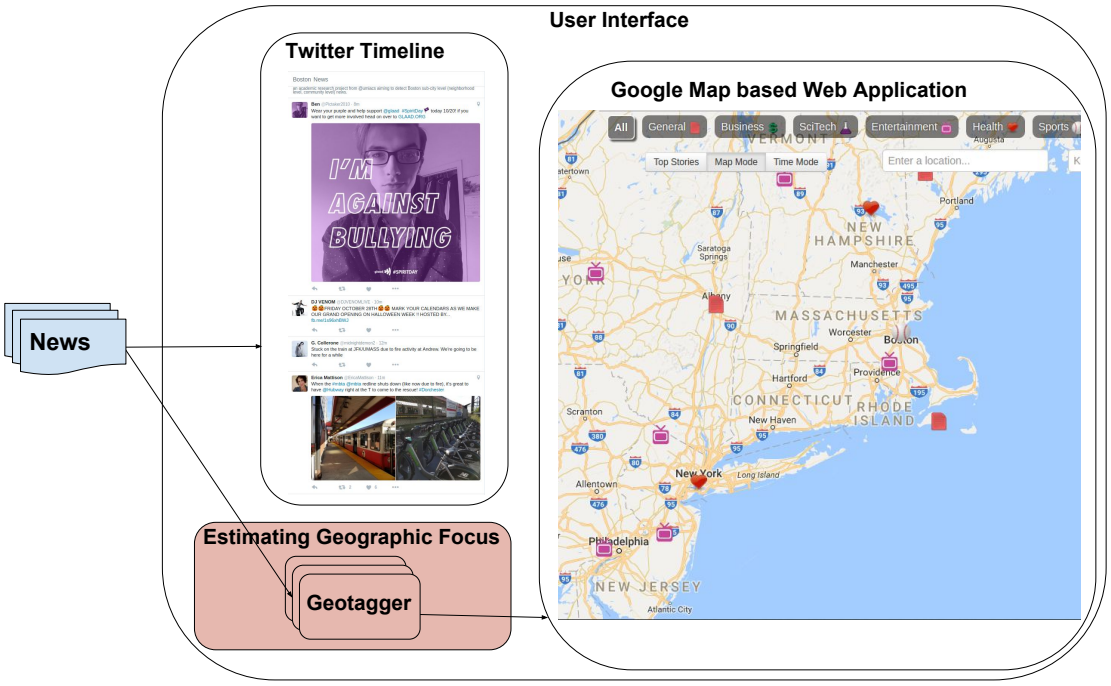


Figure 4.4: System user interface.

4.4 Experiments

4.4.1 Online Processing Settings and Efficiency

Our system adopts sliding time window techniques to meet the demand for online processing of a live tweet stream. The experiments are evaluated on a Spark cluster of 5 computing nodes where each node has two 6-core Intel Xeon E5-2620 v3 CPUs and 128GB of RAM.

For Online Geotagging, we utilize the Spark Stream to discretize the live tweet stream from the Twitter statuses/sample API into intervals of 23 seconds, which is the average time to accumulate 1000 tweets. Similarly, a 6-hour sliding time window is applied on the enhanced local live stream for locality-aware keyword extraction and online

clustering. The 6-hour window size is intuitively set in recognition of the fact that television media usually has four times of a locally-oriented news broadcasts in one day. The day of Jan 16, 2017 is chosen to evaluate our system for news detection with respect to the Boston metropolitan area i.e., the rectangle area [42.008339, -71.803026, 42.732923, -70.577545].

In our experiments, we find the major overhead is the Boosting Phase in Algorithm 3, which takes around 76 minutes to finish. But this procedure runs only once to start up the system and does not affect the timeliness of subsequent procedures. After the Boosting Phase, the online geotagging procedure takes an average of 3 seconds to process 1,000 tweets from the Twitter statuses/sample API, and geotags an average of 47 unknown-location Twitter users per second. Afterwards, Algorithm 4 processes 70 tweets per second on average (which is also the approximate arriving rate of tweets in enhanced local live stream) and reports about 3 tweet clusters per minute.

4.4.2 Twitter User Geotagging via Spark

4.4.2.1 Boosting Dataset

To boost the startup of geotagging Twitter users, we utilize a set of tweets collected between 09/2015 and 09/2016. This dataset consists of 2,876,822,081 tweets, 102,382,292 users and 824,303,126 pairs of neighbor-ships. Among these users, 31,250,047 have valid location source (successfully aligning profile-location to GeoNames or having embedded GPS coordinates) and are used to build-up the variable user→latlon. Accordingly, variable user→neighb. builds from the extracted neighbor-ships. Filtering down to

only reciprocal neighbors, we have a reciprocal graph of 24,946,962 vertices (8,787,152 of them have lat/lon coordinates) and 54,550,871 bidirectional edges.

4.4.2.2 Effectiveness

In lack of a ground-truth for Twitter users' locations, we exploit the *boosting dataset* to evaluate the effectiveness on coverage and accuracy. Specifically, for the 8,787,152 Twitter users with lat/lon coordinates in the reciprocal graph built in Section 4.4.2.1, their lat/lon coordinates are obtained from the profile-location or GPS coordinates in their tweets, and are thus treated as ground-truth. We then perform a leave-p-out validation by randomly sampling 10% (i.e., 878,715) of these Twitter users to evaluate the coverage and accuracy. The coverage is to calculate how many Twitter users in the sampling set would get geotagged, while accuracy is to calculate the mean distance error between the ground-truth and their estimated location.

To geotag the 10% users, we again utilize the sub-procedure iii) in the Online Geotagging of Algorithm 3. Our experiment shows that with $\gamma = 100$ km, $\lambda = 2$, 13.6% (i.e., 119,505 out of 878,715) test users get geotagged with a mean error of 228.66 km and a median of 27.93 km, which as shown in [24], is accurate at city-level for majority of test users.

Effect of Outlier Removal To evaluate the effect of outlier removal, we now exclude the step of outlier-removal in Algorithm 3 to geotag the 10% test Twitter users with γ fixed at 100 km and λ at 2. This brings us a lower 7.1% coverage with a larger mean error of 279.81 km, showing that removing outliers significantly increases the chances for

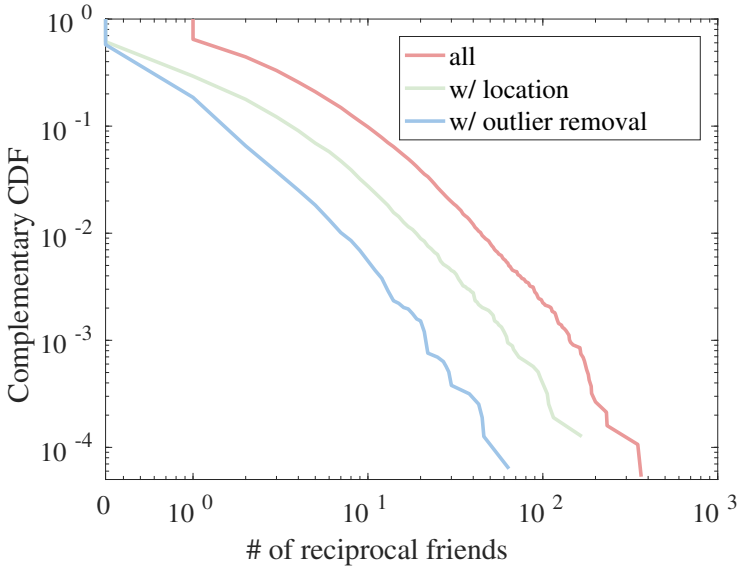


Figure 4.5: CCDFs of reciprocal friends.

test users to get successfully geotagged while without compromising accuracy.

λ	Coverage (%)	Mean Error (km)
1	53.3%	7900.54
2	13.6%	228.66
3	6.5%	251.34
4	3.9%	213.62
5	2.5%	191.65
10	0.6%	234.72
20	0.1%	187.16

Table 4.1: Effect of λ.

Effect of λ (the Minimum Number of Reciprocal Friends with Valid Locations) We first plot the distributions on the number of reciprocal friends of these 10% Twitter users in Figure 4.5, as well as the ones with locations and the ones that have survived from the outlier removal step. Figure 4.5 shows that lots of the Twitter users have very few reciprocal friends that have locations. In such a sparse reciprocal graph, it may not be fair to decide the location for a Twitter user only based on very few of his friends locations. To avoid generating noisy location estimations, a Twitter user is not going

to be geotagged until the number of his reciprocal friends having locations exceeds the required minimum.

To investigate the sensitivity of the minimum constraint parameter λ in Algorithm 3, we fix $\gamma = 100$ km and use different values of $\lambda = \{1, 2, 3, 4, 5, 10, 20\}$ for the 10% sampling test users and list the corresponding coverage and mean errors in Table 4.1. The results show that although $\lambda = 1$ is able to geotag more than half of the test users, it brings about an acceptably large error; $\lambda = 2$ seems to reach the best trade-off point between coverage and accuracy; while larger λ values have similar accuracy, they have relatively low coverage.

4.4.3 Enhanced Local Live Tweet Stream

At the start of the day on Jan 16, 2017, 176,007 users are found in the input Boston bounding box. Among them, 101,409 provide valid location source (profile-location or GPS), and the remaining 74,598 are geotagged using Algorithm 3. Following these two sets of users to track their real-time postings comprises of the two sources of Streaming API w/ "follow" I and II as listed in Table 4.2, respectively[8].

Source	# of tweets		# of news	
	Local tweets	Cluster tweets	Involved	Exclusive (acc. %)
Sample API	6,182	638	167	21 (35.5%)
Str. API w/ loc.	76,983	2,123	359	76 (52.9%)
Str. API w/ fol. I	2,986,291	23,120	2,489	1,241 (58.5%)
Str. API w/ fol. II	1,730,889	16,654	1,857	609 (52.7%)
Total	4,800,345	43,535	N/A	N/A

Table 4.2: Contribution of different local live tweet sources.

Table 4.2 first shows how many local tweets (i.e., the tweets that fall in the given

[8] Multiple API tokens are used because one only follows up to 5000 users.

area or are published by people there) as well as how many cluster tweets (i.e., the tweets that compose the detected clusters) each source contributes to our enhanced local live tweet stream. We collected a total of 4,800,345 tweets from the Boston area during a 24-hour period. Among which, Twitter Streaming API statuses/filter "follow" (I and II) contributes the most, by making up 98.27% of all the tweets in the enhanced local live tweets, while the other two sources only output a very small amount of local tweets. Similarly, regarding the tweets comprising the clusters, 93.66% come from source Streaming API statuses/filter "follow". For example, all the tweets related to the news in Figure 4.7 are in source "follow". More importantly, Table 4.2 further shows that tracking Twitter users who don't have valid location sources also make significant contributions just like tracking the users with valid location sources. This reinforces the important role that the online Twitter user geotagging procedure plays in our system.

In addition, Table 4.2 also lists the number of "Involved" news (i.e., how many news a source's tweets have participated in forming) and the number of "Exclusive" news (i.e., how many news a source's tweets have exclusively formed, in other words, these news are formed by tweets only from this source), along with its accuracy of positive local news (the accuracy evaluation method is detailed in Section 4.4.4.4). The result shows that the majority of news events are generated using the tweets in Streaming API statuses/filter w/ "follow", indicating that by tracking local Twitter users, our method is able to find much more news than solely using the Twitter's publicly available tweet streams.

4.4.4 Local News Detection

In this section, we evaluate the performance of our system on detecting news from the enhanced local live tweet stream using *mutual recall* and *precision*. Mutual recalls are evaluated between our system and a set of local news media agencies, together with a few baseline approaches. As for precision, we recruited 3 volunteers to individually judge the detected news and collect the results using the strategy of majority votes.

4.4.4.1 Parameter Settings

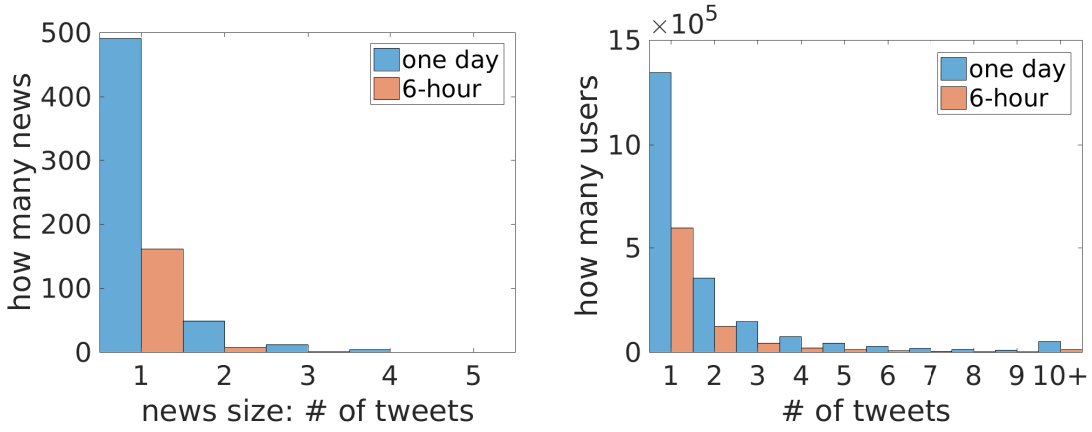
Note that although some of the following parameter settings depend on the specific input city, they are simply statistics and easy to infer for other places. There are 3 constraints for a word to be locality-aware: R_{TF}, R_{DF} and R_{CF}. For R_{TF}, the main goal is to capture a local news's nature of being new and to reflect a person's word-usage anomaly, by requiring both TF and TF' to be small (of course, at least greater than 0). This means that, an upper boundary needs to be imposed on TF. To obtain an empirical value of this, we collect the tweets posted by the Twitter accounts listed in Table 4.4 (note that the Twitter account of @fox25news has changed to @boston25 in April 2017), and perform an analysis, for each individual agency, of how many of its tweets are about the same news. The results, presented in Figure 4.6a, show that an agency usually tweets only 1 or 2 tweets (5 at most) about the same news. The situation is similar when the time period narrows down to a 6-hour (e.g., from 15:00 to 21:00). We therefore set the upper bound of TF to 5. Figure 4.6b reminds us that this value could work for most of Twitter users as they usually post less than 10 tweets, either in one day or in a 6-hour time window,

This value, however, seems too strict for Twitter users who publish 10 or more tweets and perhaps keep posting updates on the same news event. We therefore turn to TF' to relax the constraint of TF, and set a threshold value of 0.3 for TF'. To summarize, we have $R_{TF} := (|T_u| < 10 \ \wedge \ TF \leq 5) \vee (|T_u| \geq 10 \ \wedge \ TF' \leq 0.3)$.

R_{TF} alone, however, is not enough because it would mark most of the words for most of Twitter users as locality-aware. We further utilize DF to explore another characteristic of local news: being "limited spread". Recalling the fact that one-fifth of the population are active Twitter users, we set $R_{DF} := 3 \leq DF \leq \frac{|U_S|}{20}$, where U_S are all the users in the time window S. Our argument is that when $DF = 3$, there might be an equal number of users reporting the same activity in Twitter. This further indicates that in reality, there might exist an ongoing news event that involves 15 people. Likewise, we set the upper boundary to 1% of the population, which is around $\frac{|U_S|}{20}$. The distribution of detected cluster size in Figure 4.8a further validates our assumption.

Finally, there is an additional constraint R_{CF} to get rid of commonly used words. Our analysis on the CF's of most common non-stopwords in English shows that they have a min CF of 0.57% (max: 2.7%, mean: 1.6% and median: 1.8%). Also considering that the average number of tweets published by a Twitter user is around 2 (e.g., in Figure 4.6b, 2.30 and 1.82 for one day and 6-hour) and DF's upper bound, we set the upper bound of CF to $\frac{|U_S|}{10}$. Therefore, R_{CF} is set as $R_{CF} := CF \leq \frac{|U_S|}{10} \wedge CF' \leq 0.57\%$, which helps us to successfully recognize words like "trump", "martin", "luther", "day" and "people" as not locality-aware.

We then have 3 more threshold values to set for online clustering in Algorithm 4. For the least number of overlapping words between two tweets to cluster together, we set



(a) News size by news agency

(b) Tweet number by Twitter user

Figure 4.6: Histograms of # of tweets. (a) Histogram of # of tweets in a news by each individual news agency. (b) Histogram of # of tweets posted by each individual Twitter user.

$m = 5$ because it is usually large enough to cover a news's "who", "what" and "where" information, e.g., the bold words in the example event of Figure 4.1. In our experience, a larger m makes clustering tightly cohesive yet might split the same news story into several clusters; while a smaller m might not fully reveal a story's own trait and groups non-related things together. To be consistent with R_{DF}, we set the least number of people in a cluster $n = 3$. At last, although we require that a local news should have more local people talking about it, other than the people from outside world, we set the local spread ratio threshold $r = 0.4$ to deal with the tweets we might miss.

4.4.4.2 Local News Media Agencies and Baseline Approaches

Reputable Local News Media Agencies We select 9 Boston local news agencies, as listed in Table 4.3 in the form of "@ScreenName", to collect their news tweets as a groud-truth dataset to compare with. The news stories in the news agencies come from

two parts: tweets posted by their accounts and articles published in their websites. The articles are collected by crawling their websites every 5 minutes listed in Table 4.3. Due to copyright constraint, we only retain an article's *title, url* and *publish datetime*.

Agency Name	Twitter			Newspaper	
	Screen Name	Tweet Num.	News Num.	Website URL	Article Num.
7News Boston	@7News	73	61	http://whdh.com/news/local/	15
boston.com	@BostonDotCom	29	21	https://www.boston.com /tag/local-news	4
The Boston Globe	@BostonGlobe	156	128	http://www.bostonglobe.com /?refresh=true	22
Boston Herald	@bostonherald	106	95	http://www.bostonherald.com /news/local_coverage	21
CBS Boston	@CBSboston	64	52	http://boston.cbslocal.com /category/news/	83
FOX25	@fox25news	85	64	http://www.myfoxboston.com /news/local	13
Globe Metro	@GlobeMetro	12	11	http://www.bostonglobe.com /metro	12
Metro Boston	@metroBOS	26	15	http://www.metro.us/ boston/news/	22
WCVB 5	@WCVB	143	110	http://www.wcvb.com /local-news	27

Table 4.3: The 9 reputable Boston local news agencies.

As most of the tweets posted by these agencies are of good quality, we perform a simple clustering algorithm to extract news from them. That is, for a single news agency, as long as any two of his tweets share 5 non-stopwords, we group them together. The value of 5 is heuristic, by accounting for the number of words to specify a story's "who, what and where". As presented in Table 4.3, the amount of tweets and the amount of news these agency cover are various, with "@BostonGlobe" being the most active and "@metroBOS" the least active.

Baseline Approaches We also compare our method with the following four base-

line approaches listed.

- *TwitterStand*: TwitterStand [102] groups news tweets into cluster of tweets to form news stories using a TF-IDF based similarity metric. In the experiments, the clustering similarity threshold ϵ is set to 0.8. It is worth mentioning that their concepts of TF and DF are different from ours in the sense that they treat each single tweet as a document while we treat all of a user's tweets as a document and each of his tweets as a sentence.

- *TwitterStand-3*: By default, TwitterStand only reports a cluster as a news story if it has more than 10 tweets. In this setting, we relax the minimum number of tweets to 3, out of the consideration of fairness for TwitterStand to be able to detect news of small scale.

- *EvenTweet*: EvenTweet [5] first identifies temporal bursty keywords (using a Gaussian distribution based discrepancy paradigm) and spatial local keywords (using the entropy of a word's spatial distribution) and then clusters them together according to their spatial density distribution. The spatial density distribution is calculated based on a $N \times N$ grid tessellation. We set $N = 50$ in our experiments. The temporal bursty keywords are identified using a Gaussian distribution based discrepancy paradigm, while spatial local keywords identified using the entropy of a word's spatial distribution on a regular grid tessellation.

- *GeoBurst*: GeoBurst [134] first generates candidate events by identifying pivot tweets based on geographical and semantic similarities and then ranks the candidates according to their spatiotemporal burstiness to filter out noisy ones. Geo-

graphical similarities between tweets are calculated by a kernel function on their spatial distance, while the semantic similarities are calculated by performing a random walk procedure with restarts on tweets' keyword co-occurrence graph. In our experiments, we adopt the default settings in their method, i.e., the spatial distance kernel bandwidth is set to 0.01, the random walk restart probability and similarity threshold are set to 0.2 and 0.02, respectively.

As summarized in Section 4.2, TwitterStand (or TwitterStand-3) is an event-anchored method and therefore is fed with the same enhanced local live tweet stream in Firefly to detect news, while the last two are location-anchored methods which only take geotagged tweets (with embedded GPS coordinates) as input. In the 4,800,345 tweets we collected, 33,966 tweets are geotagged (Streaming API w/ follow: 23,810; Streaming API w/location: 10,101; Sample API: 55) and chosen as the input to EvenTweet and GeoBurst. Note that the geotagged tweets are less than the total tweets obtained from Streaming API w/location because this API also returns non-geotagged tweets containing place names that fall in the given query area.

By default, EvenTweet represents each cluster as a group of keywords, and we retrieve the tweets from which the keywords are extracted to represent its clusters to be consistent with other methods. To maximize the number of potential news detected in EvenTweet and GeoBurst, a cluster in them is selected to be output as long as it has at least 3 tweets.

4.4.4.3 Mutual Recalls

The mutual recalls are computed by examining how many news in the news agencies or baseline approaches have been found by our system and *vice versa*. We claim a news cluster $c_{\mathbf{X}}$ in agency **X** recalls a news cluster $c_{\mathbf{Y}}$ in agency **Y** if there is a tweet in $c_{\mathbf{X}}$ and another tweet in $c_{\mathbf{Y}}$ that share at least 5 non-stopwords. The results are summarized in Table 4.4, in which a news agency's "@Screen Name" is to represent its tweets news. Also, to make the table compact, we give each agency an order denotation in the column headers, ranging from **A** to **W**. Below the column headers are the number of news found in an agency or our system Firefly. So for a cell, it shows how many news row **X** covers over column **Y**.

Table 4.4 shows that Firefly achieves high recalls against most of news agencies. For example, we successfully detect news like "Stabbing Reported at Stoughton Home of UMass Boston Chancellor", "Dog killed by coyote in Gloucester, police issue warning" and "A woman caught in the line of fire in Lyn" etc which are also reported by "@7News". In contrast, a very large portion of news in Firefly don't receive coverage from any of the listed news agencies, e.g., "There is a growing collection of lonely hand warmers at Fallon Field in #Roslindale", "Hockey star Kacey Bellamy took a break from prepping for the 2018 Winter Olympics to chat with @BrooksSchool girls hockey team today!" and "Just a portion of the many people that volunteered today to build STEM kits for Boston schools" etc. This confirms the effectiveness of our design of enhancing local live tweet stream and extracting locality-aware keywords.

The result is in accordance with our observation that there would be lot more news

happening in an area than reported locally [85] , and is consistent with our expectation because we try to identify various kinds of news, activities and news like missing pets, sales events, concerts and farmer's market etc., while local news agencies usually publish news of greater public interest.

		A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**
# of news		3364	409	2331	184	179	61	21	128	95	52	64	11	15	110	15	4	22	21	83	13	12	22	27
Firefly	**A**	3364	305	1213	135	164	48	21	85	32	41	49	6	10	69	11	4	16	6	20	9	7	4	10
TwitterStand	**B**	200	409	1607	71	46	7	5	13	15	4	4	0	2	12	1	0	5	2	11	1	0	1	3
TwitterStand-3	**C**	215	395	2331	51	66	8	6	16	19	4	5	0	5	15	2	0	6	2	13	1	1	2	5
Eyewitness	**D**	236	218	292	184	151	6	6	3	3	15	2	0	3	15	1	0	1	1	6	1	2	3	1
GeoBurst	**E**	132	64	202	126	179	7	1	3	3	7	5	0	3	5	0	0	0	2	3	1	0	1	3
@7News	**F**	49	73	212	2	13	61	3	4	7	2	6	0	1	9	13	0	2	1	3	0	1	2	1
@BostonDotCom	**G**	21	22	47	1	1	3	21	6	5	0	2	1	0	3	0	3	3	1	2	1	1	1	1
@BostonGlobe	**H**	85	83	210	2	6	4	6	128	4	3	1	2	0	5	0	2	12	0	4	0	6	3	0
@bostonherald	**I**	38	82	179	1	3	7	5	4	95	0	7	1	1	5	2	2	2	9	1	1	0	0	1
@CBSboston	**J**	41	64	149	2	8	2	0	3	0	52	1	0	0	4	0	0	1	0	7	0	1	0	0
@fox25news	**K**	49	23	64	2	5	6	2	1	7	1	64	0	2	2	4	1	0	2	0	9	0	0	1
@GlobeMetro	**L**	6	0	0	0	0	0	1	2	1	0	0	11	0	0	0	0	4	0	0	0	3	0	0
@metroBOS	**M**	11	27	62	2	5	1	0	0	1	0	2	0	15	0	1	0	0	0	0	1	0	3	2
@WCVB	**N**	69	95	217	7	13	9	3	5	5	4	2	0	0	110	1	0	1	0	4	0	0	1	13
7News Boston	**O**	12	2	2	1	0	13	0	0	2	0	4	0	1	1	15	0	1	0	0	0	1	1	1
boston.com	**P**	4	0	0	0	0	0	3	2	2	0	1	0	0	0	0	4	1	1	0	0	1	0	0
The Boston Globe	**Q**	17	30	72	1	0	2	3	12	2	1	0	4	0	1	1	1	22	0	2	0	4	1	1
Boston Herald	**R**	7	6	15	1	3	1	1	0	9	0	2	0	0	0	0	1	0	21	1	1	0	1	1
CBS Boston	**S**	20	147	352	6	3	3	2	4	1	7	0	0	0	4	0	0	2	1	83	0	0	1	0
FOX25	**T**	13	1	8	1	2	0	1	0	1	0	9	0	1	0	0	0	0	1	0	13	0	0	0
Globe Metro	**U**	9	0	1	0	0	1	1	6	0	1	0	3	0	0	1	1	4	0	0	0	12	1	1
Metro Boston	**V**	4	13	30	6	3	2	1	3	0	0	0	0	3	1	1	0	1	1	1	0	1	22	2
WCVB 5	**W**	10	25	43	1	3	1	1	0	1	0	1	0	2	13	1	0	1	1	0	0	1	2	27

Table 4.4: The mutual recalls between Firefly, baseline approaches and the 9 reputable Boston local news agencies.

In contrast, the default settings of TwitterStand have much lower recalls across the 9 local news agencies. Although relaxing its cluster size to have minimum of 3 tweets yields many more clusters, it doesn't yield clearly higher recalls. We conjecture that in doing so, TwitterStand-3 is reporting many small clusters for the same news due to the fragmentation problem in its online clustering [102]. For example, the 409 news of TwitterStand are covering 1,607 news of TwitterStand-3. This also explains TwitterStand-3's

extremely asymmetric mutual recalls over the local news agencies. In contrast, Firefly's locality-aware keywords based clustering is more reliable by finding word-usage anomaly from the perspective of a Twitter user instead of a tweet itself.

It comes as no surprise that EvenTweet and GeoBurst, both of which only run on sparsely available geotagged tweets, have low recalls across the local news agencies too. This is essentially because geotagged tweets cover very limited news in our dataset. For example, none of the news tweets posted by local news agencies contain geotagged tweets. Similarly, in all the tweets clusters generated by our system Firefly, only 633 of them contain geotagged tweets and only 107 of tweets clusters are formed by only geotagged tweets. This shows that by utilizing non-geotagged tweets, we are able to detect much more local news than methods EvenTweet and GeoBurst and further reinforces the importance of enhancing local live tweet stream by finding and tracking local Twitter users.

Another factor contributing to the low recalls of TwitterStand-3 might be its classifier step which throws away more than half of the tweets (68.9%). To verify this, we omit the classifier in TwitterStand-3 and the resulting clusters are able to recall 1,135 ones detected in Firefly. It, however, outputs a total of as high as 9,314 clusters but 5,713 of them are covered by Firefly, indicating that in so doing, TwitterStand-3's clustering is working very poorly without effectively merging similar groups of tweets. In contrast, Firefly's locality-aware keywords based clustering doesn't rely on a pre-trained classifier and is more reliable by finding word-usage anomaly from the perspective of a Twitter user instead of a tweet itself.

Table 4.4 also shows that our system Firefly misses quite a few of the news for some

agencies such as "@bostonherald" and its newspaper "Boston Herald". To dig out the reasons behind this, we collect the 63 news of "@bostonherald" that we missed but only identified 5 of them as relevant. The major reason we missed these 5 news is because there are extremly few tweets covering them. For example, the news 'Good Samaritan rescues trapped dog from inferno" seemed contributed only by "@BostonHerald" as we only find this single 1 related tweet. The situation is similar regarding the website articles we missed in "Boston Herald", except that some of these articles don't appear in the tweets of its official news agency Twitter account.

It is also not unusual to find that some website articles relate to no tweets as we find local news agencies were not always publishing tweets about their website articles. One example is "@CBSboston" v.s. "CBS Boston": "@CBSboston" didn't post a single tweet about an accident of "Driver Suffered Serious Injuries When Car Crashed Into Pole In Carver" published in its website. This might give more explanations for website articles that our system didn't capture, and also inspire us to integrate cross-domain news source [139] to further mitigate the tweet data sparsity in the future. Another interesting observation from Table 4.4 is that different news agencies tend to cover different stories, with very few overlapping ones. This makes platforms like ours more valuable as a user doesn't have to browse different news agencies to learn about what is happening out there.

4.4.4.4 Precision

We asked 3 human judges to independently examine the 3,364 clusters of tweets detected in Firefly. As shown in Figure 4.7, each candidate news is a set of tweets with

their urls. The set of tweets are selected by having the most non-stopwords, retweet numbers and overlapping words with each other and no more than 5 tweets. The drop-down list provides 3 available options: "Positive", "Neutral" and "Negative", which are used by the judges to answer the question: "Are the three or more tweets describing the same local news?". The instructions given to the judges are summarized as follows:

Each candidate news has a set of tweets, followed by their urls. Please read the tweets and answer if they are talking about the same news. A local news, here, refers to an event that happens in Boston Metropolitan area. For example, local news can be about traffic, weather, missing persons/pets, farmer's market, yard-selling and book-selling, happy hour of bars and restaurants, crimes, protests, gatherings, award-nominations, and parties, meetings, celebrations, conferences, sports games etc. You can utilize the tweets' urls to get more information such as where the news happened. If you can't determine where it happened, choose "Negative". National/international news are recognized as "Neutral". News that happened in another place, like sports held in another city, should be "Negative". Also if you don't think the presented tweets are representing a news, select "Negative".

Figure 4.8b presents the distribution of judges' answers of the 2,574 events out of 3,364 that received a majority of "Positive"s or "Negative"s. Among the 2,574 events, 73.6% had 2 or more "Positive"s and were consented to be local news. The median number of tweets and median number of users in these local news are only 7 and 6, respectively, as shown in Figure 4.8a. We also discovered that most of the clusters with a majority of "Negative" were formed by a set of people tweeting like "My fitbit for 1152017 6145 steps and 29 miles traveled". This surprised us because this crowd behavior

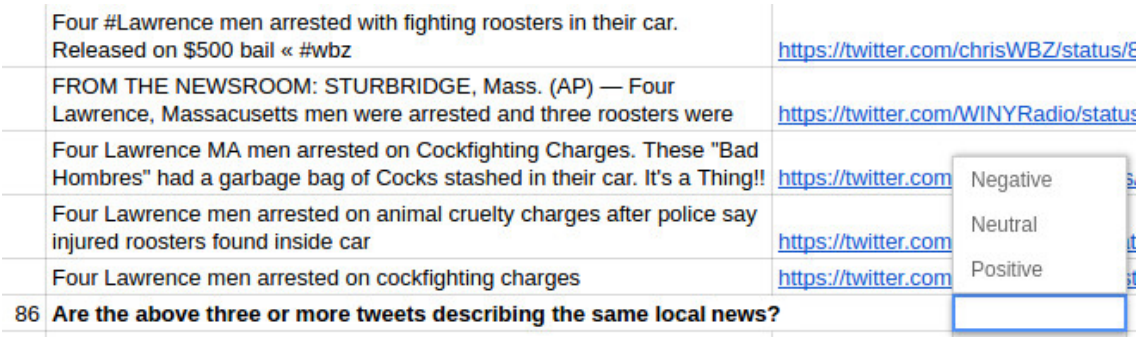
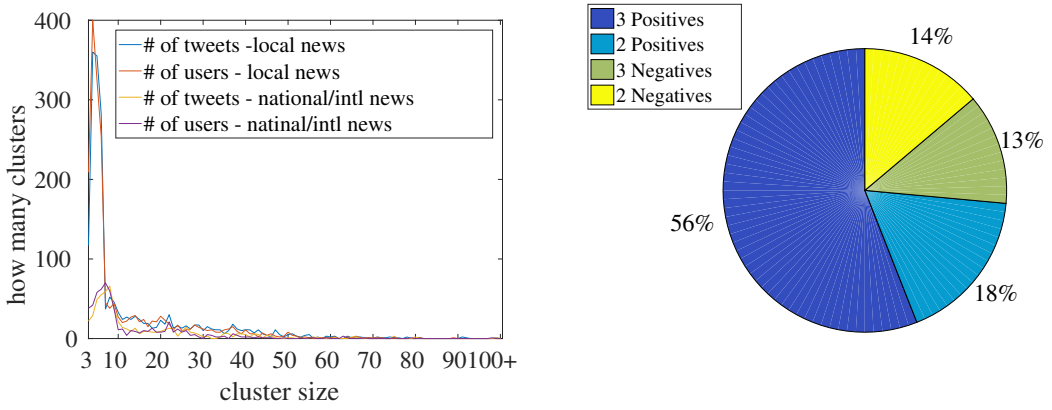


Figure 4.7: Example of human judging UI.



(a) Distribution of tweets and users.

(b) Distribution of human evaluation.

Figure 4.8: Distribution of news cluster sizes and human evaluation.

meets our constraint for local news. In addition, out of 3,3364 news we detected, 649 received 2 or more "Neutral"s and were considered to be national or international news.

	≥2 Positives	≥2 Neutrals	≥2 Negatives
Firefly	1,894 (56.3%)	649 (19.3%)	680 (20.2%)
TwitterStand	14 (3.4%)	306 (74.8%)	76 (18.6%)
TwitterStand-3	123 (5.28%)	1302 (55.9%)	816 (35.0%)
EvenTweet	44 (23.9%)	27 (14.7%)	90 (48.9%)
GeoBurst	52 (29.1%)	21 (11.7%)	70 (39.1%)

Table 4.5: Proportions of different types of tweet clusters.

Next, we evaluate the clusters in TwitterStand, TwitterStand-3, EvenTweet and GeoBurst in the same way and list their proportions of news receiving more than 2 "Pos-

itives", 2 "Neutrals" and 2 "Negatives" respectively in Table 4.5. In comparison, among the 409 clusters in TwitterStand, only 14 are identified as local news. The low proportion of local news in the default settings of TwitterStand is caused by its constraint that it takes at least 10 tweets to form a cluster. Although relaxing this limit to 3 tweets in TwitterStand-3 captures more local news, its non-news proportion increases much more by falsely recognizing some repeating tweets from Twitter users as news, e.g. "@healy-like". In contrast, by only exploiting the sparsely available geotagged tweets, EvenTweet and GeoBurst are only able to detect a small number of positive local news. Similarly, in Firefly, out of the 107 clusters that are formed by only geotagged tweets, 47 of them receive ≥ 2 "Positives" and are considered positive local news. This further illustrates that making only use of geotagged tweets will miss the majority of local news reported in non-geotagged tweets.

Note that TwitterStand captures national or international news ($\geq$2 Neutrals) at a very high accuracy by setting the cluster size to be ≥ 10 tweets. And the mean and median number of tweets in such clusters in TwitterStand are 127 and 49, respectively, much larger than 16 and 11 in Firefly. This is because Firefly has a different strategy to find such news in the sense that, from the perspective of an individual Twitter user, Firefly is only interested in some of his latest tweets that are discussing different content from his old ones, while TwitterStand might take all his tweets as news-related. This difference becomes more significant when it comes to columnists or sports reporters who might post many updates on the same news event. In addition, our 6-hour sliding window and the constraint for locality-aware keywords to be used by a limited number of people might also contribute to the relatively smaller number of tweets in national or international news

clusters detected in Firefly.

4.5 Conclusions and Future Work

In this chapter, we presented a system called Firefly to detect news for a given geographical area. In order to deal with the infamous sparsity problem in publicly available Twitter data, Firefly first enhances the local live tweet stream by identifying a large body of Twitter users in an area to follow via an online geotagging procedure and thereby significantly increases the amount of tweets generated from that area. With the enhanced local live tweet stream, we propose a method to identify locality-aware keywords and further use them to cluster tweets together to detect news. Comparing with news extracted from a set of local news agencies' tweets, our system achieves the highest recalls, and at the same time, outperforms the baseline approach TwitterStand regarding both recall and precision in detecting local news, and more importantly, is able to detect much more local news than the approaches that only use geotagged tweets.

A small portion of news might be present in two or more clusters if these news don't get updates in a time period that exceeds 6-hour, which is the main reason whey Table 4.4 is not symmetric for Firefly. A remedy to this problem in the future might be to simply lengthen the time window or to keep a pool of news clusters before the current sliding time window and keep them active if they receive updating tweets. In addition, the importance of various users should be addressed differentially. For example, reporter or news agencies should be more trustworthy to publish news. Additionally, as the human verification yields a ground-truth of local news, a learning procedure might be explored

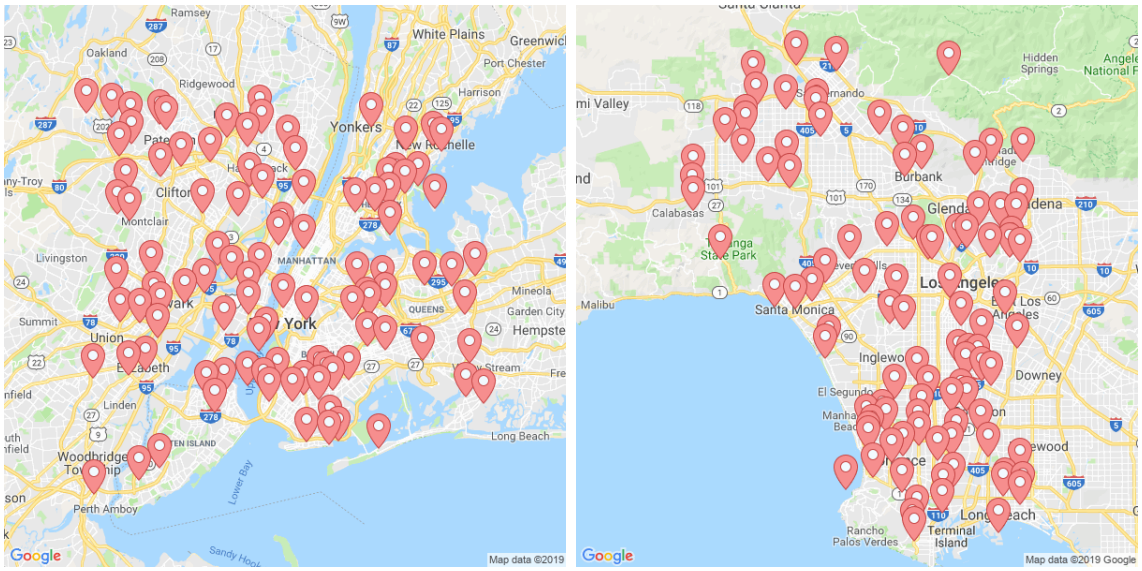
to help determine the parameter values in extracting locality-aware keywords and online clustering. We leave these questions for our future work.

Chapter 5: Learning Embeddings of Spatial, Textual and Temporal Entities in Geotagged Tweets

5.1 Introduction

Twitter, one of the most popular micro-blogging services, allows users to write and share short messages, called *tweets*, on a wide variety of topics like life experiences and daily activities. Contributing to its convenience and promptness in composing contents, Twitter serves as an essential platform in learning people's real-world activities in almost real time. Geotagged tweets are particularly interesting in the sense that they provide complement information about the place of interest, e.g., where the activities occur. Such location information is crucial when profiling human activities by completing the three pieces of information regarding *where*, *when* and *what*.

In this chapter, we aim to uncover the correlation between location, time and topic in human's urban activities hidden in geotagged tweets. For instance, Figure 5.1 presents some sampled locations of interest that people often check in New York City and in the city of Los Angeles. Looking at these locations, one may wonder why people go to these places, what they are actually doing there and when do they usually go there, etc. On the other hand, people may also wonder: given a city, which places are popular for its resi-



(a) NYC (b) LA

Figure 5.1: An overview of 100 random locations in NYC and LA.

dents to participate in certain activities like visiting museums, eating food and shopping, etc; and if they do, when they would do so. Effectively answering such questions has a wide range of applications like urban planning [48], location recommendation [139], disaster recovery and location function classification [75]. For example, one can query the popular locations for certain activities in his living place and thus help newcomers quickly integrate to local life or tourists to make sightseeing plans.

It is, however, challenging to extract location, time and topic context from geotagged tweets. First, although geotagged tweets provide GPS coordinates indicating where people participate in activities, these coordinates often impose certain disagreement even for the same event at the same place, due to the flexibility of people's movement and sometimes the noise of GPS satellite signals. For instance, one may notice that the GPS points extracted from the check-in tweets on the same local restaurant may form a cloud of points on map instead of a single pinpoint. Aside from this, it is also unrealistic

to include every possible geographical location because of space continuity. Therefore, it is imperative and practical to extract locations of interest in the form of clusters of GPS points. Second, it is hard to effectively and efficiently capture the cross-modal correlations between the spatial, temporal and textual aspects of people's daily-life activities. For example, the techniques of document-term matrix, TF-IDF and Single Value Decomposition (SVD) are often applied to analyze the co-occurrence relationship between locations and words. Such methods, however, can not be easily modified to cope with data of three or more dimensions. Tensor rank decomposition [36] is more promising in modeling high-dimension data but less applicable for large-scale datasets due to its computational complexity. In order to find the relationship between locations and topics, some studies treat location as a latent variable in their generative topic model [131, 44, 140, 117]. These models usually do not yield uniform representations for location, time and topic in the same latent space and are less suitable for applications like performing cross-modal searches.

This chapter aims to learn to represent the spatial, temporal and textual entities in the geotagged tweets by means of embedding vectors in the same semantic space. We propose LeGo to accomplish this learning task. The general idea of LeGo works as follows. First, LeGo extracts essential spatial, temporal and textual entities from the geotagged tweets. Spatial entities refer to locations of interests which witness the aggregation of people. They are usually identified using a clustering algorithm [48, 134] and are in the form of groups of tweet locations. For the publish time of tweets, LeGo uses the features like hour-of-day and day-of-week as temporal entities because (1) Such features are in recognition that these integral values of time play as very strong signals for people to

follow in order to arrange and conduct life activities; (2) These features are of great importance in revealing a location's topic category [133, 48, 75]. As for textual entities in tweets, we address the extracted keywords and phrases by getting rid of stopwords. Second, LEGo systematically constructs a co-occurrence graph that spans spatial, temporal and textual entities in tweets. In particular, the nodes represent the entities, and the edges are weighted by the number of times that two nodes co-occur in tweets. Third, LEGo exploits a graph learning algorithm that approximates the stationary residing probabilities between nodes which result from performing personalized random walk procedures. In essence, the learning algorithm aims to generate low-dimensional representations of nodes so that such representations would maximize the likelihood of one node yielding another node to be the probability of this node residing at another node. At last, although the co-occurrence graph between entities from different dimensions is beneficial for capturing the cross-modal correlations, it neglects the explicit similarities that may potentially exist between locations, such as spatial proximity and topic likeliness. To better support queries like location-similarity search, LEGo-LS extends the basic co-occurrence graph by supplementing edges between locations to address their spatial proximity and topic likeliness. Since the edges in the graph are now weighted differently, the graph learning algorithm is also modified to be compatible with the heterogeneous graph.

The contributions of this chapter are summarized as follows.

- First, we comprehensively profile people's activities in Twitter from 4 aspects: location, words, hour-of-day and day-of-week.
- Second, for cross-modal search, we construct a co-occurrence graph to calculate stationary residing probabilities between nodes, which subsequently guides the learning

process in the graph embedding algorithm.

- Third, for location-similarity search, we add explicit edges between locations to the co-occurrence graph to address their spatial proximity and topic likeliness, and thereby capture such within-location affinities.

The rest of this chapter is organized as follows. Section 5.2 reviews related work. Section 5.3 briefly describes the problem and provides an overview of the method. Section 5.4 details the design and implementation of the proposed method. Section 5.5 presents the experimental evaluation results. Section 5.6 contains concluding remarks as well as directions for future research.

5.2 Related Work

There has been much work on identifying correlations between locations and textual contents and sometimes time factors. Some work has focused on discovering geographical topics [44, 77, 117]. For example, Hong et al. [44] introduce the geographical location in a tweet as an additional component to a mixture of graphical language models to reveal underlying topics at a location. Wei et al. [117] furthermore introduce the time factor into the probabilistic graphical models and show its importance in discovering latent local events. Our method is different from these works because they rely on probability graphical models which impose prior distribution assumptions on the existing data while we rely on the simple co-occurrence relationship to learn embeddings. Liu et al. [77] treat hashtags in tweets as potential topics and investigate their associations with time and regions. The associations are primarily determined by the closeness of tweets

with respect to location, time and textual content. Although the study [77] is closer to our work in the sense that we both utilize the correlations between textual content and locations, there remain significant differences. First, we encode the correlations between location and topics in embeddings while [77] relies on an ad-hoc index value. Second, the embeddings learned in the same semantic space enable us to perform cross-modal queries while [77] only addresses the query of finding which hashtags are most frequently associated with the given time and region windows. Instead of geographical topics, there are also some works on detecting geographical trends from tweets [4, 3, 80]. For simplicity, the geographical trends can be viewed as bursty words whose occurrences are experiencing significant and abrupt increases within the given time and space windows. The main goal of geographical trend detection is to efficiently identify and query bursty words in a live tweet stream. For example, both Abdelhaq and Gertz [3] and Magdy et al. [80] have designed specific storage and indexing data structures as well as anomaly detection metrics to support the queries of geographical trends. The key factor differentiating our method from these works is that they essentially perform anomaly detection while we focus on normal real-life associations between locations, time and topics. This also makes our work different from a set of studies on local event detections [58, 140, 131, 134] which also emphasize anomaly detection.

Therefore, we are more interested in studies which similarly represent locations and topics in the form of vectors. The document-term matrix-based techniques (such as TF-IDF, SVD) provide a typical way of presenting multi-dimensional data as vectors. However, these vectors typically capture the information only from the same dimension and are limited in cross-modal queries. Besides, such techniques are difficult to extend

to high-dimensional data. Recently, there have been efforts bringing the technique of word2vec [87] to location-based social networks in order to learn embedding representation of locations and users [91, 86, 54]. The foundation of these methods lies in a graph embedding strategy proposed in DeepWalk [92]. In specific, in order to exploit the idea of word2vec in a graph, DeepWalk analogizes a node in the graph as a word and creates pseudo sentences by simulating random walks to generate node sequences. These pseudo words and sentences are then fed into word2vec models to obtain vector representation of graph nodes. For example, Pang and Zhang [91] create a bipartite graph which includes both user and location as nodes in the graph and thereby learn their both vector representations in the same semantic space. These methods rely on the availability of location history contextual data, which, however, is relatively sparse or incomplete with respect to individual Twitter users.

Similar to our method, CrossMap [132] also exploits co-occurrence relationships to jointly learns embeddings for location, time and text. The key differences are threefold: (1) We try to minimize the gap between embedding-based probabilities and graph-based stationary residing probabilities while CrossMap minimizes the difference between embedding-based probabilities and outdegree-based probabilities; (2) We include the features of hour-of-day and day-of-week in time while CrossMap relies on detected temporal hotspots; (3) We address both topical likeliness and spatial proximity in location-similarity search. ReAct [133] also uses co-occurrences between location, time and text in tweets to learn embeddings. However, its location is in the form of grid cells of 300mx300m. Although such a tessellation of space may simplify the processing of geospatial location, its assumption of a uniform distribution may not fit well to real-life

tweet data and is sensitive to parameters like grid cell size and sometimes noise. In contrast, our method identifies spatial clusters of tweets as locations of interest beforehand.

5.3 Preliminaries

5.3.1 Problem

Given a set of geotagged tweets $\mathcal{D}$, our goal is to learn compact vector representations of spatial, temporal and textual entities found in $\mathcal{D}$ in the same semantic space while preserving their correlations. For simplicity, a geotagged tweet $d \in \mathcal{D}$ can be seen as a tuple $\langle GPS_d, TIME_d, TEXT_d \rangle$ in which GPS_d is the geographical location (i.e., a pair of lat/long coordinates), $TIME_d$ is the publication time, and $TEXT_d$ refers to the textual content. After preprocessing, we extract essential entities in d and rewrite it as $\langle loc_d, hour_d, wday_d, word_d \rangle$ in which loc_d refers to which location of interest (e.g., which cluster) GPS_d falls in or close by, $hour_d$ and $wday_d$ are hour-of-day and day-of-week in $TIME_d$, and $word_d$ is a bag of key words extracted from $TEXT_d$. A co-occurrence relationship exists between a location and a word if they co-exist in a tweet. Actually, such co-occurrence relationships can be established for every pair of the 4 different types of entities. The objective in this chapter is to represent these entities in the form of compact vectors which preserve such co-occurrence relationships.

5.3.2 System Overview

Figure 5.2 demonstrates LeGo's overall design. After extracting the essential spatial, temporal and textual entities and building the basic graph based on the co-occurrence

between entities, LeGo may work in two modes: cross-modal search (LeGo-CM) and location-similarity search (LeGo-LS). The major difference is that the LeGo-LS adds extra edges within locations themselves and runs the graph embedding algorithm on the subgraphs simultaneously while LeGo-CM directly runs the embedding algorithm on the whole graph after performing random walk procedures.

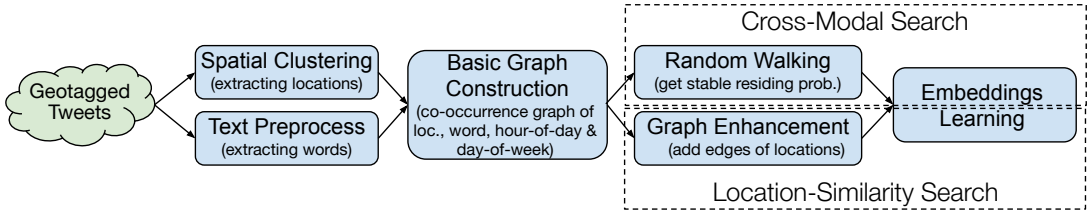


Figure 5.2: System overview.

5.4 Method

In this section, we present the workflow of LeGo. Specifically, we first discuss entity extraction from tweets and also graph construction based on co-occurrence. Next, we detail the implementation of LeGo-CM and LeGo-LS.

5.4.1 Spatial, Temporal and Textual Entity Extraction

5.4.1.1 Spatial Entity Extraction

Although tessellating space into uniform grid cells [133] may ease the processing of location information, it suffers from the sensitivity of the grid-size parameter as well as inherently assumes a uniform distribution of tweets over space, which is not practical in real-life. On the other hand, it is also impractical to consider every single spatial point

simply because of space continuity. We therefore seek to identify locations of interest as the spatial entities in our system. Such locations of interest represent the aggregation points or regions of human presence for the purpose of conducting various daily life activities.

Many clustering algorithms have been investigated and adapted to work on spatial data [48], including k-means [78], mean shift [23] and Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) [13]. In this chapter, we use mean shift to group together GPS points in tweets and thus identify locations of interest. The advantage of mean shift lies in its simplicity and generality [132] which does not assume any prior knowledge about the underlying data distribution. In order to do so, we first convert the geographical GPS coordinates from spherical to planar using the Azimuthal equidistant projection. Such a projection preserves the distance between objects, although it may distort their shapes. This distortion imposes few effects on our application.

Mean shift is a clustering algorithm that assigns circular regions of data points to clusters by iteratively shifting towards the modes. The mode can be understood as a local maxima of the density function upon the samples of data points. Formally, let z^t be the estimation of mode at iteration t, z^{t+1} can be defined as:

$$z^{t+1} = \frac{\sum_{p \in \mathcal{N}_b(z^t)} K(\frac{p-z^t}{b})p}{\sum_{p \in \mathcal{N}_b(z^t)} K(\frac{p-z^t}{b})} \tag{5.1}$$

where $\mathcal{N}_b(z^t)$ represents a set of points falling inside the circular region centered at z^t with a radius of b (also called the bandwidth in the mean shift); $K(\frac{p-z^t}{b})$ is a kernel

function that determines the weight of nearby points based on their distance to the mode estimation. In this chapter, we use a flat kernel as follows:

$$K(\frac{p-z^t}{b}) := \begin{cases} 1, & \text{if } \|\frac{p-z^t}{b}\| \leq 1 \\ 0, & \text{if } \|\frac{p-z^t}{b}\| > 1 \end{cases} \tag{5.2}$$

The mean shift continues to iterate until z^t converges to a small variance, e.g., $\|z^{t+1} - z^t\|$ goes below a small value, and thereby yields a location of interest.

5.4.1.2 Temporal and Textual Entity Extraction

Comparing to extracting locations of interest from tweets, it is quite intuitive and straightforward to extract temporal and textual entities. For example, we can directly calculate the local hour-of-day and day-of-week from the UNIX timestamp in a tweet's publication time. As for essential words in tweets, we exploit the off-the-shelf tool [96, 97] [1] to attain entities and noun phrases as textual entities [134].

5.4.2 Co-occurrence Graph Construction

Up to now, we have 4 types of entities: location, hour-of-day (hour), day-of-week (wday) and word as illustrated in Figure 5.3. These entities function as the vertices in the co-occurrence graph $G = (V, E)$. During building the co-occurrence graph G, an edge between e_{ij} between node v_i and node v_j establishes and add 1 to its weight if v_i and v_j co-exist in the tweet d. Note that the nodes of "word" have additional edges within

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

themselves to capture the co-occurrence between words in tweets, which is different from the other 3 types of entities.

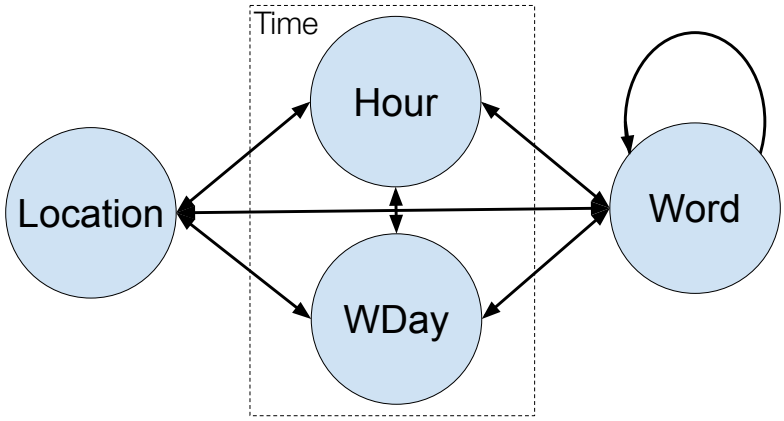


Figure 5.3: Illustration of basic co-occurrence graph.

5.4.3 Cross-Modal Search

The objective of cross-modal search is to answer this question: given an entity from one modal, which entities in other modals are most likely to be associated with it? For example, given a query location, what words are more likely to be observed from it, and what time periods in a day or which days in a week does this location tend to hold activities? Formally, given a source entity v_i^m from modal m and a target entity v_j^n from modal n, what is the possibility of observing v_j^n from v_i^m? In the following, we try to approach this possibility from two perspectives: co-occurrence graph and vectorized embeddings, and then utilize their relationship to guide the embedding learning process.

5.4.3.1 From the Perspective of Co-occurrence Graph

On the co-occurrence graph G, we resort to modifying personalized graph random walk procedures [90, 121, 79, 115] to approach the above probability, which is actually equivalent to the possibility of the random surfer residing at vertex v_j^n if he initially starts v_i^m. For convenience, let us denote this probability by $p(v_i^m \rightarrow v_j^n)$.

We first give a brief description of the random walk process and then present our modifications in order to calculate $p(v_i^m \rightarrow v_j^n)$. The mechanism behind the random walk on the graph can be briefly explained by an intuitive random surfer model where this surfer starts at a vertex and follows an outbound edge at random to visit the next vertex. The possibility of eventually residing at each vertex is then coded in the probability for the random surfer reaching that vertex, calculated as the sum of probabilities of the surfer following all possible edges towards to that vertex. Additionally, a damping factor h is usually defined to control the probability that the random surfer chooses, before start to visit next vertex, to follow an edge to reach the next vertex or simply teleports to the next vertex. This damping factor is used to avoid the random surfer being trapped in some disconnected components (if that exist) and guarantees the convergence of the random walk. In summary, suppose we have a graph $G = (V, E)$, the personalized random walk procedure can be iteratively defined as follows:

$$\mathbb{R}^{t+1} = (1-h) * \mathbf{\Pi} + h * \mathbb{R}^t \times \mathbb{M} \tag{5.3}$$

where $\mathbb{R}^t = \begin{bmatrix} r_{v_1}^t & r_{v_1}^t & \dots & r_{v_N}^t \end{bmatrix}$ is the residing possibility after iterating t times, N is the

number of vertices $N = |V|$, and each element $r^t_{v_i}$ represents the residing probability on the vertex v_i; h is the damping factor ranging from 0 to 1; $\mathbf{\Pi} = \begin{bmatrix} \pi_1 & \pi_2 & \dots & \pi_N \end{bmatrix}$ is a teleportation vector in which the element $\pi_i = 1$ if the surfer starts at v_i with the rest being zeros; and $\mathbb{M}$ is the transition probability matrix which is a $N \times N$ matrix with each element m_{ij} specifying the probability that the surfer transitions to vertex v_j from vertex v_i by following an existing edge in the graph. Typically, the transition probability m_{ij} is 0 if vertex v_i doesn't have a outbound edge to vertex v_j, and $m_{ij} = \frac{1}{|OUT_i|}$ if such an edge exists, where OUT_i denotes the set of vertices to which v_i has an outbound edge.

Since the edges in the co-occurrence G have weights, let us adjust the transition probabilities between vertices accordingly. Specifically, if w_{ij} denotes the weight of an edge $e_{ij} \in E$, then the transition probability from v_i to v_j can be calculated as [121]:

$$m_{ij} = \frac{w_{ij}}{\sum\limits_{v_k \in OUT_i} w_{ik}} \tag{5.4}$$

Finally, suppose that we now have the converged residing probabilities for graph nodes $\mathbb{R} = \begin{bmatrix} r_{v_1} & r_{v_1} & \dots & r_{v_N} \end{bmatrix}$, instead of directly using the residing probability of vertex v^n_j (i.e., $r_{v^n_j}$), we calculate its normalized variant as the $p(v^m_i \rightarrow v^n_j)$:

$$p(v^m_i \rightarrow v^n_j) = \frac{r_{v^n_j}}{\sum\limits_{v_k \in V^n} r_{v_k}} \tag{5.5}$$

where V^n denotes the set of vertices from the modal n.

5.4.3.2 From the Perspective of Vectorized Embeddings

Remember that our goal is to approach the possibility of generating the entity v_j^n from the entity v_i^m which is from a different modal. Suppose that we have the vectorized embeddings of the entities, it is relatively easy to model the objective probability using the embeddings compared to co-occurrence grahp. For example, we may use $p(v_j^n|v_i^m)$ as the objective probability, which is defined as:

$$p(v_j^n|v_i^m) = \frac{e^{\mathbf{v}_i^m \cdot \mathbf{v}_j^n}}{\sum\limits_{v_k \in V^n} e^{\mathbf{v}_i^m \cdot \mathbf{v}_k}} \tag{5.6}$$

where $\mathbf{v}$ is the vectorized embedding of entity v and V^n similarly denotes the set of entities from the modal n.

5.4.3.3 Learning Embeddings

Given the probability of $p(v_i^m \rightarrow v_j^n)$ from the co-occurrence graph and the probability of $p(v_j^n|v_i^m)$ from the initial embeddings, the goal of learning embeddings is to iteratively update values in embeddings so that $p(v_j^n|v_i^m)$ becomes more and more close to $p(v_i^m \rightarrow v_j^n)$. In so doing, the eventually computed vectorized embeddings will be able to preserve the structure information of the co-occurrence graph. We use the Kullback-Leibler divergence $KL(\cdot)$ to measure the difference between two probability distributions.

Subsequently, we define the loss function between any two modals of entities as:

$$\begin{aligned}\mathcal{L}(m,n) = &\sum_{v_i^m \in V^m} KL(p(v_i^m \rightarrow \cdot) || p(\cdot | v_i^m)) \\ &+ \sum_{v_j^n \in V^n} KL(p(v_j^n \rightarrow \cdot) || p(\cdot | v_j^n))\end{aligned} \tag{5.7}$$

Such a loss function basically means to minimize both of the distribution differences to generate one modal of entities from entities in another modal and conversely to generate another modal of entities from entities in this modal. At last, the total loss function is the sum of different $\mathcal{L}(m,n)$ with respect to all different edges types in Figure 5.3. Note that the computation of such loss functions can be solved efficiently using stochastic gradient descent and negative sampling [87, 132].

5.4.4 Location-Similarity Search

The objective of location-similarity search is to answer this question: given a location in a city, which other locations are most similar to it? Although the location embeddings learned in Section 5.4.3 may help answer this question to some extent, the basic co-occurrence graph omits certain explicit relationships within locations like spatial proximity and thereby imposes limitations to the learned embeddings. In this section, we modify the basic co-occurrence graph by supplementing two types of edges within locations and subsequently learn location embeddings for supporting similarity search.

5.4.4.1 Co-occurrence Graph Enhancement

Intuitively, two factors may exert noticeable effects on the similarity between locations: spatial proximity and topic likeliness. For example, several clothing stores tend to locate close to each other in real-life shopping malls. On the other hand, two isolated restaurants may also bear resemblance in the sense that they all provide food catering. Therefore, we enhance the basic co-occurrence graph by add two types of edges within locations to address these two factors as shown in Figure 5.4.

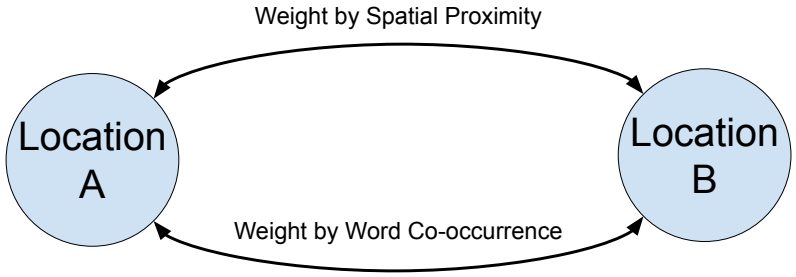


Figure 5.4: Add-on edges within locations themselves.

Next, the weights of the add-on edges are determined as follows. Suppose that we have two location vertices v_i^l and v_j^l. For spatial proximity, its weight is determined by a Gaussian attenuator, i.e., $w'_{ij} = e^{\frac{-dist(v_i^l, v_j^l)^2}{2b^2}}$, where $dist(\cdot)$ is the planar distance function and b is the bandwidth defined in the mean shift clustering algorithm in Section 5.4.1.1. As for topic likeliness, we count how many times a word co-occurrence happens between two locations. In particular, a word ω co-occurs between locations v_i^l and v_j^l if ω appears in both of their tweets. Furthermore, if such a co-occurrence happens in a shorter time, it should be considered more important. Therefore, we define the weight by word co-occurrence between v_i^l and v_j^l as: $w''_{ij} = \sum_{\omega \in \Omega_{ij}} \exp(\frac{-diff_\omega(v_i^l, v_j^l)^2}{2\sigma_{ls}^2})$, where Ω_{ij} denotes the set of common words shared by v_i^l and v_j^l, $diff_\omega(\cdot)$ measures the smallest gap in time

between the time point ω appears in v_i^l and the time point ω appears in v_j^l, and σ_{ls} is the time bandwidth. In practice, encoding all-to-all edges within locations will bring about significant computation cost if there are many locations of interest. To speedup this, we make the following two modifications. First, we combine the two types of edges plotted in Figure 5.4 into one by combining their weights as $\lambda w'_{ij} + (1-\lambda) w''_{ij}$. Second, instead of adding all-to-all edges within locations, for a location v_i^l, we only consider its nearest N_{sn} spatial neighbors to supplement such edges.

5.4.4.2 Learning Embeddings

Unlike the basic co-occurrence graph, it becomes impractical or too complex to perform the random walk procedures on the enhanced graph because it encodes two different types of edge weights. Therefore, we need a different way to compute $p(v_i^m \rightarrow v_j^n)$ on the enhanced graph. Essentially, $p(v_i^m \rightarrow v_j^n)$ reflects the possibility that the vertex v_i^m transitions to vertex v_j^n, and thereby inspires us to utilize the transition probability in Equation 5.4 for an approximation [132]. For example, we can now define $p(v_i^m \rightarrow v_j^n)$ as the normalized transition probability as follows:

$$p(v_i^m \rightarrow v_j^n) = \frac{w_{ij}}{\sum\limits_{v_k \in V^m \cap OUT_i} w_{ik}} \tag{5.8}$$

Recall that V^m is the set of vertices from modal n and OUT_i is a vertex's outgoing neighbors. With Equation 5.8, the actual embeddings learning process is similar to the one in Section 5.4.3.3 except for putting on an extra loss function to address the edges within locations themselves.

5.5 Evaluation

5.5.1 Experimental Settings

5.5.1.1 Datasets

The evaluation is performed on two sets of geotagged tweets collected from 2014-08-01 to 2014-11-30 in two corresponding cities: New York City (NYC) and Los Angeles, CA (LA) [133]. The total number of tweets, is about 1.5 million in NYC and 1.2 million in LA, respectively. We randomly take $10,000$ tweets for testing and the rest for learning the embeddings of location, words, hours-of-day and days-of-week.

5.5.1.2 Baseline Approaches

We compare with the following baseline approaches:

- TF-IDF first builds a document-term matrix between locations and words. There are various schemes for determining the value of each entry in the matrix. Here in this method, the scheme is TF-IDF. Each location is then represented by a row vector and each word is represented by a column vector. Additional zeros are padded into the smaller-size dimentionality to make row vectors and column vectors have the same size.

- SVD also builds a document-term matrix between locations and words. But unlike TF-IDF, the value of each entry in this matrix represents the number of times of each word appears in each location. The Singular Value Decomposition is then applied to this

matrix to reduce the dimentionality size of rows and columns and thereby extract them as compacted vectors of locations and words.

- Doc2Vec [63] is an extension of Word2Vec technique [87] and directly learns the vector representations of both documents and words in the same semantic space by introducing an additional document feature vector. Here, we use the Doc2Vec implementation in the gensim library [2] and choose DBOW (Distriuted Bag of Words) as the underlying model.

- ReAct [133] first discretizes space into grid cells and time into hours and subsequently learns the embeddings of grid cells, hours and words in tweets in a semi-supervised way. During its learning process, ReAct tries to optimize two objectives. First, given two attributes in location, time and words, it tries to maximally recover the third attribute. Second, given a tweet, it tries to maximally predict its category. Note that the category information of a tweet is obtained in advance if a Foursquare link is contained in that tweet. For fairness in comparison, ReAct takes the spatial clusters generated in Section 5.4.1.1 as locations in space instead of grid cells, and takes the natural integral hours for time.

- CrossMap [132] proposed two methods of learning cross-modal embeddings for space, time and texts: ReconEmbed and GraphEmbed. ReconEmbed is very similar to ReAct [133] except for only focusing the objective of attribute recovery. We therefore only compare with GraphEmbed here. Note that, CrossMap also performs hotspot detection in temporal dimension instead of directly using temporal features like hour-of-day and

[2] https://radimrehurek.com/gensim/models/doc2vec.html

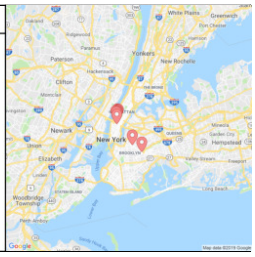
Word	Hour	WDay
ladyliberty	16	Thu
statueliberty	13	Fri
statueofliberty	12	Wed
monument	14	Tue
statue	15	Sun
statuelibrtynps	17	Sat
tripnyc	10	Mon
national	11	
trip	09	
feria	19	

(a) Spatial query: [40.690, -74.045]-Liberty Statue

Word	Hour	WDay
rockefellercenter	14	Wed
top	11	Mon
radiocitymusichall	19	Sun
flaca	09	Tue
spectacular	17	Thu
rockettes	18	Sat
topoftherock	15	Fri
rockcenternyc	20	
observationdeck	13	
nintendo	12	

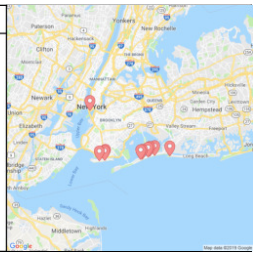
(b) Spatial query: [40.759, -73.980]-Rockefeller Center

Word	Hour	WDay
broadway	19	Thu
	13	Mon
	23	Tue
	22	Wed
	18	Fri
	14	Sun
	17	Sat
	10	
	20	
	21	



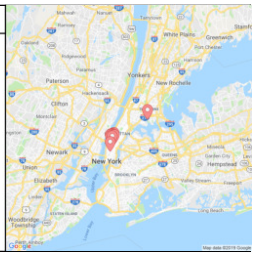
(c) Textual query: broadway.

Word	Hour	WDay
beach	14	Thu
	16	Fri
	15	Sat
	12	Wed
	18	Tue
	17	Sun
	13	Mon
	20	
	10	
	19	



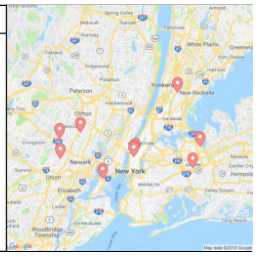
(d) Textual query: beach.

Word	Hour	WDay
like	10	Wed
get		
one		
good		
day		
new		
got		
time		
today		
last		



(e) Temporal query: hour=10, day=Wed.

Word	Hour	WDay
tonight	22	Sat
time		
night		
like		
get		
last		
saturday		
love		
good		
got		



(f) Temporal query: hour=22, day=Sat.

Figure 5.5: Examples of spatial, textual and temporal queries in NYC.

day-of-week.

By default, TF-IDF, SVD and Doc2Vec handle data of only two dimensions. We perform the following preprocessing in these methods in order to incorporate all the entities of location, words, hours-of-day and days-of-week. We treat each location as a document and its sentences comprise the tweets falling inside the location. The hour-of-day and day-of-week values extracted from posting time of each tweet are parsed as special words and appended to that tweet's bag of words.

5.5.1.3 Parameter Settings

The major parameters in LeGo are set as follows. For embedding dimension length, we set $N_{dim} = 200$. For time entities in tweets, we extract its natural integral hours-of-day and days-of-week from the publication of tweets converted to local time, i.e., $hour = \{0, 1, 2, \cdots, 21, 22, 23\}$ and $wday = \{Mon, Tue, Wed, Thu, Fri, Sat, Sun\}$, in order to reflect patterns of people's daily life in urban areas. We set the bandwidth b of mean shift[3] for clustering tweet locations to $160m$, which yields around 18,000 location clusters in NYC and 17,000 location clusters in LA. As for the random walk procedure to calculate stationary residing probabilities between vertices in the co-occurrence graph, we use a default damping factor $h = 0.8$ and run 20 iterations in all cases. In the embedding learning process, we set the number of epochs for training $N_{epoch} = 256$ and the learning rate $\alpha_{learn} = 0.02$. As for LeGo-LS for location-similarity search, we set the number of spatial neighbors $N_{sn} = 8$, the weight coefficient of spatial proximity over topic likeliness $\lambda = 0.5$, the bandwidth σ_{ls} for the time difference of a spatial word co-occurrence to be 2 months.

For comparison, all methods are tested using the same N_{dim} except for TF-IDF. Also note that TF-IDF, SVD and Doc2Vec use the same representations of location and time as LeGo. Although ReAct and CrossMap are also fed with the same form of locations, they use natural integral hours and time hotspots for time representations as in their implementation, respectively.

[3] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.MeanShift.html

5.5.2 Illustrative Cases

We select several examples of cross-modal search and location-similarity search in Figures 5.5 and 5.6, respectively.

5.5.2.1 Cross-Modal Search

We perform several case studies of cross-modal searches in NYC and present the results in Figure 5.5. Each example is described by 4 types of data with locations plotted as red GPS pin markers in the accompanying maps. Ahead of the map of GPS points are location-correlated words, hours-of-day and days-of-week. To perform the spatial, textual and temporal queries, we input the information in one dimension and retrieve the most similar information in other dimensions under the metric of vector cosine similarity, which are listed from top to bottom in the figures.

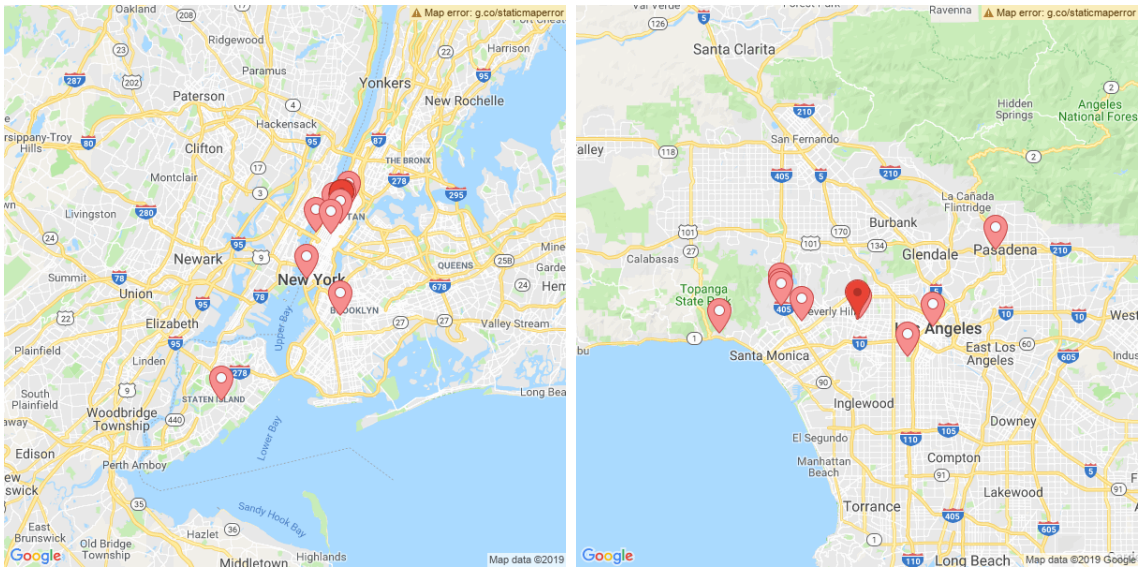
Spatial Queries. Figure 5.5a and Figure 5.5b illustrate two spatial queries in NYC. The two spatial inputs are the GPS points of two landmark buildings in NYC: the Statue of Liberty and the Rockefeller Center. The textual, temporal outputs are listed in the columns of *Word*, *Hour* and *WDay*, correspondingly. From the perspective of top retrieved words, they are very demonstrative because they exhibit close relatedness to the input location. For example, "ladyliberty", "statuliberty" and "statueofliberty" etc are just close variants of the name of the landmark building. From the perspective of top retrieved hour values, we find that the Statue of Liberty is more related to the afternoon before 4pm. This is reasonable in the sense that it is usually open to public up to 4pm. On the contrary, we find that the Rockefeller Center has a stronger relation to noon and early evening. This

is because one of its most famous sights is "top of the rock" which attracts people to go there and see the landscape of NYC during the day and witness the dusk and night view of NYC. From the perspective of top retrieved days-of-week, there seems no noticeable difference between weekdays and weekends in both cases of spatial query. Our intuitive explanation is that both of these two locations are very popular tour locations and more importantly, they are also open during the weekends. Therefore, they attract people to visit regardless of which day it is in the week.

Textual Queries. For textual queries, we give two words "broadway" and "beach" as the inputs to retrieve most related information on hours-of-day, days-of-week and locations. The results are presented in Figure 5.5c and Figure 5.5d, respectively. First, we find that the returned locations (the GPS pin marker in the map) are highly relevant to the input textual query. For example, the word "broadway" yields a set of locations in which most are concentrated around the Broadway theatre district in downtown Manhattan, while the remaining two fall on the Broadway Ave. in the Brooklyn borough of NYC. Meanwhile, in the case of "beach", we find all its top retrieved locations are geographically meaningful by falling along waterside. Second, with respect to the retrieved hours-of-day, one may notice that "broadway" are more correlated to evening time while "beach" more correlated to late afternoon. This is consistent with the Broadway show schedules and people's habits of enjoying beach time. Third, when it comes to the retrieved days-of-week, "broadway"-related activities tend to happen on weekdays, and in comparison, "beach"-related activities are more relate to late weekdays and early weekends. This provides meaningful insights to people's urban life pattern changes from weekdays to weekends: on weekdays, due to job constraints, people usually choose nearby urban locations for

entertainments, and when it is close to weekends, people may get early-release from job duties and may choose vacation resorts for enjoying the longer free time.

Temporal Queries. Because temporal features include both hour-of-day and day-of-week, we set both of their values in temporal queries. In Figure 5.5e, we set $hour = 10$ and $wday = Wed$ to retrieve the top related words and locations. Similarly, in Figure 5.5f, we set $hour = 22$ and $wday = Sat$. With respect to the top retrieved words, one can see that the words are expressing more emotional feelings (e.g., "like" and "good") rather than indicative of specific topics. This makes sense because it is not expected that all people would be involved in the same type of activities even during the same time. Even so, some words of common sense may pop up like "day" and "night", respectively. As to the top retrieved locations in temporal queries, we find the results significantly different in the case of weekday morning and weekend evening. For example, the locations for a Wednesday morning in Figure 5.5e mainly stay closely in urban downtown working zones and the locations for a Saturday evening in Figure 5.5f scatter over different resident zones. In order to distinguish which temporal feature (hour-of-day or day-of-week) is determining the distribution pattern of retrieved locations. We make one additional temporal query with $hour = 22, wday = Wed$ and the results show that the location distribution in Wednesday evening is sparser than the one in Figure 5.5e but denser than the one in Figure 5.5f. This indicates that such a location distribution pattern results from both effects of hour-of-day and day-of-week features. It is worth mentioning that this change in location pattern slightly echoes the observations in textual queries: people's urban life pattern may change when it transitions from weekdays to weekends.



(a) NYC (b) LA

Figure 5.6: Examples of location similarity queries in NYC and LA, respectively.

5.5.2.2 Location Similarity Search

We now present two examples of location similarity search that encodes both spatial proximity and topic likeness in NYC and LA, respectively in Figure 5.6. Each example features a query location (shown as the dark red solid map pin marker) and a list of similar locations (light red hollow map pin markers). The similar locations are retrieved under the metric of cosine vector similarity. In the case of NYC, the query location is a GPS point of [40.781022,-73.973197] where the American Museum of Natural History locates. The cluster of returned locations that fall close to the query location in Figure 5.6a are all hot check-in places of nearby museums such as the Metropolitan Museum of Art, the Museum of Modern Art, the Met Breuer which is a museum of modern and contemporary art and etc. Although there are 3 locations slightly far away from the input query location, they exhibit topic likeliness because they are also about museums at different places like

the 911 Memorial Museum and the Brooklyn Museum. Therefore, the locations we retrieved in Figure 5.6a are very reasonable as they exhibit both spatial proximity and topic likeliness. Furthermore, we also conduct a similar search in LA by inputting the following coordinates 34.062787,-118.361282, which is a point surrounded by a cluster of museums majorly about art. The retrieved locations are plotted on the map in Figure 5.6b. One can see that, in comparison to NYC, the retrieved locations in the case of LA are relatively farther away from its query point and distribute more dispersedly. Even so, they reveal a more significant topic likeliness: The majority of the retrieved locations in LA are about museums and moreover, are museums of art. For example, among the retrieved locations are the Getty center which has museums about art, the Norton Simon Museum which is about public art, and the Hammer Museum which is gallery with a permanent collection of historical works and special exhibits of edgy contemporary art. Such a strong similarity in topics may mitigate their relatively large distances in geography. This further demonstrates the effectiveness of our encoding both spatial proximity and topical likeliness in learning latent embeddings of locations for similarity search.

5.5.3 Quantitative Analysis

5.5.3.1 Effectiveness

We evaluate the effectiveness of different embedding methods by performing the tasks of ranking tweets with negative attributes. The task of ranking a tweet d with a negative attribute is conducted as follows: First, we define a ranking score function $f^{RS}(\cdot)$ to compute a score of d by averaging the cosine similarities between its embedding vectors

of attributes (Note that the embedding vectors of words in $word_d$ will be first averaged to get a single vector); Second, we mask one of d's attributes in $\langle loc_d, word_d, hour_d, wday_d \rangle$ and fill in a negative candidate value randomly sampled from other tweets in the dataset. We repeat this process 10 times and get 10 variants of d, denoted by $\{d'_i | i = 1 \cdots 10\}$. Finally, we compute the ranking order of $f^{RS}(d)$ in $\{f^{RS}(d'_i) | i = 1 \cdots 10\}$ in a descending order, and denote it by $\mathcal{R}_d$. Therefore, topper ranking orders indicates better embeddings of tweets. To quantify the rankings orders of testing tweets, we adopt the metric of Mean Reciprocal Rank (MRR) [133, 132], which is defined as:

$$MRR = \frac{\sum_{d \in \mathcal{D}_{Test}} \frac{1}{\mathcal{R}_d}}{|\mathcal{D}_{Test}|} \tag{5.9}$$

where $\mathcal{D}_{Test}$ represents the testing dataset of tweets. It is easy to see that higher-quality embeddings will yield larger MRR values. In our settings, we set $|\mathcal{D}_{Test}| = 10,000$ and then compute such an MRR for each of the attributes in $\langle loc_d, hour_d, wday_d, word_d \rangle$.

Cross-Modal Search The results of LeGo-CM for cross-modal search are listed

Method	NYC				LA			
	Loc	Word	Hour	WDay	Loc	Word	Hour	WDay
TF-IDF	0.275	0.274	0.279	0.280	0.277	0.279	0.283	0.286
SVD	0.402	0.321	0.321	0.321	0.350	0.317	0.341	0.342
Doc2Vec	0.448	0.491	0.342	0.345	0.469	0.523	0.338	0.336
ReAct	0.470	0.459	0.167	N/A	0.560	0.561	0.167	N/A
CrossMap	0.516	0.619	N/A	N/A	0.514	0.642	N/A	N/A
LeGo-CM	**0.589**	0.598	**0.348**	**0.348**	**0.616**	0.612	0.339	0.339

Table 5.1: Comparison results using Mean Reciprocal Rank.

in Table 5.1, and the MRR value in our method is bold if it is the highest value in the comparison results. It shows that LeGo-CM outperforms almost all baseline approaches

including TF-IDF, SVD and Doc2Vec and achieves better results than the state-of-the-art methods in most cases. In particular, a significant improvement is observed over ReAct and CrossMap with respect to the MRR values of locations. Note that the MRR values with respect to day-of-week are not reported for ReAct because this method does not use this feature. Similarly, CrossMap uses the hotspots in the temporal dimension to represent time and thus does not report the MRR values for the integral features of hour-of-day and day-of-week. In general, TF-IDF has the worst performance in most cases due to its direct use of sparse row/column vectors extracted from the document-term matrix. SVD improves over TF-IDF by performing dimentionality reduction and thereby only preserves the most essential information in the compacted row/column vectors in the document-term matrix. In comparison, Doc2Vec gets much better results on location and word by encoding them in the same latent space. We also notice that the results of ReAct are not as good as we expect. This is probably contributed by its online learning process which only addresses the most recent information happening at a location and chooses to forget the past information in a exponential time-decay manner. This also explains its low MRR values of hour-of-day. Although CrossMap achieves slightly better MRR values than our method LeGo-CM with respect to word, it has significant lower MRR values with respect to location. Two main types of difference in our implementation may contribute to this difference in the results. First, although our introduction of features like hour-of-day and day-of-week may help distinguish between similar places because different places usually exhibit slightly different temporal activity patterns, such integral time features may overfit certain topics which is not always practical because activities at the same time may even reveal great varieties. Second, our learning process focuses on

the stationary residing probabilities between nodes in the graph. This emphasis may be useful to the nodes representing locations but not to the nodes representing words. This is because location nodes do not have mutual edges connecting to each other and thereby have weak affects on each other during the random walk process.

Location Similarity Search For the completeness of the quantitative evaluation, we also list the MRR values of our method LeGo-LS for location similarity search. It is worth mentioning that these MRR values are presented not for the purpose of comparing with LeGo-CM and its baseline approaches but for the completeness of the evaluation under the metric of MRR. For example, in order to correctly recover the location attribute, the graph structure and the objective in LeGo-CM are trying to diminish the ambiguity between different locations while LeGo-LS's objective is to find similar locations and thus in some extent encourages the ambiguity between similar locations. Such a difference is also reflected in the MRR values listed in Table 5.2. In general, LeGo-LS has lower MRRs in both cities. This is particularly noticeable in the NYC dataset because its locations of interest have a very dense distribution over the downtown area with a lot of them falling closely to each other. As a result of introducing similarity weights between locations in the graph, these geographically close locations in NYC are becoming less distinguishable. In comparison, the distribution of the locations of interest in Los Angeles is more sparse. In other words, the distances between locations are larger, which makes them less vulnerable to weights of spatial proximity.

Method	NYC				LA			
	Loc	Word	Hour	WDay	Loc	Word	Hour	WDay
LeGo-LS	0.523	0.553	0.317	0.318	0.601	0.606	0.310	0.312

Table 5.2: MRR values in LeGo-LS.

5.5.3.2 Efficiency

To fairly investigate the efficiency of the learning process, we omit all the data preparation operations and only address the step of model training. The experiments are conducted on an AWS EC2 instance with 240GB memory and an Intel Xeon CPU (E5-2686 2.30GHz). In each method, we record the time spent in processing the training tweets. The results are reported on the NYC dataset as it contains relatively more tweets.

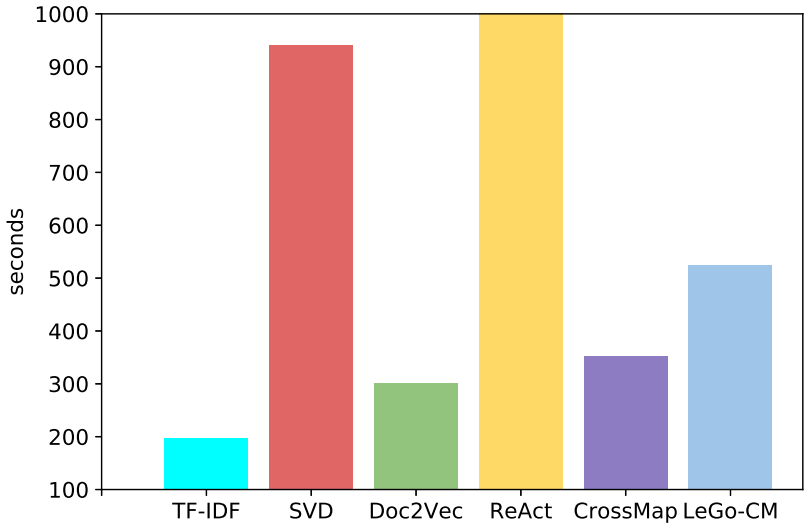


Figure 5.7: Model training time consumption.

Figure 5.7 presents the training time of different methods in seconds. It shows that TF-IDF runs the fastest because of its simplicity. Our method LeGo-CM achieves moderate efficiency comparing to CrossMap considering that we address 4 types of nodes in the graph while ReAct addresses 3 types of nodes. Also note that, introducing extra edges to connect nearby locations adds extra time consumption in LeGo-LS which usually takes about 650 seconds to finish training. The method ReAct runs the slowest because of its

small batch size which leads to frequent weight updating in its online training procedure.

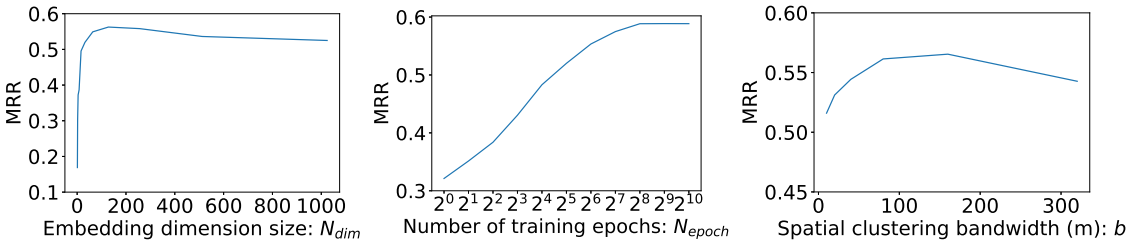
5.5.4 Parameter Sensitivity

In this section, we evaluate the sensitivity of several essential parameters in our methods. Specifically, in LeGo-CM, we study the following parameters: the embedding dimension length N_{dim}, the number of training epoch N_{epoch} and the spatial clustering bandwidth b. We also study the number of spatial neighbors N_{sn} and the weight coefficient of spatial proximity over topic likeliness λ in LeGo-LS. By default, we set $N_{dim} = 200$, $N_{epoch} = 100$, $b = 160m$, $N_{sn} = 8$, and $\lambda = 0.5$. Both of these methods are evaluated against the MRR values with respect to locations on the NYC dataset. The results are plotted in Figure 5.8 and Figure 5.9, respectively.

Figure 5.8a shows that the method improves significantly from $N_{dim} = 10$ to $N_{dim} = 200$ and soon becomes stable. A similar phenomenon is also observed in Figure 5.8b as a larger N_{epoch} value leads to a better MRR value and the curve afterwards seems to converge when N_{epoch} continues to increase. From Figure 5.8a, we can see that the MRR value increases to the maximum and then starts to drop when the spatial clustering bandwidth b increases. This meets with our expectation because a small b may divide a spatially and topically coherent cluster into many smaller ones and a large b may otherwise group spatially and topically irrelevant locations into the same cluster.

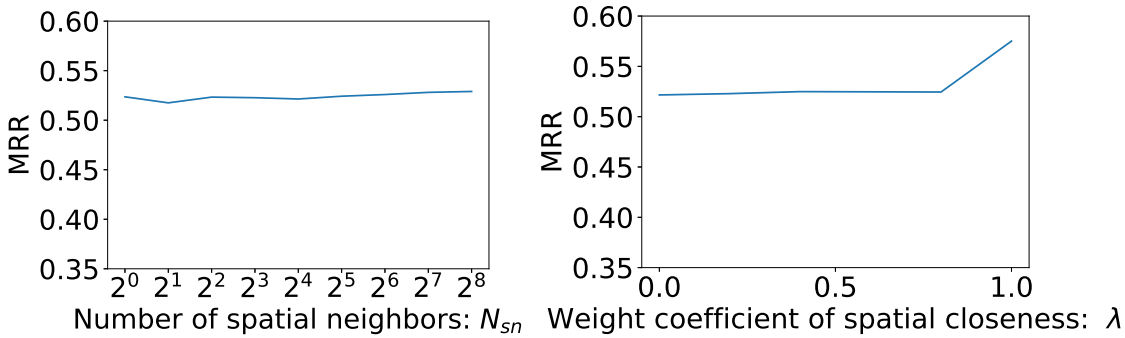
Figure 5.9 plots the location MRRs in LeGo-LS regarding the number of spatial neighbors to connect (i.e., how many nearby locations a location is going to have edges connecting to) and the weight coefficient of spatial proximity over topic likeliness. The

results show that LeGo-LS is less sensitive to these two parameters except that the location MRR significantly increases when the locations do not add on the weight resulted from topic likeliness. This makes sense because such a weight makes nearby locations look more similar if they have similar sets of words, and thus makes it hard for the model to distinguish between them.



(a) (b) (c)

Figure 5.8: Location MRRs vs. N_{dim}, N_{epoch} and b in LeGo-CM



(a) (b)

Figure 5.9: Location MRRs vs. N_{sn} and λ in LeGo-LS

5.6 Conclusions

In this chapter, we presented LeGo for learning embeddings of spatial, textual and temporal entities in geotagged tweet. In essence, LeGo has two working modes: LeGo-CM for cross-modal search and LeGo-LS for location-similarity search. Prior to the

learning process, a mean shift-based spatial clustering procedure is performed to detect locations of interest. For the time dimension, we extract hour-of-day and day-of-week as temporal entities which are consistent with people's daily-life habits and patterns. We then utilize the co-occurrence between locations, words, hours-of-day and days-of-week to build graphs for LeGo-CM and LeGo-LS, respectively. The graph in LeGo-LS is slightly different by adding edges between nearby locations whose weights are determined under the consideration of both spatial proximity and topic likeliness. Another factor which differentiates LeGo-CM from LeGo-LS is that the former learns the embeddings of graph nodes by approximating the stable residing probabilities between nodes while the latter learns the embeddings in subgraphs. The evaluation results on two selected cities show that LeGo outperforms competitive baselines in most cases, thereby showing the effectiveness of the proposed method.

For future work, learning and evaluating the embeddings of locations across different cities is an interesting topic. For example, with a query location in one city, one may search for similar locations in a different city to find similar tourism spots. Also, the exploration of utilizing the learned embeddings to classify location into different categories like "business" and "residence" is also an interesting direction.

Chapter 6: Future Work

When it comes to measuring the spatial influence of Twitter users, we have explored the role of the geographical consideration, i.e., incorporating the distance-decay function to determine the influence weights between connected Twitter users. There are, however, many aspects to further explore. First, topical considerations can be added to thereby identify influential Twitter users at specific domains. For example, after identifying top local users, additional procedures such as classification like LDA might be exploited to pick out news-related users to improve the quality of news seeders. More interestingly, topic-sensitive ranking is also another direction. For example, Weng et al. [119] propose TwitterRank to find topic-sensitive influential Twitter users by modifying the original PageRank to be aware of the topics hidden in the users' tweets. To be specific, after identifying a topic distribution for each user, the similarities between users' topic distributions are then incorporated into the transition probabilities of the random surfer walking from one user to another. Similarly, we may also utilize LDA to identity topics related to local news and events and thereby incorporate topic similarity into the transition matrix. However, the difficult part is to reasonably balance the weights of geographical distance and the weights of topic similarities. Second, most of the existing work in building an interaction graph from Twitter interactions neglect the changes in the influence of an interaction

edge over time [57, 29, 120, 14, 41, 53, 46]. However, as one aspect of being dynamic, the creation time of an interaction plays an important factor and should be accounted for. For example, Cha et al. [16] have reported that the people's influence determined by interactions might rise and fall over time and different time period might have different influential users. The influence of some athletes might substantially increases during the event of Olympic Games but fade away when they are closing. Therefore, we plan to apply a Gaussian attenuator on each of interaction edges by assuming the influence an interaction edge brings about reaches maximum at the time when it happens and fades away over time. In particular, the Gaussian parameter takes into account the difference in days between a given time point and the interaction's creation time.

When it comes to detecting local events from geotagged tweets, we have investigated how to recognize an unusual increase in the tweet volume as a signal of potential local events. Although the current method DeLLe achieves comparative performance, there are some remaining constraints. First, DeLLe currently works on city-level scope and has difficulties in scaling up over larger areas like nation-wide. For example, one may notice that the spatial ranges that we choose to perform evaluation in the city of Seattle, WA and New York City are relatively limited on the urban regions. This is due to our grid tessellation of space and the subsequent modeling tweet count data as images in the neural network models. Considering that the space grid resolution in Chapter 3 is $50m \times 50m$, directly expanding to the national level of U.S. would lead to a grid size of approximately 86000×51000, which is not easy for training with the current neural network architecture and hardware. Therefore, it is an interesting topic to look for a different space tessellation technique that can i) reduce the number of dimensions but meanwhile have a fine resolu-

tion and ii) be easily adapted into the current deep neural network models. For example, in order to reduce the memory footprint of convolution as well as increase the voxel resolution in 3D representations, Riegler et al. [94] propose OctNet, a representation for deep learning with sparse 3D data. The core idea of OctNet in dealing with large but sparse 3D data is to innovatively introduce an octree into its data structure and subsequently modify the corresponding neural network operations. Similarly, geotagged tweets may also unevenly distribute over different regions. Some areas like the urban districts may see a high density of tweets, while some areas like open waters may see a very sparse set of tweets. Therefore, resolutions should be addressed differently for these two areas. Inspired by OctNet [94], we can similarly build a quadtree representation of tweet count information, which yields bigger but fewer grid cells for areas with sparse tweets, and smaller but more grid cells for areas with dense tweets. Different from OctNet [94] which runs only one-time convolution operation, we will have to address consecutive neural network operations on a quadtree data structure because the prediction model consists of multiple layers of neural networks. Second, DeLLe currently relies on a ranking score which multiplies the weights of different factors including spatial burstiness, temporal burstiness and topical coherence. However, it is often hard to manually design a gold-standard function to accurately pick out true local event candidates. Not to mention that it also require a parameter to limit the top-K candidates. In the future, since the human evaluation yielded a groundtruth of local events, it enables us to explore other learning methods that may classify spatiotemporal unusualness into true or false local events using features like burstiness and topical coherence.

When it comes to enhancing local live tweets to enable the detection of more and

smaller local news and events, we have presented Firefly, which first identifies a large body of local people and then collects their tweets to largely increase the number of possible local tweets. The results show that in so doing, we are able to detect much more local news comparing with local news agencies and baseline approaches that only exploit geotagged tweets. There are, however, more questions to answer. First, the locality-aware keywords based clustering algorithm is not sensitive to the difference between some national and local news. For example, Firefly is more likely to report a national news as local news if there are only a very few local people discussing that national news. In the future, in order to get rid of such national news, we may monitor Twitter in general to maintain a set of national trends. And for each news we detect from a local place, we check if it meets one of the national trends and if so, decline to report it. Second, instead of the current heuristics-based parameter tuning, a learning procedure might be explored to help determine the parameter values in extracting locality-aware keywords and online clustering. Third, we have noticed that in a sole closed-domain data source like Twitter, some local news may not even get reported at all. For example, in Section 4.4.4.3, we found that some news articles published on the news agencies' websites relate to no tweets at all. It shows that there would be lot more news happening in an area than posted in Twitter. This motivates us to integrate cross-domain news sources such as Instagram, Facebook and local news websites, to further mitigate the data sparsity in the future.

Chapter 7: Conclusion

With people posting tweets to discuss real-world happenings of events, Twitter provides an invaluable source of data on local news and events. However, detecting and extracting such local news and events is still challenging due to problems like too many noisy tweets, complicated spatiotemporal context and data sparsity. We have presented several directions to deal with these challenges and explored their effectiveness in detecting local news and events.

First, we focused on finding spatially influential Twitter users on a query location based on the interactions, especially the locally influential users. Because in practice, we found locally influential Twitter users are usually good news and event seeders. In order to do this, we have built a large-scale directed interaction graph of Twitter users, and proposed a variant of a PageRank procedure to select top locally influential people. The proposed ranking procedure incorporates geographical distance to the transition matrix used for the random walking. The experiments show that by making use of the spatial locality, our proposed method Edge-Locality PageRank (ELPR) outperforms other related algorithms in finding local influential Twitter users. More importantly, we showed that the majority of such locally influential users have high credibility in outputting local place-relevant tweets, and thereby may act as potential news and event seeders.

Second, we emphasized on recognizing the unusualness in tweet volume at a local place as a signal of potential ongoing local events. In order to capture the unusual increase in the tweet count, we first try to make a prediction on the expected value of the number of tweets at a local place. This is accomplished by a novel neural network, which formulates tweet count prediction as a spatiotemporal sequence forecasting problem and design an end-to-end convolutional LSTM based network with skip connection. We then present DeLLe to detect local events in real-time from geotagged tweet streams. With the help of novel spatiotemporal tweet count prediction models, DeLLe first finds unusual locations which have aggregated unexpected number of tweets in the latest time period. For each such unusual location, DeLLe uses a ranking score to identify the ones most likely having ongoing local events by addressing the temporal burstiness, spatial burstiness and topical coherence. Furthermore, DeLLe infers an event candidate's spatiotemporal range by tracking its event-focus point. Finally, DeLLe chooses the most influential tweets to summarize local events and thereby presents succinct but yet representative descriptions. The evaluation on the city of Seattle, WA as well as a larger city of New York show that the proposed method generally outperforms competitive baseline approaches.

Third, we try to mitigate the data sparsity of local tweets in Twitter, and thereby enable the detection of more and smaller local events. A system, called Firefly, is proposed that overcomes the data sparsity and captures thousands of local stories per day from a metropolitan area (e.g., Boston). In order to deal with the infamous data sparsity of local tweets in publicly accessible Twitter data, Firefly first enhances the local live tweet stream by identifying a large body of Twitter users in an area to follow via an online geotagging procedure and thereby significantly increases the amount of tweets generated from that

area. Our experiments show that, given an area, this procedure easily feeds us almost all of its active Twitter users. With this enhanced local live tweet stream, Firefly tries to identify locality-aware keywords and further uses them to cluster tweets together to detect news. Comparing with a set of local news agencies, our system achieves the highest recalls. At the same time, our system also outperforms the baseline approach TwitterStand regarding both recall and precision. More importantly, Firefly is able to detect much more local news than the approaches that only use geotagged tweets.

Finally, we try to learn universal representations of spatial, temporal and textual entities in the geotagged tweets under the same semantic space. A methodology, called LeGo, presented to learn compact vector embeddings for locations, times and topics in tweets and at the same time, preserve their correlations. In order to do so, LeGo first extracts spatial, temporal and textual entities from tweets to build their co-occurrence graphs and then use graph embedding learning techniques to encode co-occurrence into node vectors. In essence, LeGo has two working modes: LeGo-CM for cross-modal search and LeGo-LS for location-similarity search. For spatial entities, a mean shift-based spatial clustering procedure is performed to detect locations of interest. For temporal entities, we extract hour-of-day and day-of-week as temporal entities which are consistent with people's daily-life habits and patterns. For textual entities, we extract named entities and noun phrases in tweets. We then utilize the co-occurrence between locations, words, hours-of-day and days-of-week to build graphs for LeGo-CM and LeGo-LS, respectively. The graph in LeGo-LS is formed slightly differently by adding edges between nearby locations whose weights are determined under the consideration of both spatial proximity and topic likeliness. Another factor which differentiates LeGo-CM from LeGo-LS is

that the former learns the embeddings of graph nodes by approximating the stable residing probabilities between nodes while the latter learns the embeddings in subgraphs. The results of evaluation on two selected cities show that LEGo outperforms competitive baselines in most cases, thereby shows its effectiveness.

There are still many other directions to explore as discussed in Chapter 6. For instance, we may additionally incorporate the temporal and topical considerations in determining local influential Twitter users. We may also explore a different underlying data structures (rather than the even grid) for modeling tweet count data, which could enable us to scale up the method beyond city-level to nation-wide. We may also try to integrate data sources from different domains and platforms like Instagram, Facebook and online local newspapers to further mitigate the data sparsity of local news and events. We leave these questions for future work.

Appendix 8: Top 70 Influentials on Boston using Different Methods

Table 8.1: Top 70 influential Twitter users (in various categories) on Boston using *InD*, *LocInd*, *PR* and *ELPR*.

Edge Locality			
InD	*LocInd*	*PR*	*ELPR*
patriots *sp.*	patriots *sp.*	patriots *sp.*	patriots *sp.*
crazyfightz *mi.*	onlyinbos *ne.*	oitnb *co.*	bostonglobe *ne.*
drjillstein *po.*	bostonglobe *ne.*	johncena *sp.*	onlyinbos *ne.*
diaryforcrush *mi.*	redsox *sp.*	bostonglobe *ne.*	redsox *sp.*
johncena *sp.*	stoolpresidente *re.*	redsox *sp.*	nhlbruins *sp.*
twichiste *mi.*	nhlbruins *sp.*	mls *sp.*	stoolpresidente *re.*
redsox *sp.*	mbta *co.*	mittromney *po.*	marty_walsh *po.*
oitnb *co.*	celtics *sp.*	harvardbiz *mi.*	mbta *co.*
harvardbiz *mi.*	marty_walsh *po.*	shareaholic *co.*	celtics *sp.*
shareaholic *co.*	universalhub *ne.*	drjillstein *po.*	davidortiz *sp.*
valaafshar *re.*	jared_carrabis *re.*	harvard *un.*	gillettestadium *sp.*
bostonglobe *ne.*	gillettestadium *sp.*	diaryforcrush *mi.*	edelman11 *sp.*

mittromney *po.*	smackhigh *mi.*	jorgeramosnews *re.*	massgovernor *po.*
mls *sp.*	boston25 *ne.*	celtics *sp.*	universalhub *ne.*
jorgeramosnews *re.*	feitsbarstool *re.*	crazyfightz *mi.*	crazyfightz *mi.*
elizabethforma *po.*	edelman11 *sp.*	valaafshar *re.*	jared_carrabis *re.*
udaqueness *mi.*	davidortiz *sp.*	draftkings *co.*	nesn *ne.*
invictossomos *ne.*	massgovernor *po.*	aly_raisman *sp.*	harvard *un.*
quickest_rts *mi.*	wcvb *ne.*	davidortiz *sp.*	bostonpolice *go.*
rapfavorites *mi.*	bostondotcom *ne.*	elizabethforma *po.*	smackhigh *mi.*
celtics *sp.*	nesn *ne.*	mit *un.*	boston25 *ne.*
videodubs *mi.*	7news *ne.*	banks *mi.*	feitsbarstool *re.*
onlyinbos *ne.*	bostontweet *ne.*	bose *co.*	mit *un.*
unapologetiicb *mi.*	bostonherald *ne.*	twichiste *mi.*	tdgarden *sp.*
asamjulian *mi.*	bostinno *ne.*	nhlbruins *sp.*	bostonmarathon *sp.*
sopandeb *re.*	bostonpolice *go.*	runkeeper *co.*	bostonherald *ne.*
banks *mi.*	cbsboston *ne.*	newbalance *co.*	cityofboston *go.*
macjohnathan *mi.*	necn *ne.*	hubspot *co.*	bostondotcom *ne.*
13reasonslife *mi.*	massstatepolice *go.*	stoolpresidente *re.*	wcvb *ne.*
stoolpresidente *re.*	cityofboston *go.*	brgsjks *mi.*	bostontweet *ne.*
josemanaio *mi.*	wbur *ne.*	edelman11 *sp.*	cbsboston *ne.*
advil *mi.*	mikereiss *re.*	usagym *or.*	bostinno *ne.*
heartlle *mi.*	projo *ne.*	13reasonslife *mi.*	massstatepolice *go.*

hannahkauthor *au.*	jeffphowe *re.*	dimperachi *mi.*	7news *ne.*
hubspot *co.*	harvard *un.*	onlyinbos *ne.*	wbur *ne.*
nhlbruins *sp.*	mit *un.*	correction *mi.*	985thesportshub *ne.*
sneakershouts *co.*	benvolin *re.*	generalelectric *co.*	harvardbiz *mi.*
draftkings *co.*	kirkandcallahan *ho.*	videodubs *mi.*	hubspot *co.*
brucevh *au.*	bostonmagazine *ne.*	unbellieveable *mi.*	draftkings *co.*
iamtidora *mi.*	harvardbiz *mi.*	sopandeb *re.*	nerevolution *sp.*
davidortiz *sp.*	csnne *ne.*	invictossomos *ne.*	necn *ne.*
radiofreetom *au.*	drjillstein *po.*	thisispvris *co.*	northeastern *un.*
yourgurljordan *mi.*	985thesportshub *ne.*	gillettestadium *sp.*	bostonmagazine *ne.*
shenanigansen *co.*	mbta_cr *co.*	quickest_rts *mi.*	mbta_cr *co.*
unbellieveable *mi.*	peteblackburn *re.*	gillette *co.*	csnne *ne.*
harvard *un.*	hubspot *co.*	unapologetiicb *mi.*	mikereiss *re.*
runkeeper *co.*	justamasshole *mi.*	therealswizzz *mu.*	jeffphowe *re.*
kendralust *mi.*	michaelfhurley *re.*	aerosmith *mu.*	oitnb *co.*
albertbreer *re.*	elizabethforma *po.*	sasakiasahi *mi.*	brighamwomens *ne.*
gillettestadium *sp.*	nwsboston *go.*	skylarkeleven *mu.*	bostonschools *ed.*
iosernigga *mi.*	bossportsextra *ne.*	cafcofficial *sp.*	charliebakerma *po.*
edelman11 *sp.*	charliebakerma *po.*	tangurls *mi.*	mittromney *po.*
irelatewords *mi.*	gerrycallahan *re.*	goodmanespn *re.*	drjillstein *po.*
mredtrain *mi.*	bostonheraldhs *re.*	albertbreer *re.*	bostonchildrens *ho.*

stephencozzone *co.*	iamjamesstewart *sp.*	newyorkredbulls *sp.*	massago *po.*
clintsmithiii *au.*	tdgarden *sp.*	squareenix *mi.*	newbalance *co.*
spacekatgal *po.*	albertbreer *re.*	cinedatabase *mi.*	nwsboston *go.*
aly_raisman *sp.*	tomecurran *re.*	advil *mi.*	massdot *go.*
tenser *mi.*	massdot *go.*	bostonmarathon *sp.*	aly_raisman *sp.*
1dwwinfo *co.*	johndennisweei *ho.*	apat246 *re.*	bostonfire *go.*
sumasarabeboys *mi.*	bigjimmurray *ho.*	marty_walsh *po.*	elizabethforma *po.*
thisispvris *co.*	bostonschools *ed.*	girlcrushbabes *mi.*	homesforourtrps *or.*
kiiojake *mi.*	nepd_loyko *ed.*	staples *co.*	kirkandcallahan *ho.*
jared_carrabis *re.*	cousinstizz *co.*	rapfavorites *mi.*	teambaa *or.*
thekitchensheat *mi.*	tgsports *ne.*	katienolan *ho.*	dannyamendola *sp.*
mayberrykush *mi.*	bosbizjournal *ne.*	spacekatgal *po.*	al_horford *sp.*
feitsbarstool *re.*	tonymassarotti *re.*	fieldyates *re.*	iamjamesstewart *sp.*
mit *un.*	northeastern *un.*	showtimepettis *sp.*	benvolin *re.*
goodmanespn *re.*	massago *po.*	reebokclassics *co.*	cousinstizz *co.*

Table 8.2: Top 70 influential Twitter users (in various categories) on Boston using $iFol-l_q$, $SVPR$ and GPR.

Source Vertex Locality		Hybrid
youtube *co.*-	youtube *co.*-	patriots *so.*+
realdonaldtrump *po.*-	realdonaldtrump *po.*-	youtube *co.*-
patriots *so.*+	patriots *so.*+	bostonglobe *ne.*+

girlposts *mi.*-	bostonglobe *ne.*+	onlyinbos *ne.*+
hillaryclinton *po.*-	onlyinbos *ne.*+	redsox *so.*+
sportscenter *so.*-	girlposts *mi.*-	realdonaldtrump *po.*-
onlyinbos *ne.*+	hillaryclinton *po.*-	wshhfans *mi.*-
bostonglobe *ne.*+	sportscenter *so.*-	nhlbruins *so.*+
sexualgif *mi.*-	redsox *so.*+	mbta *co.*+
sincerelytumblr *mi.*-	nytimes *ne.*-	hillaryclinton *po.*-
dory *mi.*-	causewereguys *mi.*-	stoolpresidente *re.*+
wshhfans *mi.*-	mbta *co.*+	marty_walsh *po.*+
redsox *so.*+	stoolpresidente *re.*+	celtics *so.*+
nytimes *ne.*-	sincerelytumblr *mi.*-	sportscenter *so.*-
causewereguys *mi.*-	untappd *mi.*-	relatablequote *mi.*-
tweetlikeagiri *mi.*-	tweetlikeagiri *mi.*-	girlposts *mi.*-
nfl *so.*-	barstoolsports *so.*-	hornyfacts *mi.*-
freddyamazin *mi.*-	dory *mi.*-	nytimes *ne.*-
cnn *ne.*-	berniesanders *po.*-	davidortiz *so.*+
berniesanders *po.*-	nfl *so.*-	universalhub *ne.*+
sodamntrue *mi.*-	nhlbruins *so.*+	entwithbeth *mi.*-
potus44 *po.*-	wshhfans *mi.*-	causewereguys *mi.*-
barstoolsports *so.*-	cnn *ne.*-	dory *mi.*-
bleacherreport *so.*-	weloverobdyrdek *mi.*-	massgovernor *po.*+

weloverobdyrdek *mi.*-	marty_walsh *po.*+	uberfacts *mi.*-
stoolpresidente *re.*+	sexualgif *mi.*-	funnyvines *mi.*-
woridstarcomedy *mi.*-	sodamntrue *mi.*-	sexualgif *mi.*-
babyanimalpics *mi.*-	forbes *co.*-	berniesanders *po.*-
justinbieber *mu.*-	potus44 *po.*-	potus44 *po.*-
espn *ne.*-	espn *ne.*-	jared_carrabis *re.*+
nhlbruins *so.*+	universalhub *ne.*+	heyifeellike *mi.*-
foxnews *ne.*-	bleacherreport *so.*-	tweetlikeagiri *mi.*-
meninisttweet *co.*-	freddyamazin *mi.*-	boston25 *ne.*+
thetumblrposts *mi.*-	cuteemergency *mi.*-	espn *ne.*-
kardashianreact *mi.*-	celtics *so.*+	bostonpolice *go.*+
mbta *co.*+	meninisttweet *co.*-	nesn *ne.*+
girlideas *mi.*-	babyanimalpics *mi.*-	harvard *univ.*+
reiatabie *mi.*-	robgronkowski *so.*-	sincerelytumblr *mi.*-
cuteemergency *mi.*-	linkedin *co.*-	feitsbarstool *re.*+
celtics *so.*+	massgovernor *po.*+	sodamntrue *mi.*-
marty_walsh *po.*+	etsy *co.*-	thetumblrposts *mi.*-
craziestsex *mi.*-	buzzfeed *ne.*-	tdgarden *so.*+
relatablequote *mi.*-	taylorswift13 *mu.*-	wsj *ne.*-
forbes *co.*-	wsj *ne.*-	mit *univ.*+
commonwhitegiri *mi.*-	commonwhitegiri *mi.*-	cityofboston *go.*+

universalhub *ne.+*	justinbieber *mu.-*	bostondotcom *ne.+*
zaynmalik *mu.-*	kardashianreact *mi.-*	smackhigh *mi.+*
taylorswift13 *mu.-*	edelman11 *so.+*	thefunnyteens *mi.-*
harry_styles *mu.-*	scottzolak *host-*	weloverobdyrdek *mi.-*
sensanders *po.-*	toucherandrich *mi.-*	bostinno *ne.+*
worldstar *mi.-*	sensanders *po.-*	_collegehumor_ *mi.-*
drake *mu.-*	foxnews *ne.-*	bostontweet *ne.+*
barackobama *po.-*	reiatabie *mi.-*	gillettestadium *so.+*
hornyfacts *mi.-*	uberfacts *mi.-*	wcvb *ne.+*
buzzfeed *ne.-*	tombradysego *mi.-*	edelman11 *so.+*
uberfacts *mi.-*	barstoolbigcat *so.-*	7news *ne.+*
jared_carrabis *re.+*	boston25 *ne.+*	bostonherald *ne.+*
michael5sos *mu.-*	woridstarcomedy *mi.-*	babyanimalpics *mi.-*
robgronkowski *so.-*	gillettestadium *so.+*	powerful *mi.-*
wsj *ne.-*	relatablequote *mi.-*	cbsboston *ne.+*
niallofficial *mu.-*	thefunnyvine *mi.-*	forbes *co.-*
thefunnyvine *mi.-*	hornyfacts *mi.-*	harvardbiz *mi.+*
washingtonpost *ne.-*	barackobama *po.-*	massstatepolice *go.+*
5sos *band-*	wcvb *ne.+*	freddyamazin *mi.-*
twitter *co.-*	feitsbarstool *re.+*	cnn *ne.-*
louis_tomlinson *mu.-*	chanelpuke *mi.-*	bostonmarathon *so.+*

ap *ne.*-	sharethis *mi.*-	wbur *ne.*+
chanelpuke *mi.*-	girlideas *mi.*-	985thesportshub *ne.*+
camerondallas *mu.*-	davidortiz *so.*+	bieberbonerz *mi.*-

In Table 8.1 and Table 8.2, we list the top 70 influential Twitter users on the city of Boston and manually annotate them into the following categories:

- *au.* → Author;
- *co.* → Company;
- *ed.* → Education;
- *go.* → Government;
- *ho.* → Host or Hospital;
- *mi* .→ Miscellaneous;
- *mu.* → Musician;
- *ne.* → News;
- *or.* → Organization;
- *po.* → Politician or Police;
- *un.* → University;
- *re.* → Reporter;

- *sp.* → Sports;

In addition, a suffix "-" means that user is outside Boston, and "+" (or null) inside Boston. By examining the categories of the top influential Twitter users, one may find out that they may act as a reliable news sources. For example, take the top locally influential Twitter users identified by method *ELPR*, the majority of them are official Twitter accounts on famous sports teams, newspapers, reporters, politicians and government agencies etc. They certainly act as high-quality sources of local news and events, and thus provide an information window to let outsiders to learn of the local happenings.

Bibliography

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015.

[2] Hamed Abdelhaq. "Localized Events in Social Media Streams: Detection, Tracking, and Recommendation". PhD thesis. Heidelberg University, Nov. 2015.

[3] Hamed Abdelhaq and Michael Gertz. "On the Locality of Keywords in Twitter Streams". In: *ACM IWGS '14*.

[4] Hamed Abdelhaq, Michael Gertz, and Christian Sengstock. "Spatio-temporal Characteristics of Bursty Words in Twitter Streams". In: *SIGSPATIAL '13*.

[5] Hamed Abdelhaq, Christian Sengstock, and Michael Gertz. "EvenTweet: Online Localized Event Detection from Twitter". In: *PVLDB '13*.

[6] M-Dyaa Albakour, Craig Macdonald, and Iadh Ounis. "Identifying Local Events by Using Microblogs As Social Sensors". In: *OAIR '13*.

[7] Isabel Anger and Christian Kittl. "Measuring Influence on Twitter". In: *i-KNOW '11*.

[8] Farzindar Atefeh and Wael Khreich. "A Survey of Techniques for Event Detection in Twitter". In: *Comput. Intell.* 31.1 (Feb. 2015), pp. 132–164.

[9] Jie Bao, Yu Zheng, and Mohamed F. Mokbel. "Location-based and Preference-aware Recommendation Using Sparse Geo-social Networking Data". In: *SIGSPATIAL '12*.

[10] Alexander Boettcher and Dongman Lee. "EventRadar: A Real-Time Local Event Detection Scheme Using Twitter Stream". In: *GreenCom '12*.

[11] Panagiotis Bouros, Dimitris Sacharidis, and Nikos Bikakis. "Regionally Influential Users in Location-aware Social Networks". In: *SIGSPATIAL '14*.

[12] Ceren Budak, Theodore Georgiou, Divyakant Agrawal, and Amr El Abbadi. "GeoScope: Online Detection of Geo-correlated Information Trends in Social Networks". In: *PVLDB '13*.

[13] Ricardo J. G. B. Campello, Davoud Moulavi, and Joerg Sander. "Density-Based Clustering Based on Hierarchical Density Estimates". In: *PAKDD '13*.

[14] Amparo E Cano, Suvodeep Mazumdar, and Fabio Ciravegna. “Social influence analysis in microblogging platforms–a topic-sensitive based approach”. In: *Semantic Web* 5.5 (2014), pp. 357–372.

[15] Michelangelo Ceci, Roberto Corizzo, Fabio Fumarola, Donato Malerba, and Aleksandra Rashkovska. “Predictive Modeling of PV Energy Production: How to Set Up the Learning Task for a Better Prediction?” In: *IEEE Transactions on Industrial Informatics* 13.3 (June 2017), pp. 956–966.

[16] Meeyoung Cha, Hamed Haddadi, Fabricio Benevenuto, and Krishna Gummadi. “Measuring user influence in twitter: The million follower fallacy.” In: *ICWSM ’10*.

[17] Junghoon Chae, Dennis Thom, Harald Bosch, Yun Jang, Ross Maciejewski, David S. Ebert, and Thomas Ertl. “Spatiotemporal social media analytics for abnormal event detection and examination using seasonal-trend decomposition”. In: *VAST ’12*.

[18] Feng Chen and Daniel B. Neill. “Non-parametric Scan Statistics for Event Detection and Forecasting in Heterogeneous Social Media Graphs”. In: *KDD ’14*.

[19] Yan Chen, Jichang Zhao, Xia Hu, Xiaoming Zhang, Zhoujun Li, and Tat-Seng Chua. “From Interest to Function: Location Estimation in Social Media”. In: *AAAI ’13*.

[20] Zhiyuan Cheng, James Caverlee, and Kyumin Lee. “You Are Where You Tweet: A Content-based Approach to Geo-locating Twitter Users”. In: *CIKM ’10*.

[21] Wei-Chien-Benny Chin and Tzai-Hung Wen. “Geographically Modified PageRank Algorithms: Identifying the Spatial Concentration of Human Movement in a Geospatial Network”. In: *PLOS ONE* 10.10 (Oct. 2015), pp. 1–23.

[22] Francois Chollet et al. *Keras*. https://github.com/fchollet/keras. 2015.

[23] Dorin Comaniciu and Peter Meer. “Mean shift: a robust approach toward feature space analysis”. In: *IEEE TPAMI* 24.5 (May 2002), pp. 603–619.

[24] Ryan Compton, David Jurgens, and David Allen. “Geotagging One Hundred Million Twitter Accounts with Total Variation Minimization”. In: *BigData ’14*.

[25] Nilesh Dalvi, Ravi Kumar, and Bo Pang. “Object Matching in Tweets with Spatial Models”. In: *WSDM ’12*.

[26] Ankur Dave. “IndexedRDD: Efficient Fine-Grained Updates for RDDs”. In: *Spark Summit ’15*.

[27] Clodoveu A. Davis Jr., Gisele L. Pappa, Diogo Rennó Rocha de Oliveira, and Filipe de L. Arcanjo. “Inferring the Location of Twitter Messages Based on User Relationships”. In: *Trans. GIS* 15.6 (Nov. 2011), pp. 735–751.

[28] Bhuwan Dhingra, Zhong Zhou, Dylan Fitzpatrick, Michael Muehl, and William Cohen. “Tweet2Vec: Character-Based Distributed Representations for Social Media”. In: *ACL ’16*.

[29] Zhaoyun Ding, Jia Yan, Zhou Bin, and Han Yi. “Mining topical influencers based on the multi-relational network in micro-blogging sites”. In: *China Comm.* 10.1 (Jan. 2013), pp. 93–104.

[30] Maximilian Franzke, Janina Bleicher, and Andreas Züfle. “Finding Influencers in Temporal Social Networks Using Intervention Analysis”. In: *ADC ’16*.

[31] Daniel Gayo-Avello. “Nepotistic relationships in Twitter and their impact on rank prestige algorithms”. In: *Information Processing & Management* 49.6 (June 2013), pp. 1250 –1280.

[32] Nick Gramsky and Hanan Samet. “Seeder Finder: Identifying Additional Needles in the Twitter Haystack”. In: *LBSN ’13*.

[33] Alex Graves. “Generating Sequences With Recurrent Neural Networks”. In: *CoRR ’14* abs/1308.0850 (June 2013).

[34] Behnam Hajian and Tony White. “Modelling Influence in a Social Network: Metrics and Evaluation”. In: *PASSAT ’11 / SocialCom ’11*.

[35] Bo Han, Paul Cook, and Timothy Baldwin. “Text-based Twitter User Geolocation Prediction”. In: *J. AIR* 49.1 (Jan. 2014), pp. 451–500.

[36] Richard A. Harshman. “Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-modal factor analysis”. In: *UCLA Working Papers in Phonetics* 16 (1970), pp. 1–84.

[37] Taher H. Haveliwala. “Topic-sensitive PageRank”. In: *WWW ’02*.

[38] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep Residual Learning for Image Recognition”. In: *CVPR ’16*.

[39] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Identity Mappings in Deep Residual Networks”. In: *ECCV ’16*.

[40] Qi He, Kuiyu Chang, and Ee-Peng Lim. “Analyzing Feature Trajectories for Event Detection”. In: *SIGIR ’07*.

[41] Jonathan Herzig, Yosi Mass, and Haggai Roitman. “An Author-reader Influence Model for Detecting Topic-based Influencers in Social Media”. In: *HT ’14*.

[42] S.L. Ho and Min Xie. “The use of ARIMA models for reliability forecasting and analysis”. In: *Computers & Industrial Engineering* 35.1 (Oct. 1998), pp. 213 – 216.

[43] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780.

[44] Liangjie Hong, Amr Ahmed, Siva Gurumurthy, Alexander Smola, and Kostas Tsioutsiouliklis. “Discovering Geographical Topics in the Twitter Stream”. In: *WWW ’12*.

[45] Juan Hu, Yi Fang, and Archana Godavarthy. “Topical Authority Propagation on Microblogs”. In: *CIKM ’13*.

[46] Lamjed Ben Jabeur, Lynda Tamine, and Mohand Boughanem. “Active Microbloggers: Identifying Influencers, Leaders and Discussers in Microblogging Networks”. In: *SPIRE ’12*.

[47] Alan Jackoway, Hanan Samet, and Jagan Sankaranarayanan. “Identification of Live News Events Using Twitter”. In: *LBSN ’11*.

[48] Sage Jenson, Majerle Reeves, Marcello Tomasini, and Ronaldo Menezes. “Mining location information from users’ spatio-temporal data”. In: *SmartWorld ’17*.

[49] Christopher Jonathan, Amr Magdy, Mohamed F. Mokbel, and Albert Jonathan. “GARNET: A holistic system approach for trending queries in microblogs”. In: *ICDE ’16*.

[50] Krishna Y. Kamath, James Caverlee, Kyumin Lee, and Zhiyuan Cheng. “Spatio-temporal Dynamics of Online Memes: A Study of Geo-tagged Tweets”. In: *WWW ’13*.

[51] Wei Kang, Anthony K.H. Tung, Feng Zhao, and Xinyu Li. “Interactive hierarchical tag clouds for summarizing spatiotemporal social contents”. In: *ICDE ’14*.

[52] Magdalini Kardara, George Papadakis, Athanasios Papaoikonomou, Konstantinos Tserpes, and Theodora Varvarigou. “Large-scale evaluation framework for local influence theories in Twitter”. In: *Inf. Process. Manage.* 51.1 (Jan. 2015), pp. 226 –252.

[53] Georgios Katsimpras, Dimitrios Vogiatzis, and Georgios Paliouras. “Determining Influential Users with Supervised Random Walks”. In: *WWW ’15 Companion*.

[54] Mayank Kejriwal and Pedro Szekely. “Neural Embeddings for Populated Geonames Locations”. In: *ISWC ’17*.

[55] David Kempe, Jon Kleinberg, and Éva Tardos. “Maximizing the Spread of Influence Through a Social Network”. In: *KDD ’03*.

[56] Maurice George Kendall. “A New Measure of Rank Correlation”. In: *Biometrika* 30.1/2 (June 1938), pp. 81–93.

[57] Alexy Khrabrov and George Cybenko. “Discovering Influence in Communication Networks Using Dynamic Graph Analysis”. In: *SocCom ’10*.

[58] John Krumm and Eric Horvitz. “Eyewitness: Identifying Local Events via Space-time Signals in Twitter Feeds”. In: *SIGSPATIAL ’15*.

[59] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. “What is Twitter, a Social Network or a News Media?” In: *WWW ’10*.

[60] Elizabeth Kwan, Pei-Ling Hsu, Jheng-He Liang, and Yi-Shin Chen. “Event Identification for Social Streams Using Keyword-based Evolving Graph Sequences”. In: *ASONAM ’13*.

[61] Thomas K Landauer, Peter W. Foltz, and Darrell Laham. “An introduction to latent semantic analysis”. In: *Discourse processes* 25 (1998), pp. 259–284.

[62] Theodoros Lappas, Marcos R. Vieira, Dimitrios Gunopulos, and Vassilis J. Tsotras. “On the Spatiotemporal Burstiness of Terms”. In: *PVLDB ’12*.

[63] Quoc Le and Tomas Mikolov. “Distributed Representations of Sentences and Documents”. In: *ICML’14*.

[64] Alex Leavitt, Evan Burchard, David Fisher, and Sam Gilbert. *The Influentials: New Approaches for Analyzing Influence on Twitter*. 2009. URL: http://www.webecologyproject.org/wp-content/uploads/2009/09/influence-report-final.pdf.

[65] Yann LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. “Efficient BackProp”. In: *Neural Networks: Tricks of the Trade 1998*.

[66] Ryong Lee and Kazutoshi Sumiya. “Measuring Geographical Regularities of Crowd Behaviors for Twitter-based Geo-social Event Detection”. In: *LBSN ’10*.

[67] Guoliang Li, Shuo Chen, Jianhua Feng, Kian-Lee Tan, and Wen-syan Li. “Efficient Location-aware Influence Maximization”. In: *SIGMOD ’14*.

[68] Guoliang Li, Jun Hu, Jianhua Feng, and Kian-Lee Tan. “Effective location identification from microblogs”. In: *ICDE ’14*.

[69] Hao Li, Zheng Xu, Gavin Taylor, and Tom Goldstein. “Visualizing the Loss Landscape of Neural Nets”. In: *CoRR* abs/1712.09913 (2017).

[70] Quanzhi Li, Armineh Nourbakhsh, Sameena Shah, and Xiaomo Liu. “Real-Time Novel Event Detection from Social Media”. In: *ICDE ’17*.

[71] Rui Li, Kin Hou Lei, Ravi Khadiwala, and Kevin Chen-Chuan Chang. “TEDAS: A Twitter-based Event Detection and Analysis System”. In: *ICDE ’12*.

[72] Michael D. Lieberman and Hanan Samet. “Adaptive Context Features for Toponym Resolution in Streaming News”. In: *SIGIR ’12*.

[73] Michael D. Lieberman and Hanan Samet. “Multifaceted Toponym Recognition for Streaming News”. In: *SIGIR ’11*.

[74] Shu-Lan Lin, Hong-Qiong Huang, Da-Qi Zhu, and Tian-Zhen Wang. “The application of space-time ARIMA model on traffic flow forecasting”. In: *ICMLC ’09*.

[75] Haibin Liu, Bo Luo, and Dongwon Lee. “Location Type Classification Using Tweet Content”. In: *ICMLA ’12*.

[76] Zhi Liu, Yan Huang, and Joshua R. Trampier. “LEDS: Local Event Discovery and Summarization from Tweets”. In: *SIGSPATIAL ’16*.

[77] Zhi Liu, Yan Huang, and Joshua R. Trampier. “Spatiotemporal Topic Association Detection on Tweets”. In: *SIGSPATIAL ’16*.

[78] Stuart P. Lloyd. “Least squares quantization in PCM”. In: *IEEE Trans. on Information Theory* 28.2 (Mar. 1982), pp. 129–137.

[79] Peter Lofgren, Siddhartha Banerjee, and Ashish Goel. “Personalized PageRank Estimation and Search: A Bidirectional Approach”. In: *WSDM ’16*.

[80] Amr Magdy, Ahmed M. Aly, Mohamed F. Mokbel, Sameh Elnikety, Yuxiong He, Suman Nath, and Walid G. Aref. “GeoTrend: Spatial Trending Queries on Real-time Microblogs”. In: *SIGSPATIAL ’16*.

[81] Amr Magdy, Mohamed F. Mokbel, Sameh Elnikety, Suman Nath, and Yuxiong He. “Mercury: A Memory-Constrained Spatio-temporal Real-time Search on Microblogs”. In: *ICDE ’14*.

[82] Jalal Mahmud, Jeffrey Nichols, and Clemens Drews. “Home Location Identification of Twitter Users”. In: *ACM TIST* 5.3 (July 2014), 47:1–47:21.

[83] Adam Marcus, Michael S. Bernstein, Osama Badar, David R. Karger, Samuel Madden, and Robert C. Miller. “Twitinfo: Aggregating and Visualizing Microblogs for Event Exploration”. In: *CHI ’11*.

[84] Michael Mathioudakis and Nick Koudas. “TwitterMonitor: Trend Detection over the Twitter Stream”. In: *SIGMOD ’10*.

[85] Andrew J. McMinn, Yashar Moshfeghi, and Joemon M. Jose. “Building a Large-scale Corpus for Evaluating Event Detection on Twitter”. In: *CIKM ’13*.

[86] Kevin Mets. *Learning meaningful location embeddings from unlabeled visits*. 2018. URL: https://www.sentiance.com/2018/01/29/unlabeled-visits/#Location_Profiling.

[87] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. “Distributed Representations of Words and Phrases and their Compositionality”. In: *NIPS ’13*.

[88] Diana Mok, Barry Wellman, and Juan Carrasco. “Does Distance Matter in the Age of the Internet?” In: *Urban Studies* 47.13 (Nov. 2010), pp. 2747–2783.

[89] Lucas A. Overbey, Benjamin Greco, Christopher Paribello, and Terresa Jackson. “Structure and prominence in Twitter networks centered on contentious politics”. In: *Social Network Analysis and Mining* 3.4 (Sept. 2013), pp. 1351–1378.

[90] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. *The PageRank Citation Ranking: Bringing Order to the Web*. Technical Report. Stanford InfoLab, 1999.

[91] Jun Pang and Yang Zhang. “DeepCity: A Feature Learning Framework for Mining Location Check-ins”. In: *ICWSM ’16*.

[92] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. “DeepWalk: Online Learning of Social Representations”. In: *KDD ’14*.

[93] Mauricio Quezada, Vanessa Peña Araya, and Barbara Poblete. “Location-Aware Model for News Events in Social Media”. In: *SIGIR ’15*.

[94] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. “OctNet: Learning Deep 3D Representations at High Resolutions”. In: *CVPR ’17*.

[95] Fabián Riquelme and Pablo González-Cantergiani. “Measuring user influence on Twitter: A survey”. In: *Information Processing & Management* 52.5 (Sept. 2016), pp. 949 –975.

[96] Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. “Named Entity Recognition in Tweets: An Experimental Study”. In: *EMNLP ’11*.

[97] Alan Ritter, Mausam, Oren Etzioni, and Sam Clark. “Open Domain Event Extraction from Twitter”. In: *KDD ’12*.

[98] Stephen Roller, Michael Speriosu, Sarat Rallapalli, Benjamin Wing, and Jason Baldridge. “Supervised Text-based Geolocation Using Language Models on an Adaptive Grid”. In: *EMNLP-CoNLL ’12*.

[99] Adam Sadilek, Henry Kautz, and Jeffrey P. Bigham. “Finding Your Friends and Following Them to Where You Are”. In: *WSDM ’12*.

[100] Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. “Earthquake Shakes Twitter Users: Real-time Event Detection by Social Sensors”. In: *WWW ’10*.

[101] Hanan Samet, Jagan Sankaranarayanan, Michael D. Lieberman, Marco D. Adelfio, Brendan C. Fruin, Jack M. Lotkowski, Daniele Panozzo, Jon Sperling, and Benjamin E. Teitler. “Reading News with Maps by Exploiting Spatial Synonyms”. In: *Commun. ACM* 57.10 (Sept. 2014), pp. 64–77.

[102] Jagan Sankaranarayanan, Hanan Samet, Benjamin E. Teitler, Michael D. Lieberman, and Jon Sperling. “TwitterStand: News in Tweets”. In: *SIGSPATIAL ’09*.

[103] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. “Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting”. In: *NIPS ’15*.

[104] Anders Skovsgaard, Darius Sidlauskas, and Christian S. Jensen. “Scalable top-k spatio-temporal term querying”. In: *ICDE ’14*.

[105] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. “Sequence to Sequence Learning with Neural Networks”. In: *NIPS ’14*.

[106] Akin Tascikaraoglu. “Evaluation of spatio-temporal forecasting methods in various smart city applications”. In: *Renewable and Sustainable Energy Reviews* 82 (Feb. 2018), pp. 424 –435.

[107] Daniel Tunkelang. *A Twitter Analog to PageRank*. 2009. URL: https://thenoisychannel.com/2009/01/13/a-twitter-analog-to-pagerank.

[108] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. “Deep Image Prior”. In: *CVPR ’17*.

[109] Svitlana Vakulenko, Lyndon Nixon, and Mihai Lupu. “Character-based Neural Embeddings for Tweet Clustering”. In: *SocialNLP ’17*.

[110] George Valkanas and Dimitrios Gunopulos. “How the Live Web Feels About Events”. In: *CIKM ’13*.

[111] Yehuda Vardi and Cun-Hui Zhang. “The multivariate L1-median and associated data depth”. In: *Proc. NAS* 97.4 (Feb. 2000), pp. 1423–1426.

[112] Maximilian Walther and Michael Kaisser. “Geo-spatial Event Detection in the Twitter Stream”. In: *ECIR ’13*.

[113] Kazufumi Watanabe, Masanao Ochi, Makoto Okabe, and Rikio Onai. “Jasmine: A Real-time Local-event Detection System Based on Geolocation Information Propagated to Microblogs”. In: *CIKM ’11*.

[114] Hong Wei, Jagan Sankaranarayanan, and Hanan Samet. “Finding and Tracking Local Twitter Users for News Detection”. In: *SIGSPATIAL ’17*.

[115] Hong Wei, Jagan Sankaranarayanan, and Hanan Samet. “Measuring Spatial Influence of Twitter Users by Interactions”. In: *SIGSPATIAL LENS’17*.

[116] Hong Wei, Hao Zhou, Jangan Sankaranarayanan, Sudipta Sengupta, and Hanan Samet. “Residual Convolutional LSTM for Tweet Count Prediction”. In: *WWW ’18 Companion*.

[117] Wei Wei, Kenneth Joseph, Wei Lo, and Kathleen Carley. “A Bayesian Graphical Model to Discover Latent Events from Twitter”. In: *ICWSM ’15*.

[118] Jianshu Weng and Bu-Sung Lee. “Event Detection in Twitter”. In: *ICWSM ’11*.

[119] Jianshu Weng, Ee-Peng Lim, Jing Jiang, and Qi He. “TwitterRank: Finding Topic-sensitive Influential Twitterers”. In: *WSDM ’10*.

[120] Feng Xiao, Tomoya Noro, and Takehiro Tokuda. “Finding News-topic Oriented Influential Twitter Users Based on Topic Related Hashtag Community Detection”. In: *J. Web Eng.* 13.5-6 (Nov. 2014), pp. 405–429.

[121] Wenpu Xing and Ali Ghorbani. “Weighted PageRank algorithm”. In: *CNSR ’04*.

[122] Jun-Ming Xu, Aniruddha Bhargava, Robert Nowak, and Xiaojin Zhu. “Socioscope: Spatio-temporal Signal Recovery from Social Media”. In: *ECML PKDD ’12*.

[123] Yuto Yamaguchi, Toshiyuki Amagasa, and Hiroyuki Kitagawa. “Landmark-based User Location Inference in Social Media”. In: *COSN ’13*.

[124] Yuto Yamaguchi, Tsubasa Takahashi, Toshiyuki Amagasa, and Hiroyuki Kitagawa. “TURank: Twitter User Ranking Based on User-tweet Graph Analysis”. In: *WISE ’10*.

[125] Minji Yoon, Woojeong Jin, and U. Kang. “Fast and Accurate Random Walk with Restart on Dynamic Graphs with Guarantees”. In: *WWW ’18*.

[126] Haiyang Yu, Zhihai Wu, Shuqin Wang, Yunpeng Wang, and Xiaolei Ma. “Spatiotemporal Recurrent Convolutional Networks for Traffic Prediction in Transportation Networks”. In: *Sensors* 17.7 (June 2017).

[127] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael Franklin, Scott Shenker, and Ion Stoica. “Resilient Distributed Datasets: A Fault-tolerant Abstraction for In-memory Cluster Computing”. In: *NSDI ’12*.

[128] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. “Spark: Cluster Computing with Working Sets”. In: *HotCloud ’10*.

[129] Matei Zaharia, Tathagata Das, Haoyuan Li, Timothy Hunter, Scott Shenker, and Ion Stoica. “Discretized Streams: Fault-tolerant Streaming Computation at Scale”. In: *SOSP ’13*.

[130] Chao Zhang, Dongming Lei, Quan Yuan, Honglei Zhuang, Lance Kaplan, Shaowen Wang, and Jiawei Han. "GeoBurst+: Effective and Real-Time Local Event Detection in Geo-Tagged Tweet Streams". In: *ACM TIST* 9.3 (Jan. 2018), 34:1–34:24.

[131] Chao Zhang, Liyuan Liu, Dongming Lei, Quan Yuan, Honglei Zhuang, Timothy Hanratty, and Jiawei Han. "TrioVecEvent: Embedding-Based Online Local Event Detection in Geo-Tagged Tweet Streams". In: *KDD '17*.

[132] Chao Zhang, Keyang Zhang, Quan Yuan, Haoruo Peng, Yu Zheng, Tim Hanratty, Shaowen Wang, and Jiawei Han. "Regions, Periods, Activities: Uncovering Urban Dynamics via Cross-Modal Representation Learning". In: *WWW '17*.

[133] Chao Zhang, Keyang Zhang, Quan Yuan, Fangbo Tao, Luming Zhang, Tim Hanratty, and Jiawei Han. "React: Online multimodal embedding for recency-Aware spatiotemporal activity modeling". English (US). In: *SIGIR '17*.

[134] Chao Zhang, Guangyu Zhou, Quan Yuan, Honglei Zhuang, Yu Zheng, Lance Kaplan, Shaowen Wang, and Jiawei Han. "GeoBurst: Real-Time Local Event Detection in Geo-Tagged Tweet Streams". In: *SIGIR '16*.

[135] Jinxue Zhang, Rui Zhang, Jingchao Sun, Yanchao Zhang, and Chi Zhang. "TrueTop: A Sybil-Resilient System for User Influence Measurement on Twitter". In: *IEEE/ACM Transactions on Networking* 24.5 (Oct. 2016), pp. 2834–2846.

[136] Junbo Zhang, Yu Zheng, Dekang Qi, Ruiyuan Li, and Xiuwen Yi. "DNN-based Prediction Model for Spatio-temporal Data". In: *SIGSPATIAL '16*.

[137] Junbo Zhang, Yu Zheng, Dekang Qi, Ruiyuan Li, Xiuwen Yi, and Tianrui Li. "Predicting citywide crowd flows using deep spatio-temporal residual networks". In: *AAAI '16*.

[138] Yu Zhang, William Chan, and Navdeep Jaitly. "Very deep convolutional networks for end-to-end speech recognition". In: *ICASSP '17*.

[139] Yu Zheng, Huichu Zhang, and Yong Yu. "Detecting Collective Anomalies from Multiple Spatio-temporal Datasets Across Different Domains". In: *SIGSPATIAL '15*.

[140] Xiangmin Zhou and Lei Chen. "Event Detection over Twitter Social Media Streams". In: *VLDB '14*.

[141] Yi Zhu and Shawn Newsam. "Spatio-temporal Sentiment Hotspot Detection Using Geotagged Photos". In: *SIGSPATIAL '16*.