Storytelling in Heterogeneous Twitter Entity Network based on Hierarchical Cluster Routing

Xuchao Zhang[1], Zhiqian Chen[1], Weisheng Zhong[1], Arnold P. Boedihardjo[2], Chang-Tien Lu[1]
[1]Virginia Tech, Falls Church, VA, USA
[2]U. S. Army Corps of Engineers, Alexandria, VA, USA
[1]{xuczhang, czq, zwscn123, ctlu}@vt.edu, [2]arnold.p.boedihardjo@usace.army.mil

***Abstract*—Connecting the dots between diverse entities such as people and organizations is a vital task for forming hypotheses and uncovering latent relationships among complex and large datasets. Most existing approaches are designed to address the relationship of entities in news reports, documents and abstracts, but such approaches are not suitable for Twitter data streams due to their unstructured languages, short-length messages, heterogeneous features and massive size. The sheer size of Twitter data requires more efficient algorithms to connect the dots within a short period of time. We present a system that automatically constructs stories by connecting entities in Twitter datasets. An entity similarity model is designed that combines both traditional entity-related features and social network attributes and a novel story generation algorithm applied on the similarity model is proposed to cope with the massive Twitter datasets. Extensive experimental evaluations were conducted to demonstrate the effectiveness of this new approach.**

I. Introduction

Social media such as Twitter has become a real-time "news press" for disseminating information at both the global and community scales. Hundreds of millions of users post tweets every minute, discussing contents ranging from their opinions about social events to their observations on the street. Compared to traditional media, Twitter has a number of interesting features including: 1) *Promptness*. Unlike traditional media, which may take hours or even days to publish, tweets can be posted instantly using portable mobile devices; 2) *Freedom of expression*. In contrast to the censorship often imposed on traditional media, tweets can more freely express idiosyncratic views and inconvenient facts; and 3) *Social properties* [1][2]. Compared to the flat information presented in traditional media, Twitter adds value through its ability to link users to their personal networks where their social information such as friends and geo-locations are held. Recent research have revealed the power of connecting the entities in traditional documents to help uncover important relationships between two concepts which are not readily observable [3]. For example, the Mexican election of its president Enrique Peña Nieto was marred by media bias and an alleged record fee charged by Televisa, the largest Mexican multimedia company. The connection between the Mexican president and the multimedia company Televisa is shown in Figure 1, where related persons and organizations are linked with junction points such as common Twitter followers, related tweets and links. In the storyline, some key persons related to the event are revealed. For instance, Carlos Loret, the news anchor who confirmed that the station's Corporate Vice President of Marketing is linked to Peña Nieto during his administration, which is supported by the tweet "Carlos Loret confirmed the Peña-Televisa alliance news from his former collaborator Laura Barranco. #Saltillo"[1].

Techniques to connect the dots in news reports [4][5], documents [6] and abstracts [7] have been well studied, but all these methods are based on a strong assumption that the textual contents are robust and well presented. This is not always the case for social media; analyzing entity relationships in a Twitter dataset requires specific techniques to address its unstructured language, short-length message, and heterogeneous elements. Therefore, existing approaches that focus solely on traditional documents cannot be applied to Twitter data because many of the features in Twitter, such as followers and mentions, cannot be dealt with using traditional methods. This poses two challenges: 1) *Modeling heterogeneous features in Twitter*. Twitter data contains various features, including hashtags, mentions and links. Entities extracted from tweets can also be mapped to Twitter users, from which follower and geo-location information can be obtained. Joint consideration of these heterogeneous Twitter features is crucial; and 2) *Handling large data size*. Hundreds of millions of tweets are generated daily from Twitter. An efficient story generation algorithm is required to handle this massive data.

In this paper, we present a method to model these heterogeneous features in a bipartite graph and generate an entity similarity graph for story modeling. A novel story generation algorithm based on hierarchical cluster is also proposed to handle the massive Twitter datasets. The key contributions of this research are summarized as follows:

- **Novel method to model Twitter heterogeneous feature similarity.** An entity similarity model was proposed to combine heterogeneous features of Twitter data in a bipartite graph and generate an entity similarity graph with a random walk algorithm. Existence of a unique sta-

[1]Translated by original spanish tweet: "Carlos Loret confirmó a Laura Barranco ex colaboradora de primero noticias la alianza Peña-Televisa #Saltillo"

978-1-4673-9005-7/16/$31.00 ©2016 IEEE

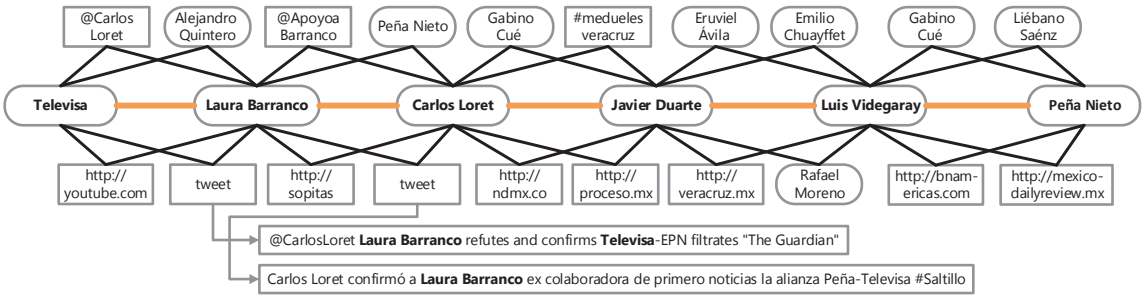


Fig. 1. Case Study of Media Bias in 2012 Mexico President Election

tionary distribution guarantees the convergence of entity similarity calculation.

- **Innovative story generation algorithm based on hierarchical clustering.** Utilizing on the entity similarity graph generated by our model, we have designed a novel story generation algorithm that efficiently prunes the data and finds a story by dividing the search space into hierarchical partitions. Linear scalability is achieved by utilizing finer partition granularity .
- **System to connect different entities in Twitter using above proposed model.** A new system was developed to connect related entities given a set of Twitter data, which contains the following steps: (i) data preprocessing, (ii) entity similarity modeling, and (iii) story generation.
- **Extensive experimental performance evaluations.** The experiments on different types of data sets presented here demonstrated that our proposed approach outperformed existing state-of-the-art algorithms. Sensitivity analyses were conducted on 10 different data sets.

The remainder of this paper is organized as follows. Section II describes the related work on entity similarity calculations and connecting the dots problems. Section III provides the problem formulation and overall architecture. Section IV and Section V study the detailed techniques involved in the entity similarity model and story generation, respectively. In Section VI, the extensive experimental results are analyzed. The paper concludes by summarizing the important findings of the study in Section VII.

II. Related Work

This section summarizes the current state of research in entity similarity modeling, microblog summarization, and connecting the dots. Both methods based on entity networks [8] and document collections [9][10][7] are discussed.

A. Entity Similarity Modeling

There is a large body of work that focuses on the analysis of semantic [11][12], syntactic [13][14] and spatio-temporal [15][16] entity similarity. All of these papers analyze the relationship between entities by utilizing their semantic context and/or spatial locations. Entity similarity methods are also used in story generation. Hossain et al. [3] proposed a method to compute entity similarity by combining Soergel Distance and a k-Clique nearest neighbor approach; Shahaf and Guestrin [6] used a linear program method to measure words coherence between documents; Dos Santos et al. [1] suggested a ConceptRank method and spatial closeness to infer relationships to entities in Twitter data; and Goel et al. [17] employed a regression model to compute Twitter users' similarity based on their common followers, pagerank score and historic follow-through rate. However, these methods can only uncover similarities in traditional documents or specific Twitter features individually and are unable to consider their features in combination.

B. Microblog Summarization

Although document summarization [18][19] has been studied for years, microblog summarization is still in its infancy. Sharifi et al. proposed the Phrase Reinforcement algorithm [20] to create a summary of microblog posts related to user defined terms and Inouye et al. [21] proposed both Hybrid TF-IDF and cluster classifier methods to generate multiple post summaries, while Harabagiu et al. [22] introduced a framework to synthesize multiple microblog posts on the same topic into a prose summary. Takamura et al. [23] took the posted time of microblogs into consideration, proposing a summarization model based on the p-median problem for a stream of microblog posts along a timeline. Later, Lidan et al. [24] proposed an online tweet stream clustering algorithm and TCV-Rank summarization method for tweet streams. All the above methods aim to extract semantic meaning from Twitter, but none of them consider the relationship between entities.

C. Connecting the dots

Connecting the dots methods have received a lot of attention in recent years. Hossain et al. [3] proposed a method using A* searching algorithm to construct a shortest path between entities based on the concept lattice network, while Dos Santos et al. [1] connected entities via a greedy approach using entity semantic, spatial, and temporal ordering. Despite considering Twitter metadata such as hashtags, and mentions, following and follower relationships, their storyline focus on reflecting temporal sequences within a predefined spatial area. Zhu [10] took a different approach, connecting documents by applying a divide-conquer algorithm to append a median node with

maximum transition probability between nodes in each iteration. Faloutsos et al. [8] proposed a method to find connected subgraphs that maximized the delivered electric current using a dynamic programming algorithm. Shahaf and Gustrin [6] proposed finding a story by maximizing its weakest edge with a fixed story length. Most of the aforementioned techniques seek to identify ways to connect entities in traditional documents, but cannot directly be applied to heterogeneous and massive Twitter data.

Our work differs from most previous works in two major aspects, namely our use of entity similarity model that combines heterogeneous features of Twitter and the novel hierarchical routing algorithm with unconstrained layers in entity similarity network that we apply to generate story lines efficiently and effectively.

III. Overview and Problem Formulation

In this section, we formally define the problem of storytelling in a Twitter entity network and present the main steps required to solve the problem, along with some key definitions and the concepts involved.

A. Problem Setting

Our goal is to reveal valuable relationships between two entities via a sequence of intermediate entities in Twitter dataset. Formally, given two entities v_s and v_t, our task is to connect them together using a sequence of entities $v_1, v_2, ..., v_k$ in a set of tweets $\mathcal{T}$. Given the Twitter dataset $\mathcal{T}$, relationships between entities can be formulated as an Entity similarity graph $\mathcal{G}_\mathcal{T}$:

Definition 1. Entity Similarity Graph: Given a set of tweets $\mathcal{T}$, an entity similarity graph is defined as an undirected graph $\mathcal{G}_\mathcal{T} = (\mathcal{V}, \mathcal{E})_\mathcal{T}$, where $\mathcal{V}$ denotes entities in Twitter dataset $\mathcal{T}$ and $\mathcal{E}$ denotes edges between entities with their similarity weights.

Generally, the coherence between entities in a storyline is evaluated by three metrics: 1) Average edge weight, 2) Dispersion Coefficient [3] and 3) Minimum edge weight [9]. Average edge weight is defined as the average similarity between each connected entities in a story. The drawback of using AvgEdge to evaluate a storyline is as follows: the average edge weight can only evaluate the whole storyline, ignoring the individual edges. Suppose one edge in the middle of a storyline is much lower than others. Here, the story is interrupted and split into two parts. The Dispersion Coefficient is a matrix that evaluates all story nodes rather than simply the conjunctive ones. A storyline that contains n entities $v_0, v_1, \ldots, v_{n-1}$ is quantified as: $\nu = 1 - \frac{1}{n-2}\sum_{i=0}^{n-3}\sum_{j=i+2}^{n-1} disp(v_i, v_j)$, where $disp(v_i, v_j)$ equals to $\frac{1}{n+i-j}$ if $S(v_i, v_j) < \theta$, otherwise it equals to zero. In the equation, $S(v_i, v_j)$ is the similarity between entity v_i and v_j; and disp(v_i,v_j) is used to evaluate the dispersion between entity v_i and v_j. If the similarity between the two non-consecutive entities is larger than some predefined threshold θ, its dispersion is zero. The intuition involved in using the Dispersion Coefficient is analogous to finding a community rather than a storyline, but the threshold parameter θ is difficult to define.

The rationale for the Minimum edge weight matrix comes from Liebig's law [25]. Drawing an analogy between a story line and a piece of string: in general the strength of the string is not determined by its strongest part but its weakest. Formally, this is defined as:

Definition 2. MinEdge: Given a storyline $s = \{v_1, v_2, \ldots, v_t\}$, MinEdge is defined as the minimum edge between each connected entity in s:

$$MinEdge \equiv \min\{e_i | \forall e_i \in \mathcal{E}\} \tag{1}$$

where $\mathcal{E}$ is the set of edges in story line s.

The MinEdge weight has a close relationship with the other two metrics in the two aspects: 1) *MinEdge* is the infimum of AvgEdge. Maximizing MinEdge weight can improve the AvgEdge weight. 2) MinEdge not only controls conjunctive nodes, but also the nodes throughout the whole storyline.

Based on these observations, we define our entity-based storytelling task as follows:

Definition 3. Storytelling in Entity Network: Given an entity similarity graph $\mathcal{G}_\mathcal{T} = (\mathcal{V}, \mathcal{E})_\mathcal{T}$, find a story chain of entities $s = \{v_1, v_2, \ldots, v_t\}$ which satisfies:

$$\underset{s \in S}{\operatorname{argmax}}\, f(s) = \{s | f(s) = MinEdge(s)\} \tag{2}$$

where $\mathcal{S}$ is the set of all possible stories starting from v_1 to v_t.

B. Architecture Overview

To achieve this goal, our task can be divided into three major parts, as shown in figure 2: 1)Data Preprocessing, 2)Entity Similarity Modeling, and 3)Story Generation.

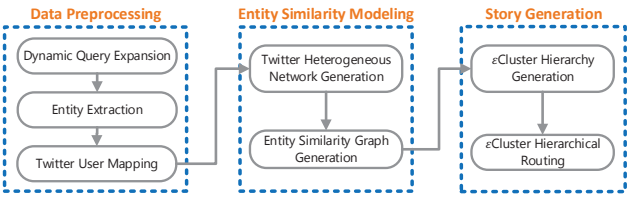


Fig. 2. System Architecture

Data Preprocessing phase retrieves Twitter data and entities within a targeted domain, which is used as the input for further steps. Here, data preprocessing consists of the following steps:

(1) Query Expansion: Searching in a uncategorized Twitter dataset[2] that contains a large amount of irrelevant information, is computationally impractical. Therefore, Dynamic Query Expansion [2](DQE), an unsupervised approach to automatically expand seed query in targeted domains is used to retrieve a subset of tweets in a specific targeted domain of our interest.

[2]The Twitter data used in this paper was purchased from Datasift Inc (http://datasift.com/). All analyses here are done in compliance with the Twitter and Datasift terms of use. Twitter data is available through either the public Twitter API (https://dev.twitter.com) or through authorized resellers such as Gnip.com and Datasift.com.

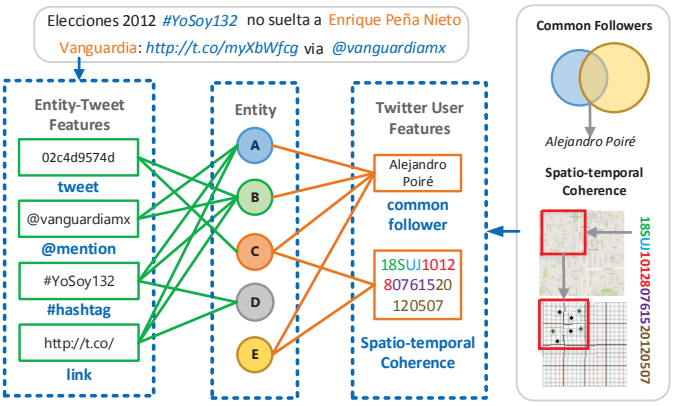


Fig. 3. **Sample of Twitter Information Heterogeneous Network**. In this graph, entities are linked to six different Twitter features: tweets, mentions, hashtags, links, common followers, and spatio-temporal coherence. All of these features belong to two main categories: Entity-Tweet feature and Twitter user features.

(2) Entity Extraction: In this step, our purpose is to extract person and organization entities from the Twitter dataset filtered by DQE. We chose a state-of-the-art method, the Stanford Named Entity Recognizer [26], to extract entities. The Stanford Deterministic Coreference Resolution System [27] is also applied to avoid duplicate entities when two or more expressions in a text refer to the same person or organization.

(3) Twitter User Mapping: The purpose of Twitter user mapping is to map the extracted entities to Twitter users. For example, the entity "*Barack Obama*" will be mapped in tweets to the Twitter user *@BarackObama*. Given the popularity of Twitter, most of the entities extracted from recent tweets have active Twitter accounts, especially for celebrities. One direct way to accomplish this task is to search for specific user names using Twitter REST API[3]. However, in most circumstances, the API returns a long list of users who have the same name and it is impossible to identify each individual mapping. Therefore, we designed a method to map entities to a specified Twitter user pool, which is iteratively increased by the followers of newly mapped Twitter users. The detailed approach is as follows:

1) Initialize Twitter user pool Φ_0 as official authenticated users[4] and E_0 as retrieved entity set.
2) Find a mapping between entity set $E_k^m \subseteq E_k$ and Twitter user set $\Phi_k^m \subseteq \Phi_k \cup \Phi_k^{'}$ where E_k is current entity set and $\Phi_k^{'}$ is the follower user set of current Twitter user set Φ_k. The mapping is based on Twitter user description name and their profile information.
3) Set $\Phi_{k+1} \leftarrow \Phi_k \cup \Phi_k^m$ and $E_{k+1} \leftarrow E_k \setminus E_k^m$
4) Repeat step 2 and step 3 until $\Phi_{k+1} = \Phi_k$ or an iteration threshold is reached.

Entity Similarity Modeling: Given a set of tweets $\mathcal{T}$, the entity similarity model constructs the Twitter Heterogeneous Information Network, which is defined as follows:

Definition 4. Twitter Heterogeneous Information Network: Given a Twitter subcollection, a Twitter heterogeneous information network is defined as an undirected bipartite graph $\mathcal{H}_\mathcal{T} = (\mathcal{F}, \mathcal{V}, \mathcal{E})$, where $\mathcal{F}$ denotes the node set of Twitter's heterogenous features, such as "tweets", "hashtags", and "mentions". $\mathcal{V}$ and $\mathcal{E}$ denote entities and edges between entities and features, respectively.

Figure 3 depicts a sample Twitter Heterogeneous Information Network in which entities are connected to six different Twitter features. Based on the bipartite graph network, the similarity can be calculated by a random walk algorithm. Then, an entity similarity graph $\mathcal{G}_\mathcal{T}$ can be generated by using the similarity as the edge weight between two entities.

More details of the entity similarity model and random walk algorithm will be presented in Section IV. After the entity similarity graph has been constructed, the problem of finding a story can then be modeled as a path-searching problem in the graph.

Storyline Generation phase performs directed exploration toward a desired entity through an entity similarity graph $\mathcal{G}_\mathcal{T}$. However, direct permutations of all the possible story lines of $\mathcal{G}_\mathcal{T}$ with thousands of entities and edges to find a story with maximum *MinEdge* is clearly impractical. Instead, our proposed εCluster story generation algorithm divides the graph into smaller partitions. Within each partition, any two entities can be connected with a line whose MinEdge is larger than a predefined ε threshold but the MinEdge cross different partition is less than the threshold. Partitioning the Entity Similarity Graph removes many of the candidate nodes from the search space. The addition of graph partitioning in hierarchical layers can further improve the search performance.

IV. Entity Similarity Model

In this section, different features of a Twitter Heterogeneous Network are firstly discussed, after which a random walk based similarity calculation method in a Twitter heterogeneous network is presented.

A. Twitter Heterogeneous Network Generation

A Twitter heterogeneous network is an undirected bipartite graph constructed by using entities and their corresponding Twitter features. Both entity-tweet features and Twitter user related features are included as follows:

Entity-tweet features:

(a) **Tweets**. If two entities exist in the same tweet, the tweet will be added as a feature into the Twitter Heterogeneous Information network and the edges between the tweet and the entities are also appended. Note that if only one entity exists in a tweet, the tweet is not allowed to be added as a feature because no other entity can be traversed via the tweet. In the example shown in Figure 3, entity A(Peña Nieto) and entity B(Vanguardia) are both connected to tweet "02c4d9574d" as they appear in the same tweet.

[3]https://dev.twitter.com/rest/public

[4]All officially authenticated Twitter accounts are followed by Twitter user *@verified*.

(b) **#Hashtags**. Hashtags connect all entities related to same tags; if an entity exists in a tweet that contains a hashtag, the hashtag and edges between the hashtag and entities will be added to the bipartite graph. As with the tweet feature, entity A and B in figure 3 are both connected to hashtag #YoSoy132.

(c) **@Mentions**. Given that the mentioned user is also an entity, we can now connect the entity to all the features in the tweet, including the mention feature itself. Figure 3 shows how entity B(Vanguardia) connects to all the entity-tweet features, including the mention @vanguardiamx itself.

(d) **Tweet Links**. Sometimes, tweets contain additional links to other web resources such as news articles or videos. After extracting entities contained in these resources, we connect the entities that also belong to entities in the Twitter dataset to the link.

Twitter user features:

(a) **Common Followers.** Common Followers [17] are an important feature for evaluating users' similarity and can also be used to recommend similar users in Twitter user recommendation systems [17]. In our bipartite graph, avoiding using all the common followers between two users, we selectively choose the common followers who are also entities in our dataset. This approach has the following two properties: (i) *Computational Efficiency*. Millions of Twitter users follow well-known people, so it is clearly impractical to include all of them in our bipartite graph. (ii) *Noise Removal*. Well-known people are often followed by irrelevant people such as fans. These unrelated users are not of interest and should therefore not be included in a story.

(b) **Spatial attribution.** Geo-location data is recorded when a Twitter user posts a tweet with location information. This information can be used to analyze the user's spatio-temporal similarity with other users. However, geo-location coverage of tweets is sparse: only 15% of tweets in our data collection are associated with geo-location information. To solve this problem, we can use Carmen [28] to enrich the geo-location data using the user's profile and the "Place" object from tweets. Nearly 40% of the tweets can be tagged with geo-location information after the enrichment process. The Military Grid Reference System(MGRS) [29] is used to represent a spatial location, allowing nearby locations to be combined for the analysis. Figure 3 shows an example for spatial position *18SUJ1012807615* on March 7, 2012.

B. Entity Similarity Graph Generation

A random walk based similarity computation method can be used to generate the similarity between each entities based on the Twitter heterogeneous network. Before introducing the similarity calculation, some of the terms used in this section will be defined.

- **Edge Weight**. We use the symbol $l_{i,j}$ to represent the edge between entity e_i and feature f_j. α_{f_j} represents the weight for the j[th] feature. Edge weight w_{e_i,f_j} is used to represent the weighted link frequency between e_i and f_j.

$$w_{e_i,f_j} = \sum_l \mathbb{1}(l_{i,j} \; between \; e_i, \; f_j) \times \alpha_{f_j} \tag{3}$$

- **Transition Probability**. Transition Probability represents the probability of transferring from one node to another. The transition probability from entity e_i to feature f_j is expressed as: $P(e_i \rightarrow f_j) = \frac{w_{e_i,f_j}}{\sum_k w_{e_i,f_k}}$. Similarly, the transition probability from feature f_i to e_j is expressed as: $P(f_i \rightarrow e_j) = \frac{w_{f_i,e_j}}{\sum_k w_{f_i,e_k}}$.

To capture the similarity between two entities, we use the probability of a random walker passing through the two entities on the bipartite graph to represent their similarity value. In other words, the higher the probability that a random walker will pass from one entity to another, the more similar they become. Yildirim and Coscia [30] proposed a method to compute an exact stationary distribution after infinite random walking. We will apply this method to our Twitter heterogeneous network to compute the similarity between entities.

First of all, the probability between entity e_i and $e_{i'}$ is the summation of all paths from e_i to $e_{i'}$ that pass through all the linked features between e_i and $e_{i'}$:

$$\begin{aligned} P(e_i \rightarrow e_{i'}) &= \sum_j P(e_i \rightarrow f_j \rightarrow e_{i'}) \\ &= \sum_j P(e_i \rightarrow f_j) \times P(f_j \rightarrow e_{i'}) \\ &= \sum_j \frac{w_{e_i,f_j}}{\sum_{j' \in J} w_{e_i,f_{j'}}} \frac{w_{e_{i'},f_j}}{\sum_{i'' \in I} w_{e_{i''},f_j}} \end{aligned} \tag{4}$$

Based on the previous definition, the transition probability between entities can be restated in terms of a Markov transition matrix T in which $T_{i,i'} = P(i \rightarrow i')$. The Perron-Frobenius theorem guarantees the existence of a unique stationary distribution if the transition matrix T has the following three properties:

1) Right-stochastic: Transition matrix T is a right-stochastic matrix[5]. According to the attributes of the bipartite graph, this property is satisfied.
2) Irreducible: every node can communicate with each of the others within finite step. To satisfy this property, a breadth first search can be applied to the bipartite graph *G* to identify its connected components, with each connected component being treated as a different bipartite graph G_i. G is thus divided into a set of sub-graphs G = ($G_1 \ldots G_n$) and any entity in one subgraph is not reachable by an entity in another subgraphs.
3) Aperiodic: there is no $\vec{x}$ and integer m $>$ 1 such that $\vec{x}T^m$ = $\vec{x}$ but $\vec{x} \neq$ T$\vec{x}$. As we only work with bipartite graphs with non-directed edges, the aperiodicity property is satisfied.

Given that transition matrix T has these three properties, the Perron-Frobenius theorem guarantees that a unique stationary

[5]right-stochastic matrix's elements are non-negative and sum of rows is 1.

distribution exists when the eigenvalue is 1: $\vec{\pi}$T = $\vec{\pi}$. After an infinite period of random walking, the similarity between nodes i and i' converges to: $S_{i,i'} = \pi_i T_{i,i'}$, where π_i is the i[th] element in the stationary distribution $\vec{\pi}$. The advantage of this model is that it avoids saturation issues such as those that occur when an additional shared entity node between two feature nodes that share only one entity node has more effect than that between two nodes who already share 100 entity nodes.

V. Story Generation Algorithm

This section details the core of the storytelling technique, εCluster, a hierarchical routing algorithm that can efficiently be executed on a large network.

A. εCluster Preliminary

To partition a large entity similarity graph into smaller subgraphs, the whole graph must first be divided into different layers. These layers are called εCluster layers. The layer itself is an undirected graph G_l=(Ψ, Π, S, E), where Ψ refers to a set of clusters, and Π represents the vertices within each cluster. These vertices are also clusters belonging to lower layer. S$\subseteq\Psi\times\Psi$ represents the set of edges between clusters and $E \subseteq \Pi\times\Pi$ refers to the set of edges within clusters. The edge between two cluster vertices $\psi_i, \psi_j \in \Psi$ is denoted as s_{ij} and the edge between vertices $\pi_i, \pi_j \in \Pi$ within a cluster as e_{ij}. The inter-cluster edge s_{ij} is represented as the maximum edge connected by vertices from different clusters. Suppose ε is a predefined threshold for edges in each cluster layer k, $s_{i,j}$ and $e_{i,j}$ satisfy the following properties: (1) $\forall\psi_i, \psi_j \in \Psi$, $s_{i,j} < \varepsilon_k$. (2) $\forall\pi_i, \pi_j \in \Pi$, $e_{i,j} \geq \varepsilon_k$. In the example of Figure 4(a), $s_{1,2} < \varepsilon_k$ and $e_{4,5} \geq \varepsilon_k$. Signal $\psi_{(i)}$ is used to represent the cluster of vertex π_i, where subscript *(i)* refers to the id of the cluster. Thus, $s_{(i),(j)}$ refers to the cluster edge linked between vertex π_i and π_j. Figure 4(a) shows an example of an εCluster Layer, in which $\Psi = \{\psi_1, \psi_2, \psi_3\}$, Π=$\{\pi_1, \pi_2, \ldots, \pi_{12}\}$, $\psi_{(4)} = \psi_2 = \{\pi_4, \pi_5, \pi_6, \pi_7, \pi_8\}$, $s_{(1),(4)} = s_{1,2}$. Finally, εCluster Hierarchy $H = \langle G_1, G_2, \ldots, G_K\rangle$ is defined as a sequence of εCluster layers in ascending order. Note that the layer number of an εCluster Hierarchy is a predefined parameter. Generally, the εCluster story generation algorithm becomes faster as more layers are constructed.

B. εCluster Hierarchy Generation

εCluster Hierarchy Generation aims to build a hierarchical cluster structure whose edges between or within clusters satisfy a specified threshold in order to decrease the computation cost. The detailed algorithm is shown in Algorithm 1. The major components of the algorithm are described below.

Initialization. Given an entity set Π_1 with a similarity edges set E_1, each entity vertex in Π_1 is assigned a unique cluster based on the vertex only. Edges of clusters S_1 are the same as the edges set E_1.

Layer Generation. During the process of layer generation, all the cluster edges in the lower layer are iterated. For each edge, if it is greater than some predefined ε value in the current layer, we can link the two entities together and merge the two clusters if they are in separate clusters. If the edge is less than ε and links different clusters, we compare the maximum edge between the two clusters. If the edge is greater than the maximum edge, we use the current edge value to replace the existing one. If they are in the same cluster, the edge is ignored.

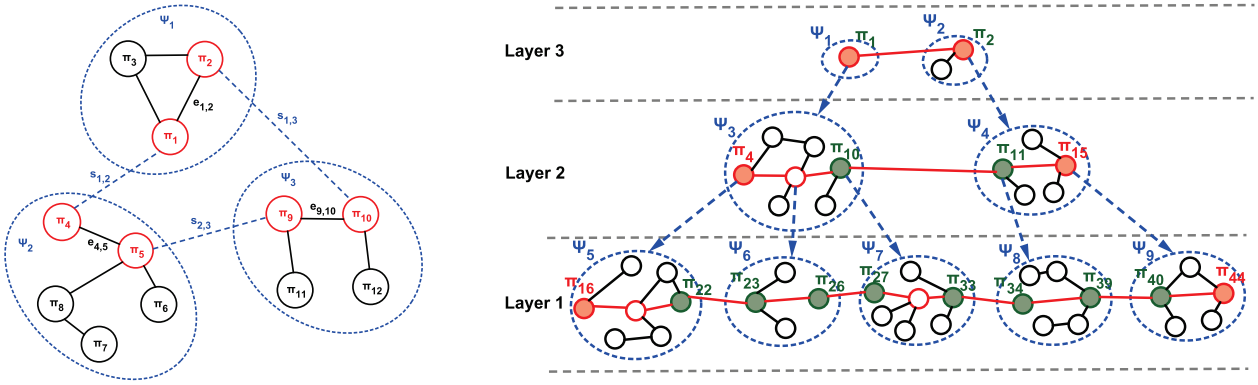
Algorithm 1: Generate εCluster Hierarchy

Input: Entity set Π_1; Edge set E_1 between entities; a sequence of threshold set $\boldsymbol{\varepsilon}$=$\langle\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_K\rangle$ for a *K*-layer hierarchy
Output: εCluster hierarchy with *K* layers
// initialize the first layer
for *each* $\pi_i \in \Pi_1$ **do**
  assign π_i to a stand-alone cluster ψ_i in G_1
  $S_1 \leftarrow E_1$ *// initialize 1st layer edge set*
$k \leftarrow 2$
while $k \leq |\boldsymbol{\varepsilon}|$ **do**
  $S_k \leftarrow \emptyset$; $E_k \leftarrow \emptyset$ *// initialize cluster edge sets*
  while $S_{k-1} \neq \emptyset$ **do**
    $e_{i,j} \leftarrow$ pop S_{k-1}
    if $e_{i,j} \geq \varepsilon_k$ **then**
      if $(i) \neq (j)$ **then**
        $\psi_{(i)} \leftarrow \psi_{(i)} \cup \psi_{(j)}$ *// merge cluster $\psi_{(i)}$ and $\psi_{(j)}$*
        for *each* $\psi_m \in \Psi$ **do**
          if *(i) ≠m and (j)≠m and* $s_{(i),m} < s_{(j),m}$ **then**
            $s_{(i),m} \leftarrow s_{(j),m}$ *// update cluster edge*
      else
        $E_k = E_k \cup e_{i,j}$ *// add $e_{i,j}$ to cluster edges*
    else
      if $e_{i,j} < \varepsilon_k$ *and* $s_{(i),(j)} < e_{i,j}$ **then**
        $s_{(i),(j)} \leftarrow e_{i,j}$
        $S_k \leftarrow S_k \cup s_{(i),(j)}$
  $k \leftarrow k + 1$

Hierarchy Generation. Based on the layer number and ε values for each layer, all the layers are generated to construct an εCluster hierarchy. Rather than connecting all the layers together, a hierarchy also stores the following information: (i) Record the mapping relation of upper layer cluster id to its lower layer cluster id set. (ii) Store the mapping from lower layer cluster id to its upper layer cluster id. (iii) Establish the mapping relationship between the entity id and its lowest cluster id. Notice that the layer number and ε values determine how the original entity similarity graph is partitioned; generally, the graph will be more finely partitioned if using a larger layer number.

C. εCluster Hierarchical Routing

The εCluster hierarchical routing algorithm identifies the story line with the maximum *MinEdge* based on the εCluster hierarchy introduced in section V-B. Pruning most of the irrelevant edges in each εCluster hierarchy layer accelerates the computation process dramatically. Overall, this storyline generation method based on cluster hierarchy is a divide and conquer algorithm. The detailed algorithm is shown in



(a) εCluster Layer Graph Sample (b) εCluster Storyline Generation Sample

Fig. 4. εCluster algorithm sample. (a) shows a sample layer of an εCluster hierarchy; (b) depicts a story generation sample based on a three-layer εCluster hierarchy.

Algorithm 2 and Figure 4(b) depicts an example based on a three-layer hierarchy.

Signal $\psi_{(i)^j}$ represents the cluster in the j$^{\text{th}}$ upper layer. For example, $\psi_{(i)^4}$ refers to the cluster in the 4$^{\text{th}}$ upper layer of π_i, and $\psi_{(i)^1}$ is equivalent to $\psi_{(i)}$. Meanwhile, $|H|$ refers to the layer number of hierarchies H. We define two types of entity vertex: the output vertex from one cluster $\pi_{[\psi_i],\psi_j}$ and the input vertex to one cluster $\pi_{\psi_i,[\psi_j]}$. Considering the example in Figure 4(b), the green vertex in cluster ψ_3 is the output vertex $\pi_{[\psi_3],\psi_4} \equiv \pi_{10}$ and the green vertex in cluster ψ_4 is the input vertex $\pi_{\psi_3,[\psi_4]} \equiv \pi_{11}$. The lower cluster of entity π_i is denoted as $\psi_{\lfloor i \rfloor}$, which means that $\psi_{\lfloor 10 \rfloor} \equiv \psi_7$ in the example of Figure 4(b).

The story line search within a cluster is performed by *findOptimalChain()*(abbreviated as *foc()*), where we use the same algorithm as that presented by Shahaf and Guestrin [9]. τ is the parameter applied to restrict the length of story chain within a cluster. Let us now consider the εCluster algorithm based on the example of story from vertex π_{16} to π_{44} shown in Figure 4(b). From the top layer, layer 3, a sequential chain $\langle \psi_1, \psi_2 \rangle$ is found, after which the problem is transferred to finding optimal chains in $\psi_{\lfloor 1 \rfloor} \equiv \psi_3$ and $\psi_{\lfloor 2 \rfloor} \equiv \psi_4$. The edge between the two clusters is connected by the edge between the output vertex of cluster ψ_3 and the input vertex of cluster ψ_4. For the starting cluster ψ_3, the chain begins with the start vertex π_4 and proceeds to the output vertex π_{10}. Similarly, for the ending cluster ψ_4, the chain runs from the input vertex π_{11} to the end vertex π_{15}. For the median clusters, for example, cluster ψ_7, the chain runs from the input vertex π_{27} to output vertex π_{33}. Finally, a storyline with sequential entity vertex in Layer 1 is generated as $\langle \pi_{16}, \ldots, \pi_{22}, \pi_{23}, \pi_{26}, \pi_{27}, \ldots \pi_{33}, \pi_{34}, \pi_{39}, \pi_{40}, \pi_{44} \rangle$.

Algorithm 2: GENERATE εCLUSTER STORYLINE

Input: Cluster Hierarchy H; Start entity π_s and end entity π_t; Story length τ restricted within clusters
Output: a sequence of entities $C = \langle \pi_1, \pi_2, \ldots, \pi_n \rangle$

$k \leftarrow |H|$ *// assign k to layer number of hierarchy*
$C \leftarrow \emptyset$ *// initialize result set*
while $k \geq 1$ **do**
  $C' \leftarrow \emptyset$ *// initialize story chain in current layer*
  if $|C| == 1$ **then**
    // find chain from start to end vertex
    $C \leftarrow C' \cup foc(G_k, \pi_s, \pi_t, \tau)$
    $k \leftarrow k - 1$
    continue
  // find chain from start to output vertex in 1st cluster
  $\pi_{out'} \leftarrow \pi_{[C_1],C_2}$
  $C' \leftarrow C' \cup foc(G_k, \pi_s, \pi_{out'}, \tau)$
  i $\leftarrow 2$
  while $i < |C|$ **do**
    $\pi_{in} \leftarrow \pi_{C_{i-1},[C_i]}$ *// find chain from input to output vertex*
    $\pi_{out} \leftarrow \pi_{[C_i],C_{i+1}}$
    $C' \leftarrow C' \cup foc(G_k, \pi_{in}, \pi_{out}, \tau)$
    $i \leftarrow i + 1$
  // find chain from input in last cluster to end vertex
  $\pi_{in'} \leftarrow \pi_{C_{|C|-1},[C_{|C|}]}$
  $C \leftarrow C' \cup foc(G_k, \pi_{in'}, \pi_t, \tau)$
  $k \leftarrow k - 1$

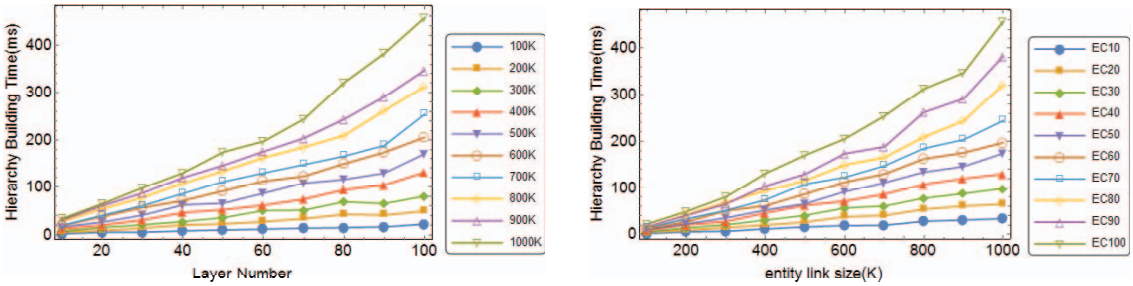
VI. EXPERIMENTAL RESULTS

In this section, the performance of εCluster is evaluated. We begin by evaluating the εCluster hierarchy construction performance and story generation throughput of εCluster on real-world data sets, after which, the effectiveness of εCluster against existing state-of-the-art methods in connecting-the-dots tasks is compared. Finally, an empirical case study of an event associated with the Mexico presidential election is used to demonstrate the utility of the new system. All the experiments presented in this paper were conducted on a 64-bit machine with Intel Xeon CPU E5-1603 @2.80GHz and 64GB physical memory.

A. Experiment Setup

To comprehensively evaluate both the performance and applicability of the new εCluster algorithm, two different types of datasets were selected: (i) GDelt[6]. The GDelt dataset was retrieved from Google GDelt Service using keyword *MH370* for the period from March 8, 2014 to April 8, 2014, and contains 6,729 documents and 3,850 entities. (ii) Twitter

[6]Supported by Google Ideas, the project monitors the world's broadcast, print, and web news all over the world



(a) Hierarchy building time of different layers

(b) Hierarchy building time of different data size

Fig. 5. εCluster Hierarchy Generation Performance. (a) shows the hierarchy building time based on different layer number from 1 to 100. (b) depicts the hierarchy building time on different data size from 1K to 1000K entity links.

civil unrest dataset[7]. This data set was obtained by randomly sampling 10%(by volume) of the civil unrest Twitter data from July 2012 to December 2012 in 4 countries in Latin America: Brazil, Paraguay, Mexico, and Venezuela. Here, 51,000 entities and 448,000 tweets are included. For this evaluation of the performance of the εCluster hierarchy constructed in this research, we compared ten different sized datasets ranging from 100,000 to 1 million entity edges.

To evaluate the effectiveness and efficiency of our approach, we compared the results obtained with those generated by two other baseline methods on connecting-the-dots tasks. The implementation and parameters settings are listed below:

Connect-the-dots(CTD): The Connect-the-dots [9] algorithm finds storylines based on *MinEdge*. The process begins by predefining a parameter for storyline length K. For each iterated entity, the algorithm will iterate across all its neighbors that can be reached in less than K steps. As the algorithm becomes very time-consuming with increasing story length, we set K = 5 as the maximum value due to the limitations of our available equipment.

Local Optimal-MinEdge(LOME): This method [10] is a divide and conquer algorithm that iteratively inserts intermediate node $d_{i'}$ between two connected nodes d_i and d_{i+1} with an objective function. The function can be defined as follows if applied to the *MinEdge* objective: $d_{i'} = \arg\max_{d_i} min\{s(d_i, d_{i'}), s(d_{i+1}, d_{i'})\}$, where $s(\cdot)$ is the function of similarity between two entities. Similar to Connect-the-dots method, story length is required to be given as a parameter.

εCluster(EC20, EC50, EC100): For the εCluster algorithm, we define a hierarchy of *n* layers in which ε is defined as an arithmetic progression from 1 to 0 whose common difference is $\frac{1}{n}$. Here, the evaluations of hierarchies consisting of 20(EC20), 50(EC50) and 100(EC100) layers were performed.

B. εCluster Algorithm Performance

To show the efficiency of εCluster algorithm, the performance of εCluster hierarchy generation and εCluster story generation is evaluated separately.

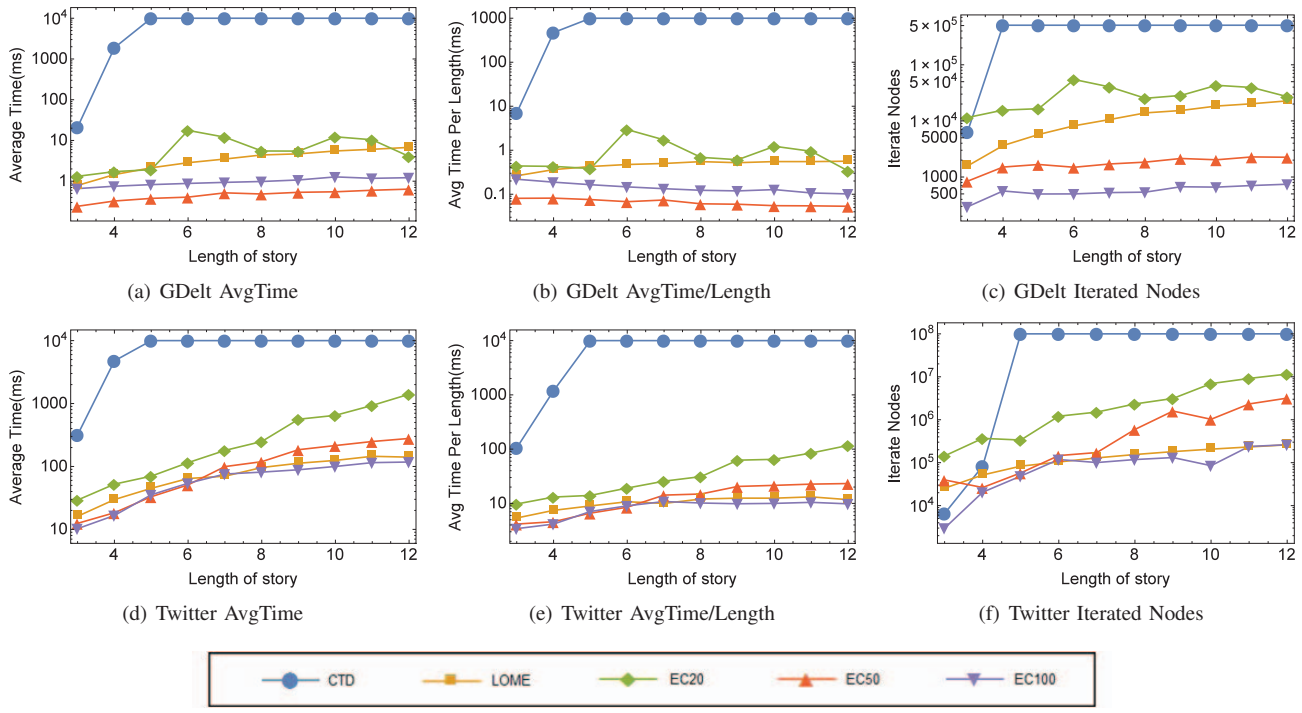
For εCluster hierarchy generation, we calculated the hierarchy building time by computing ten different layer hierarchies on ten different sized datasets. Figure 5(a) shows that the hierarchy building time increases linearly as the layer number of hierarchies increases from one to one hundred. Figure 5(b) shows that as the data size increases, the hierarchy building time also increases linearly in different εCluster layers from EC10 to EC100. The edges between entities are used to represent the complexity of entity similarity graph. A dataset from one thousand to one million edges is selected. Given that greater layer hierarchies are more efficient in large datasets, as shown in section VI-B, these properties allow us to choose more layers for the hierarchy as the data size increases, making the new εCluster hierarchy approach a very practical solution for large sized datasets.

The goal of εCluster story generation experiment is to assess the performance of different baseline algorithms. For this purpose, we use (i) Average Time, (ii) Average Time per length of story and (iii) the number of nodes explored to compare the performance of diverse methods. To compare these for the same story length, diverse lengths of story lines generated by all the algorithms from 10,000 randomly selected story pairs were collected, and categorized in terms of their story length, from 3 to 12. As Figure 6(a) and 6(b) show, the εCluster algorithm has the most efficient running time. The average runtime trends shown in Figure 6(a) approximately mirror the number of nodes explored in Figure 6(c) because running time depends on the number of nodes iterated. One exception to this rule is that even though EC100 iterates less nodes, it spends slightly more time than EC50 due to the additional time required to traverse between layers. Also, the performance results for the Twitter dataset are almost the same as for the GDelt dataset, except that more time is taken and more nodes are iterated. It is also worth mentioning that the εCluster with more layers performs faster than those with fewer layers as the size of the dataset increases; Figure 6(e) shows that EC100 is capable of processing the stories more rapidly than EC50.

C. Story Quality Evaluation

Despite the different path objectives of the story generation algorithms, we can still use them as baselines to numerically assess the story quality of the new εCluster algorithm. The pairwise entity edge similarity is determined using the three

[7]Purchased from Datasift Inc



(a) GDelt AvgTime (b) GDelt AvgTime/Length (c) GDelt Iterated Nodes

(d) Twitter AvgTime (e) Twitter AvgTime/Length (f) Twitter Iterated Nodes

Fig. 6. εCluster Performance Evaluation based on GDelt and Twitter data. (a) and (d) compare the average running time, (b) and (e) the average running time per story length and (c) and (f) the number of iterated nodes between the new algorithm and the baseline algorithms.

metrics discussed in section III: (i) MinEdge (ii)Average Edge and (iii) Dispersion Coefficient. Since the main purpose of our approach is to optimize MinEdge, the AvgEdge or the Dispersion Coefficient metrics are listed for reference purpose.

To evaluate the story using the three metrics, we used both the GDelt dataset and Twitter dataset and aimed to generate $10,000$ stories between randomly selected entity pairs. Table I depicts the results of the successful searches. As the table shows, the εCluster methods were competitive, outperforming the other methods in both data sets. Although the CTD method is designed to find an optimal MinEdge storyline, in practice a maximum story length that can be computed by our test machine is 5, making its MinEdge score lower than expected. Table I also shows that both EC50 and EC100 have higher MinEdge scores than EC20 in larger sized datasets due to their finer-grained cluster having less impact on the limitation on the number of searching steps within a cluster. Last but not least, εCluster algorithm also outperforms other methods in AvgEdge and the Dispersion Coefficient metrics. Although our method does not aim at optimizing these two metrics, the result shows that MinEdge can fairly improve both of them.

TABLE I
STORYLINE QUALITY COMPARISON.

	GDelt Dataset			*Twitter Dataset*		
	Min. Edge	Avg. Edge	Disp.	**Min. Edge**	Avg. Edge	Disp.
CTD	0.223	0.243	0.742	0.099	0.126	0.574
LOME	0.241	0.380	0.863	0.103	0.253	0.862
EC20	**0.312**	**0.461**	**0.899**	0.165	0.413	0.864
EC50	**0.312**	0.459	0.898	**0.168**	0.410	**0.866**
EC100	**0.312**	0.459	0.898	**0.168**	**0.414**	**0.866**

D. Case Study

During the experiment, a number of interesting story lines were observed using the proposed approach, one of which was from the Mexican presidential election in 2012. As reported[8], the election of Enrique Peña Nieto[9] was marred by media bias and voter fraud. A secret file[10] revealed an alleged record of fees apparently charged by Televisa, the largest Mexican multimedia company, for raising Peña Nieto's profile. To discover the relationship between Televisa and Peña Nieto, we generated a story line between them, shown in Figure 1, where the junction points are related features, namely tweets, links and common followers. The storyline constructed reveals several key persons related to the event: (i) *Laura Barranco*, a journalist working on the news team of Carlos Loret in Televisa, who revealed that Carlos confirmed the transaction between Televisa and Peña Nieto in an internal chat; and (ii) *Carlos Loret*, a popular news anchor in Mexico, who confirmed that Alejandro Quintero, the station's Corporate Vice President of Marketing, is linked to Peña Nieto in the scandal.

[8]http://www.theguardian.com/commentisfree/2012/jul/09/irregularities-reveal-mexico-election-far-from-fair

[9]Current president of Mexico, candidate of Institutional Revolutionary Party(PRI)

[10]http://www.theguardian.com/world/interactive/2012/jun/08/mexico-media-scandal-televisa-pena-nieto-claims

(iii) *Javier Duarte*, the Governor of Veracruz, is also linked with Carlos Loret because Carlos revealed that 11 journalists had been murdered during his administration since 2010. The event is categorized as *#MeDuelesVeracruz* in Twitter. (iv) *Luis Videgaray*, who serves as the Secretary of Finance and Public Credit in the cabinet of Enrique Peña Nieto, has the same ties as Peña Nieto with the company Higa Group, one of several companies winning government contracts. The case shows the ability of our system to help users extract hidden connections between entities from thousands of tweets, *#hashtags*, *@mentions*, and coherent locations, which is very time-consuming to handle it manually.

VII. Conclusion

This paper presents a novel approach to storytelling in large Twitter data sets that enables analysts to gain deep insights into individual entities and their relationships. The new entity similarity model uncovers the underlying relationships between entities based on their heterogeneous features in Twitter, supported by a hierarchical cluster routing algorithm that generates storylines with competitive performance and quality. The extensive experimental results for the various datasets tested clearly demonstrate the effectiveness and efficiency of our new approach through a comparison with four state-of-the-art methods.

References

[1] Raimundo F Dos Santos Jr, Sumit Shah, Feng Chen, Arnold Boedihardjo, Patrick Butler, Chang-Tien Lu, and Naren Ramakrishnan. Spatio-temporal storytelling on twitter. 2013.

[2] Liang Zhao, Feng Chen, Jing Dai, Ting Hua, Chang-Tien Lu, and Naren Ramakrishnan. Unsupervised spatial event detection in targeted domains with applications to civil unrest modeling. *PLoS ONE*, 9(10):e110206, 10 2014.

[3] M. Shahriar Hossain, Patrick Butler, Arnold P. Boedihardjo, and Naren Ramakrishnan. Storytelling in entity networks to support intelligence analysts. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 1375–1383, New York, NY, USA, 2012. ACM.

[4] EricA. Bier, EdwardW. Ishak, and Ed Chi. Entity workspace: An evidence file that aids memory, inference, and reading. In Sharad Mehrotra, DanielD. Zeng, Hsinchun Chen, Bhavani Thuraisingham, and Fei-Yue Wang, editors, *Intelligence and Security Informatics*, volume 3975 of *Lecture Notes in Computer Science*, pages 466–472. Springer Berlin Heidelberg, 2006.

[5] Hyunmo Kang, C. Plaisant, Bongshin Lee, and B.B. Bederson. Netlens: Iterative exploration of content-actor network data. In *Visual Analytics Science And Technology, 2006 IEEE Symposium On*, pages 91–98, Oct 2006.

[6] Dafna Shahaf and Carlos Guestrin. Connecting the dots between news articles. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pages 623–632, New York, NY, USA, 2010. ACM.

[7] M. Shahriar Hossain, Joseph Gresock, Yvette Edmonds, Richard Helm, Malcolm Potts, and Naren Ramakrishnan. Connecting the dots between pubmed abstracts. *PLoS ONE*, 7(1):e29509, 01 2012.

[8] Christos Faloutsos, Kevin S. McCurley, and Andrew Tomkins. Fast discovery of connection subgraphs. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 118–127, New York, NY, USA, 2004. ACM.

[9] Dafna Shahaf and Carlos Guestrin. Connecting two (or less) dots: Discovering structure in news articles. *ACM Trans. Knowl. Discov. Data*, 5(4):24:1–24:31, February 2012.

[10] Xianshu Zhu. *Finding Story Chains and Story Maps in Newswire Articles*. PhD thesis, Catonsville, MD, USA, 2013. AAI3610054.

[11] M Andrea Rodríguez and Max J Egenhofer. Determining semantic similarity among entity classes from different ontologies. *Knowledge and Data Engineering, IEEE Transactions on*, 15(2):442–456, 2003.

[12] Rubén Tous and Jaime Delgado. A vector space model for semantic similarity calculation and owl ontology alignment. In *Database and Expert Systems Applications*, pages 307–316. Springer, 2006.

[13] Petko Bogdanov and Ambuj Singh. Accurate and scalable nearest neighbors in large networks based on effective importance. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 1009–1018. ACM, 2013.

[14] Dan Klein and Christopher D Manning. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 478. Association for Computational Linguistics, 2004.

[15] Quannan Li, Yu Zheng, Xing Xie, Yukun Chen, Wenyu Liu, and Wei-Ying Ma. Mining user similarity based on location history. In *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*, page 34. ACM, 2008.

[16] Josh Jia-Ching Ying, Eric Hsueh-Chan Lu, Wang-Chien Lee, Tz-Chiao Weng, and Vincent S Tseng. Mining user similarity from semantic trajectories. In *Proceedings of the 2nd ACM SIGSPATIAL International Workshop on Location Based Social Networks*, pages 19–26. ACM, 2010.

[17] Ashish Goel, Aneesh Sharma, Dong Wang, and Zhijun Yin. Discovering similar users on twitter. In *11th Workshop on Mining and Learning with Graphs*, 2013.

[18] Xiaojun Wan and Jianwu Yang. Multi-document summarization using cluster-based link analysis. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 299–306. ACM, 2008.

[19] Wen-tau Yih, Joshua Goodman, Lucy Vanderwende, and Hisami Suzuki. Multi-document summarization by maximizing informative content-words. In *IJCAI*, volume 7, pages 1776–1782, 2007.

[20] Beaux Sharifi, Mark-Anthony Hutton, and Jugal Kalita. Summarizing microblogs automatically. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 685–688. Association for Computational Linguistics, 2010.

[21] David Inouye and Jugal K Kalita. Comparing twitter summarization algorithms for multiple post summaries. In *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on*, pages 298–306. IEEE, 2011.

[22] Sanda Harabagiu and Andrew Hickl. Relevance modeling for microblog summarization. In *Fifth International AAAI Conference on Weblogs and Social Media*, 2011.

[23] Hiroya Takamura, Hikaru Yokono, and Manabu Okumura. Summarizing a document stream. In *Advances in Information Retrieval*, pages 177–188. Springer, 2011.

[24] Lidan Shou, Zhenhua Wang, Ke Chen, and Gang Chen. Sumblr: continuous summarization of evolving tweet streams. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 533–542. ACM, 2013.

[25] HJW De Baar. von liebig's law of the minimum and plankton ecology (1899–1991). *Progress in Oceanography*, 33(4):347–386, 1994.

[26] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *In ACL*, pages 363–370, 2005.

[27] Heeyoung Lee, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. Stanford's multi-pass sieve coreference resolution system at the conll-2011 shared task. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 28–34. Association for Computational Linguistics, 2011.

[28] Mark Dredze, Michael J Paul, Shane Bergsma, and Hieu Tran. Carmen: A twitter geolocation system with applications to public health. In *AAAI Workshop on Expanding the Boundaries of Health Informatics Using AI (HIAI)*, pages 20–24. Citeseer, 2013.

[29] Thomas D'Roza and George Bilchev. An overview of location-based services. *BT Technology Journal*, 21(1):20–27, 2003.

[30] Muhammed A. Yildirim and Michele Coscia. Using random walks to generate associations between objects. *PLoS ONE*, 9(8):e104813, 08 2014.