StreamCube: Hierarchical Spatio-temporal Hashtag Clustering for Event Exploration over the Twitter Stream

Wei Feng[†], Chao Zhang[‡], Wei Zhang[†], Jiawei Han[‡], Jianyong Wang[†], Charu Aggarwal[§], Jianbin Huang[♮]

[†] *Tsinghua National Laboratory for Information Science and Technology (TNList)*
Department of Computer Science and Technology, Tsinghua University, Beijing, China
[‡] *Department of Computer Science, University of Illinois at Urbana-Champaign, IL, USA*
[§] *IBM T. J. Watson Research Center, NY, USA*
[♮] *Department of Computer Science, Xidian University, Xi'an, China*
[†] {feng-w10,zweil1}@mails.tsinghua.edu.cn, jianyong@tsinghua.edu.cn
[‡] czhang82@illinois.edu, hanj@cs.uiuc.edu, [§] charu@us.ibm.com, [♮] jbhuang@xidian.edu.cn

***Abstract*—What is happening around the world? When and where? Mining the geo-tagged Twitter stream makes it possible to answer the above questions in real-time. Although a single tweet can be short and noisy, proper aggregations of tweets can provide meaningful results. In this paper, we focus on hierarchical spatio-temporal hashtag clustering techniques. Our system has the following features: (1) Exploring events (hashtag clusters) with different space granularity. Users can zoom in and out on maps to find out what is happening in a particular area. (2) Exploring events with different time granularity. Users can choose to see what is happening today or in the past week. (3) Efficient single-pass algorithm for event identification, which provides human-readable hashtag clusters. (4) Efficient event ranking which aims to find burst events and localized events given a particular region and time frame. To support aggregation with different space and time granularity, we propose a data structure called StreamCube, which is an extension of the data cube structure from the database community with spatial and temporal hierarchy. To achieve high scalability, we propose a divide-and-conquer method to construct the StreamCube. To support flexible event ranking with different weights, we proposed a top-k based index. Different efficient methods are used to speed up event similarity computations. Finally, we have conducted extensive experiments on a real twitter data. Experimental results show that our framework can provide meaningful results with high scalability.**

I. Introduction

With 500 million tweets (messages) posted everyday, Twitter has become one of the leading social media services around the world. In Twitter, users can post short tweets, which are limited to 140 characters, to share 'what is happening' with their followers. Meanwhile, interesting tweets can be retweeted (re-posted) to propagate further in the social network.

The tweet stream can be considered as an up-to-date news sources of the physical world. For example, during the United States presidential election of 2012, the Twitter Political Index was used to measure users' sentiments towards the candidates. By treating Twitter users as human sensors, [1] can detect earthquakes in real-time by monitoring earthquake related tweets. Furthermore, it is shown that the up and down changes

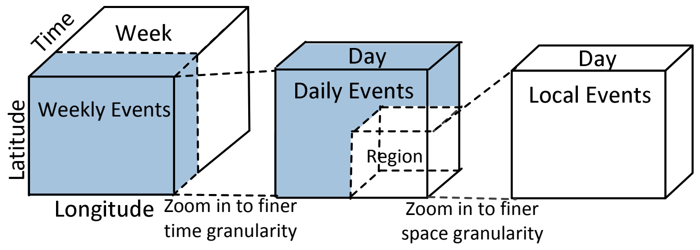


Fig. 1. Zoomable Event Cube

in stock market are correlated with users' moods in Twitter [2].

Different from plain text, tweets contain rich attributes, such as: (1) Geographical Information. Thanks to GPS-enabled smart phones, more and more tweets are not only timestamped but also geotagged. Also, it is common to find location information in user profiles. (2) Hashtags. A hashtag is a word or phrase preceded by '#', and is used to identify messages on a specific topic. For example, '#Election2012' can be used to indicate the topic of the United States presidential election of 2012.

In this paper, we propose a novel framework for hierarchical spatio-temporal hashtag clustering over the Twitter stream, which aims to help users explore the Twitter data interactively. An event is considered as a hashtag cluster. For example, the United States presidential election of 2012 can be represented by hashtag cluster {'#Election2012', '#Obama', '#Romney'}, where '#Obama' and '#Romney' represent two main candidates in the campaign respectively. As shown in Figure 1, we organize hashtag clusters into data cubes according to their timestamps and geographical information. Different from traditional data cubes from database literature [3], our framework constructs data cubes from tweet stream in real-time. Thus our framework is called StreamCube. StreamCube has the following features:

- Exploring events in different time granularity. Depending on the data analysis purpose, different users may have

different requirements on the time granularity. If we want to summarize the top-10 events of a year, we may choose year as the basic time granularity. In this case, the recency of an event is not very important. However, if we want to detect emerging events from the stream, we may choose a finer granularity.
- Exploring events in different space granularity. If we focus on the world-wide breaking events, we may choose global space as our basic space granularity. If we focus on localized news, we may choose district or city as our space granularity.
- Detecting burst events and localized events. Given a time frame (e.g., today), users may want to know what events break out during that time. Given a region (e.g., the current region shown on the Google map), users may want to know the local events with respect to this particular region. Finally, given a time frame and a region, we can also find the burst localized events.

Hierarchical spatio-temporal hashtag clustering over the Twitter stream faces many challenges:

- Efficient single-pass hashtag clustering. In order to provide real-time event exploration, the clustering algorithm should avoid iterative computation. The clustering results should be updated in an incremental way. Traditional clustering models assume data points are static. However, the content of hashtags are evolving and similar hashtags may become dissimilar as new tweets come in. For example, hashtag '#Obama' is closely related to '#Election2012' during the election but will be related to other hashtags as time goes by.
- Efficient ranking algorithm with a clear semantic definition on burstiness and localness in the context of the data cube. There are thousands of events happening every hour from different places. However, given a particular time frame (e.g., today) and a region the user is interested in, events that occurred in the given time frame from the chosen region are more valuable than others. Finding burst localized events has not been well studied before.
- Merging new data with historical data incrementally. Suppose we already have identified events for each day of the week. To get a whole picture of the events in the week, we need to make sure that the results of each day can be merged incrementally with little cost. In this way we can avoid computing from the the whole week's data.

StreamCube consists of three components: (1) Spatial-temporal aggregation (Section III), which organizes tweets according to the spatial and temporal hierarchy. The hierarchy enables users to explore hashtag clusters in different time and space granularity. Moreover, clustering results can be merged incrementally according to the defined hierarchy. (2) A Single-pass hashtag clustering algorithm (Section IV). The algorithm is designed to handle content-evolving hashtags and provide real-time results as tweets arrive. (3) Event ranking (Section V). Given a particular region and a time frame, we aim to find the localized events and burst events.

To summarize, we make the following contributions.
- To the best of our knowledge, we are among the first to study hierarchical spatio-temporal hashtag clustering techniques over the Twitter stream. This is a novel application and it aims to provide real-time interactive exploration of Twitter data.
- We develop an efficient one-pass hashtag clustering algorithm, which can handle content-evolving hashtags in an online manner.
- Given a particular time span and region, we have defined the semantics of burst localized events and propose an efficient method for event ranking.
- Inspired by data cubes, we have proposed a structure for aggregation with time hierarchy and space hierarchy, which can merge newly generated results with historical results in an incremental way.

The rest of this paper is organized as follows. We introduce the overall framework in Section III. In Section IV, we propose a one-pass event identification algorithm. In Section V, we study how to rank events to discover burst localized events. Experimental results are provided in Section VI. The related work is covered in Section II. Finally, we conclude the paper in Section VII.

II. Related Work

Data Cube. Our work is mainly inspired by the data cube [3] from the database literature, which is used for organizing and exploring multidimensional data with operations like slice, dice, drill up/down. Our work can be considered as an extension of the traditional data cube for Twitter data. There are several differences: (1) StreamCube is constructed incrementally over the Twitter stream according to a time hierarchy and a space hierarchy. (2) The semantic meaning of each cube is different. We aim to provide human-readable events (i.e., clusters of hashtags). (3) Given a cube with respect to a particular time frame and region, we aim to provide a meaningful ranking (finding burst events and local events), which is not considered in the traditional data cube.

Clutering Techniques for Social Media. Clustering techniques for social media has attracted many researchers recently [4]. Existing work on clustering techniques for social media can be classified into three categories: (1) Tweet-based clustering [5]–[9], which is an extension of the traditional text clustering. Different from high quality news, tweets are short and noisy. They either argument the tweets with external knowledge base [5], term relatedness [6], or propose a more advanced similarity measurement [8]. (2) Burst-keyword-based clustering [1,10]. These work focus on the burst keywords instead of single tweets. However, this will only cover partial events with high burstiness. (3) Hashtag-based clustering. The common drawback of tweet-based clustering and burst-keyword-based clustering is that their results are less human readable. Our work defines an event as a cluster of hashtags, which are more expressive since they are user-provided keywords. Moreover, our hashtag-based clustering can overcome the drawbacks of noisy tweets in a more elegant

TABLE I
COMPARISON WITH EXISTING WORK

Method	SUMBLR [7]	TMONITOR [10]	SMCA [30]	Ours
Tweet Level	√	√		
Hashtag Level			√	√
Zoomable	√			√
Interpretable		Partial	√	√
Ranking		Partial		√

way. To the best of our knowledge, hashtag clustering has not been well studied. The most relevant work are [11] and [12]. However, both of them consider hashtags as static documents while our work deals with content-evolving hashtags.

Event Ranking. We are particularly interested in finding burst events and local events. [13] proposes a sketch-based approach for detecting localized events. [14] also employ the sketch structure for burst detection. Given a term, [15,16] identify its bursting time interval and region in polynominal time. The common drawback of these work is that they aim to find burst patterns or localized patters from the whole dataset, where only a fraction of events can be detected. In contrast, our work aims to give an overall ranking for all the events in the chosen cube w.r.t. to a particular time frame and region.

Other Applications. Although not directly related, our work is inspired by many other applications in social media. Location-based social network [17]–[19] connects users based on their locations, where different variations of spatio-textual index are studied. Information visualization [20]–[22] helps users to explore Twitter events interactively, where different ways to organize tweets are proposed. Real-time search [23]–[26] studies how to incrementally update the index to keep up with the Tweet stream, where recent tweets are preferred. Spatio-temporal topic modeling [27]–[29] employs probabilistic graphical models to explain the relation among location, time and topics. Recently, [21] has proposed a framework for visualizing stories extracted from a large corpus of documents with a complex offline optimization method. However, it cannot be applied to the real-time Twitter stream.

The difference between our work and existing work is summarized in Table I. Firstly, existing work can be classified based on the clustering objects. SUMBLR [7] and TMONITOR [10] are designed to cluster short and noisy tweets, where data points are static. SMCA [30] and our method are hashtag-based clustering algorithms, where the content of the data points keeps changing as tweets arrive. The advantage of hashtag clustering algorithms is that the results (hashtag clusters) are human-interpretable. Moreover, thanks to the spatial-temporal hierarchy, our method supports zooming in and zooming out with different time and space granularity. Finally, our framework has a ranking component to detect local events and burst events since it would be too hard for human to keep track of all the clusters. All the methods listed in Table I are used as baselines in our experimental study.

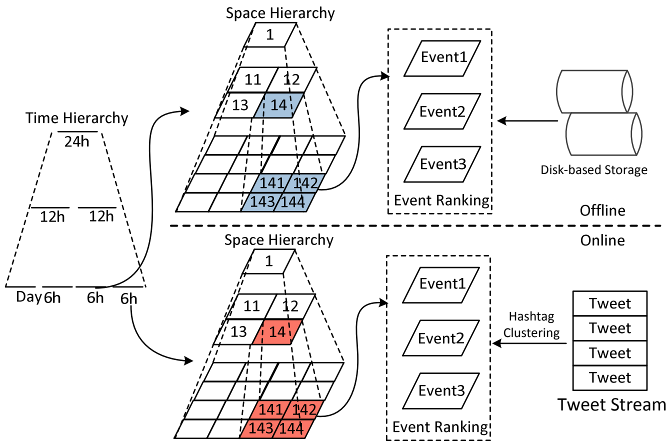


Fig. 2. Framework

III. THE OVERALL FRAMEWORK

The overall framework is shown in Figure 2. STREAMCUBE has three components: (1) spatial-temporal aggregation, (2) hashtag clustering, and (3) event ranking. As shown in Figure 2, events are organized according to a space and a time hierarchy. In this case we can provide aggregation results immediately when a user is exploring events with different time and space granularity.

Space Hierarchy. We use a quad-tree like structure to organize the global space into hierarchy. (1) The top level represents the global space. (2) The global space is divided into four smaller regions with equal size at the second level. (3) For each region at the second level, it is further split into four sub-regions with equal size at the third level.

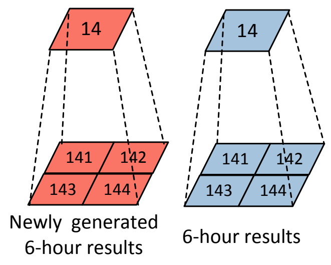
This space hierarchy has a nice support for exploring Twitter data on a zoomable digital map like Google Map or Bing Map, since their map tile system also uses this hierarchy. However, it is also possible to organize the space into a hierarchy of countries, states, cities, and districts.

Time Hierarchy. Similar to the space hierarchy, the time hierarchy is defined as follows: (1) The top level corresponds to the coarsest granularity, which is one day in the example shown in Figure 2. (2) The second level has two 12-hour time frames. (3) Then each 12-hour time frame is further split into two 6-hour time frame. The splitting rule applies for the rest of the levels.

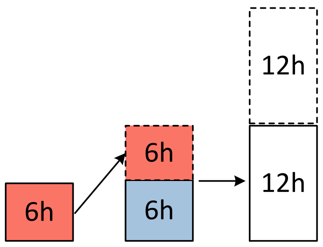
Connecting Time Hierarchy to Space Hierarchy. Each time frame has a nested space hierarchy. For example, we can see from Figure 2 that each 6-hour time frame has a nested space hierarchy. In this way, given a fixed time frame, users can explore different events with different space granularity. Note that we only need to keep events from the last 6 hours in memory for increment updates. Historical data are fixed and flushed into disk-based storage.

With the knowledge of the time hierarchy and space hierarchy, now we introduce our spatial-temporal aggregation algorithm. The pseudo-code is shown in Algorithm 1, which mainly has three steps:

Construct new cubes (*lines* 1-4). For each tweet d from the current time frame, we first assign the tweet to its cube



(a) Spatial Merge



(b) Temporal Merge

Fig. 3. Spatial-temporal Aggregation

from the lowest level of space hierarchy according to its geographical information (*line* 1). Note that each region can have an independent working thread. Thus all the regions from the lowest level can be updated in parallel.

For the chosen cube (*line* 2), we call `hashtag-clustering` method (*line* 3), which either changes the content of an existing event or create a new event. In either case, we call `update-event-ranking` method (*line* 4) to update event ranking scores.

Spatial merge (*lines* 5-7). This step merges the newly generated small cubes into large cubes according to the space hierarchy. We illustrate this process with Figure 3a. In Figure 3a, region #14 has four children (region #141, #142, #143, and #144). Each of the children contains newly generated event lists for the past 6 hours. By calling `spatial-merge` (*line* 7), the events from regions #141, #142, #143, and #144 are combined into events in region #14. The challenge is how to define the semantics of merging events. For example, region #141 has an event a, and #142 has two events b and c. It is not clear whether we should merge a with b or with c, or keep them as separate events.

Temporal merge (*lines* 8-10). This method merges the newly generated small cubes into large cubes according to the time hierarchy. We will illustrate this process with Figure 3b. Starting with the regions from the lowest level of the space hierarchy, we have newly generated red-colored region #141, #142, #143, and #144, temporal aggregation will merge them with blue region #141, #142, #143, and #144, respectively. Thus from two 6-hour cubes we can get a 12-hour cube, as shown in the second bucket of Figure 3b. The newly generated 12-hour cube will be stored in the third bucket, thus we need another 12-hour cube to get a 24-hour cube. The same rule applies to red #14 and blue #14 from the upper level.

Algorithm 1: `Spatial-temporal Aggregation`

```
Input: $D=\{d_1, d_2, ..., d_n\}$: tweet stream from the current
       time frame $t$
Output: The updated StreamCube
1  for each tweet $d \in D$ do
2      $cube \leftarrow cubes[t][d.region]$
3      $cube$.hashtag-clustering($d$)
4      $cube$.update-event-ranking()
5  for each space level $sl$ from bottom to top do
6      for each region at space level $sl$ do
7          $cubes[t][region] \leftarrow$ spatial-merge($cube$,
           $cube.children()$)
8  for each space level $sl$ from bottom to top do
9      for each region at space level $sl$ do
10         $cubes[2t][region] \leftarrow$
           temporal-merge($cubes[t-1][region]$,
           $cubes[t][region]$)
```

TABLE II
FREQUNTLY CO-OCCURRED HASHTAGS

Hashtag	Frequently Co-occurred Hashtags
NBA	Heat Lakers Knicks Bulls Celtics
GRAMMYs	Grammys2014 lorde daftpunk eredcarpet
Christmas	xmas santa love family holidays

IV. Hashtag Clustering

Before aggregating events according to the spatial-temporal hierarchy, hashtag clustering is performed at the lowest level of the hierarchy. Different from existing tweet clustering algorithms, we treat hashtags as basic data points for clustering. First we introduce the concept of hashtags.

Definition 1 (Hashtag)**.** *A hashtag is a word or phrase preceded by '#', and is used to identify messages on a specific topic.*

Example. We randomly picked an example tweet discussing the United States presidential election of 2012. "Why did Mitt Romney lose the presidential election? Here is a roundup of conservative commentary. #Election2012", where hashtag '#Election2012' is used to indicate the topic of the message.

Hashtags play important roles in Twitter and have many advantages: (1) Hashtags are human-readable keywords and less noisy. Only user consented hashtags can become popular. (2) Hashtags, when represented by its tweets, can be seen as long documents and are easier to find their semantic relatedness, which can lead to good clustering results.

With the knowledge of hashtags, we now give a formal definition of events.

Definition 2 (Event)**.** *An event is a group of hashtags that focus on the same topic.*

Example. The United States presidential election of 2012 can be described with hashtags '#Election2012', '#Obama', and '#Romney', where '#Obama' and '#Romney' represent two main candidates in the campaign.

A. Hashtag Representation and Similarity

Hashtags can have different representations in different contexts.

A Hashtag as a bag of words. Given a hashtag h, it can be represented as a bag of words, which is an aggregation of all the tweets that contain h. Although it is hard to determine which word is important in a single tweet, it is possible to find out the important words for a hashtag. For example, on January 26, 2014, hashtag '#GRAMMYs', which refers to the grammy awards, became widely adopted by Twitter users. The top words all refer to the popular musicians in the awards.

Formally, let W denote all the words, hashtag **h** is represented as a normalized weighted vector

$$\mathbf{h}_{word} = (w_1, w_2, ..., w_{|W|}) \tag{1}$$

where w_i is the weight of the i-th word and $||\mathbf{h}_{word}|| = 1$.

A Hashtag as a bag of hashtags. According to our dataset, 55% hashtags have co-occurred with other hashtags. Co-occurred Hashtags are user-provided examples for hashtag clusters. For example, we have a tweet that says 'Five lessons for the UK from Obama′s victory, #Obama #Romney #Election2012.'. This tweet contains three hashtags, i.e., '#Obama', '#Romney', and '#Election2012'. It indicates that these hashtags are related to each other. We have listed more examples in Table II.

Formally, let H denote the hashtag set, hashtag **h** is represented as a normalized weighted vector

$$\mathbf{h}_{tag} = (h_1, h_2, ..., h_{|H|}) \tag{2}$$

where h_i is the weight of the i-th hashtag and $||\mathbf{h}_{tag}|| = 1$.

With the above knowledge, we introduce our hashtag representation and event representation.

Hashtag Representation and Similarity. Let $\mathbf{h}^i_{word}$ and $\mathbf{h}^i_{tag}$ denote the word vector and hashtag vector of the i-th hashtag $\mathbf{h}_i$. Given two hashtag $\mathbf{h}_1$ and $\mathbf{h}_2$, their similarity is defined as

$$\begin{aligned} sim(\mathbf{h}_1, \mathbf{h}_2) &= \alpha \cdot cos(\mathbf{h}^1_{word}, \mathbf{h}^2_{word}) + \beta \cdot cos(\mathbf{h}^1_{tag}, \mathbf{h}^2_{tag}) \\ &= \alpha \frac{\mathbf{h}^1_{word}, \mathbf{h}^2_{word}}{||\mathbf{h}^1_{word}|| \cdot ||\mathbf{h}^2_{word}||} + \beta \frac{\mathbf{h}^1_{tag}, \mathbf{h}^2_{tag}}{||\mathbf{h}^1_{tag}|| \cdot ||\mathbf{h}^2_{tag}||} \\ &= \alpha \sum_{i=1}^{|W|} w^1_i w^2_i + \beta \sum_{i=1}^{|H|} h^1_i h^2_i \\ &= (\alpha^{\frac{1}{2}} \mathbf{h}^1_{word}, \beta^{\frac{1}{2}} \mathbf{h}^1_{tag}) \cdot (\alpha^{\frac{1}{2}} \mathbf{h}^2_{word}, \beta^{\frac{1}{2}} \mathbf{h}^2_{tag}) \end{aligned} \tag{3}$$

where α and β are user-specified weights and $\alpha + \beta = 1$. We set β to 0.7 in this paper. From the above equation, we can see that a hashtag can be represented as a vector, i.e.,

$$\mathbf{h} = (\alpha^{\frac{1}{2}} \mathbf{h}_{word}, \beta^{\frac{1}{2}} \mathbf{h}_{tag})$$

Event Representation and Similarity. Recall that an event is a group of hashtags. Thus events are represented in the same way with hashtags, i.e.,

$$\mathbf{e} = (\alpha^{\frac{1}{2}} \mathbf{e}_{word}, \beta^{\frac{1}{2}} \mathbf{e}_{tag})$$

Different from tweets, the content of a hashtag is evolving with more tweets coming in. In other words, similar hashtags may become not so similar as time goes by. This brings us a new problem: **how to cluster content-evolving hashtags**. We will address the problem in two steps: (1) We relax the problem by assuming hashtags are static, and discuss how to cluster a stream of hashtags in Section IV-B. (2) We introduce how to cluster content-evolving hashtags from the tweet stream in Section IV-C.

Algorithm 2: HASHTAG-CLUSTER-STATIC(E, h)

```
Input: Event set $E$={$e_1$, $e_2$, ..., $e_k$}
Hashtag $h$
Output: Updated event set $E$
1 $e$ = nearest-neighbor($E$, $h$)
2 if sim(e, h) > e.threshold then
3     add $h$ to $E$ as a new event
4 else
5     add $h$ to the existing event $e$
```

Algorithm 3: NEAREST-NEIGHBOR(E, h)

```
Input: Event set $E$={$e_1$, $e_2$, ..., $e_k$}
Hashtag $h$={$w_1$, $w_2$, ..., $w_n$}
Output: Nearest neighbor $e$
1 for each word $w_i$ order by max_partial($w_i$) do
2     for each event $e$ in the invert list of $w_i$ do
3         sim(h, e) += $w^h_i$ * $w^e_i$
4     Let $e^1$ denote the most similar event
5     Let $e^2$ denote the second similar event
6     if sim(h, $e^1$) > sim(h, $e^2$) + $\sum_{j=i+1}^{n} max\_partial(w^e_j) \cdot w^h_j$ then
7         return $e^1$
```

B. Clustering Static Hashtags

We have the following considerations when designing the clustering algorithm: (1) Free from choosing the number of clusters. Since new events can emerge at any time from the hashtag stream, we cannot know the number of clusters (events) in advance. (2) Avoid iterative computation. Since we need to do clustering with one pass of data, iterative loops over the data should be avoided.

With the above considerations, we proposed the `HASHTAG-CLUTER-STATIC` algorithm, which is inspired by the `CluStream` algorithm proposed in [31]. As shown in Algorithm 2, the algorithm has two steps:

- Locate the Nearest Neighbor ($line$ 1). Given a hashtag h to be clustered, we first locate its nearest neighbor in the current event list, which is the best candidate for absorbing the hashtag.
- Check the Absorbing Condition ($line$ 2). Given the nearest neighbor e, we check whether hashtag h is similar enough to be absorbed into the e based on an e-related threshold ($e.threshold$ at $line$ 2).

Now we introduce how to speed up the nearest neighbor computation and how to choose a reasonable threshold.

Fast Nearest Neighbor Computation. A naive implementation is to compute a similarity score between the new hashtag and each of the existing clusters. The time complexity is O(kd), where k is the number of clusters and d is the number of dimensions. Recall that frequently co-occurred hashtags are user provided examples for clustering. So we use the following

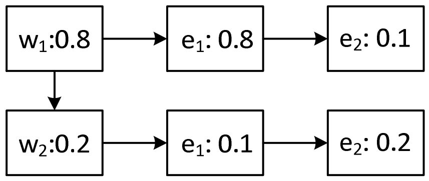


Fig. 4. Nearest Neighbor Search

heuristic to reduce the number of candidates.

HEURISTIC 1. *(Candidate Pruning by Co-occurrence) A candidate can be pruned for nearest neighbor search if it has no co-occurrence with the current hashtag.*

According to our dataset, a hashtag has co-occurred with 9 hashtags on average. Thus this pruning rule can significantly reduce the candidate size.

However, popular hashtags still co-occurr with many hashtags. For example, '#GRAMMYs' co-occurs with more than three thousand hashtags. This motivates us to find more advanced pruning techniques. Remind that hashtags are represented as weighted vectors and the cosine similarity score is the addition result of many partial scores (see Equation 3). By treating the current hashtag as a query, and the existing k clusters as indexed documents, we can use an inverted index to speed up the computation, which is widely used in information retrieval. Particularly, given a hashtag h, we adopt TAAT (Term-at-a-time) [32] to comput the nearest neighbor. The pseudo code is shown in Algorithm 3. We will introduce this algorithm with an example shown in Figure 4. Suppose we have two events $e_1 = (0.8, 0.1)$ and $e_2 = (0.1, 0.2)$, and let w_1 and w_2 denote the first and the second dimension, respectively. Suppose the current hashtag is $h = (0.8, 0.2)$. Scanning the first row in the inverted index, we can get $sim(h, e_1) = 0.64$ and $sim(h, e_2) = 0.08$ (*lines* 1-3). We know that the max partial value for the second row in the inverted index is 0.2. Since we have $sim(h, e_1) > sim(h, e_2) + 0.2 * 0.2$, we can conclude that e_1 is the nearest neighbor.

Absorbing Condition. The minimum threshold for event e is the nearest distance between e and any other clusters. Suppose the most similar event is e'. If the $sim(h, e')$ is smaller than $sim(e, e')$, it is better to treat hashtag h as a new cluster, since e and e', even with a bigger similarity score, belong to two different clusters.

C. Clustering Content-Evolving Hashtags

In this section, we discuss how to cluster content-involving hashtags. We have the following considerations: (1) Only cluster hashtags mentioned by enough number of tweets. When a new hashtag emerges, it only has few tweets. Under this situation, the hashtag can be an outlier, in which case it may be an personal event-irrelevant hashtag, or a spamming hashtag. It is better to keep a hashtag as a cluster alone instead of merging it into existing clusters. (2) Adjust clustering results as the content of hashtag evolves. Two similar hashtags may become dissimilar, and two isolated hashtags may become similar, since the tweets are affecting hashtag content continuously.

Algorithm 4: HASHTAG-CLUSTER-DYNAMIC(E, d)

```
Input: Event set $E$={$e_1$, $e_2$, ..., $e_k$}
       Tweet $d$
Output: Updated event set $E$
1  if $d.hashtag$ not in $E$ then
2      add $d.hashtag$ to as a new event to $E$ with an inactive status
3  else
4      $e \leftarrow$ FIND-EVENT($d.hashtag$)
5      update $e$ with to incorporate new tweet $d$
6      if $e.status$ is still inactive then
7          do nothing
8      else if $e.state$ transfers to active status then
9          HASHTAG-CLUSTER-STATIC($E$, $h$)
10     else
11         if $h.count$ reach check point then
12             $e' \leftarrow e \setminus h$
13             if $sim(h, e') > e'.threshold$ then
14                 HASHTAG-CLUSTER-STATIC($E$, $h$)
15         if $e.count$ reach check point then
16             HASHTAG-CLUSTER-STATIC($E$, $e$)
```

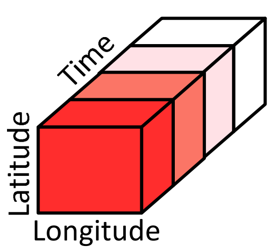
We should provide 'split' and 'merge' operations to adjust the clustering results.

With the above considerations, we designed the HASHTAG-CLUSTER-DYNAMIC algorithm as shown in Algorithm 4 to cluster content-evolving hashtags. It has the following parts:

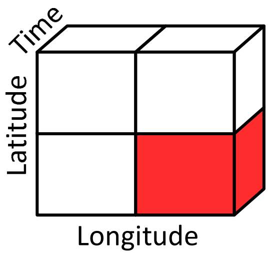
Handling new hashtags (*lines* 1-2). As discussed before it is the best to leave a new hashtag as an isolated cluster. The hashtag stays in an inactive status until it is supported by enough tweets.

Handling inactive events (*lines* 4-9). First, we update the event to incorporate the new tweet (*lines* 4-5). Second, we check the status of the updated event. If the event still does not get enough tweets, we keep the event isolated (*lines* 6-7). If the event's status becomes active after the update, we treat it as a static hashtag and call HASHTAG-CLUSTER-STATIC to cluster it (*lines* 8-9). In our system, a hashtag has to be mentioned by more than 30 tweets to become active.

Handling active events (*lines* 11-16). If the event is already active, we check whether this update can lead to (1) a split operation (*lines* 11-14), which remove the event from the cluster and re-clusters it, or (2) a merge operation (*lines* 15-16), which tries to merge the cluster with other clusters. Note that it is impractical to check split and merge condition on every update, since a single tweet has limited impacted on the hashtag content. A more efficient way is to check split and merge condition at certain interval. We define the check point as every thirty updates.



(a) Burstiness



(b) Burstiness

Fig. 5. Localness

D. Spatial-temporal Aggregation

Spatial aggregation from low level to higher level is equivalent to zooming out on maps. By time aggreagtion, users can see a week's event report instead of day's event report. The merge operation is the key in spatial-temporal aggregation.

Suppose we want to merge two events e_1, and e_2 from event sets E_1 and E_2, respectively. We have following cases when merging e_1 and e_2:

Case 1: e_1 is equal to e_2. In other words, they are represented by the same group of hashtags. Thus we can directly merge e_1 with e_2 based on the hashtag.

Case 2: e_1 has no overlap with e_2, i.e., they are represented by different groups of hashtags. Since e_1 and e_2 either come from different time frames or different regions, it is very likely that e_1 and e_2 correspond to different events.

Case 3: e_1 and e_2 are partially overlapped. In this case, we merge the hashtags in common. Then we check whether we can absorb the rest of the hashtags. For example, suppose $e_1 = (h_1, h_2)$ and $e_2 = (h_2, h_3)$. First, we merge h_2 and use it as the core. Then we calculate whether h_1 and h_3 are within the threshold of h_2 to check whether we can absorb h_1 and h_3. If h_1 or h_3 cannot be merged, they will be re-clustered.

V. Event Ranking

Given a particular region and a time frame, event ranking helps users find localized events in the region and burst events in the time frame.

A. Ranking Factors

An event can be ranked from different perspectives. We consider three measurements in this paper: (1) popularity, (2) burstiness, and (3) localness. In the following, we give formal definitions and illustrate the concepts in Figure V-A.

Popularity. Given a cube with respect to region r and frame t, the popularity of event e, i.e., $pop(e)$, is defined as

$$popularity(e) = \frac{freq(e)}{N} \tag{4}$$

where $freq(e)$ is the frequency of e in this cube measured by the number of tweets, and N is the total number of tweets in the cube. We use the normalized frequency because different time frame (weekday and weekends) are not equally active. So does different regions (Twitter may be less popular in some countries).

Burstiness. Let (p_{t-k}, p_{t-k+1}, ..., p_{t-1}) denote the popularity of the event in previous k time frames before t, where f_i denotes the popularity of the event in the i-th time frame. The burstiness is defined as

$$burstiness(e) = \frac{p_t - \mu}{\sigma} \tag{5}$$

where μ and σ are the mean and the standard deviation of (p_{t-k}, p_{t-k+1}, ..., p_{t-1}), respectively. As shown in Figure 5a, we model the popularity history as a gaussian distribution. The burstiness of an event is measured by how far away the popularity goes up compared with its normal state. Thus we use standard score, which represents the number of deviations the popularity above the mean.

Localness. Let (p_1, p_2, ..., p_n) denote the event popularity in n different regions, where p_i denotes the popularity of the event in the i-th region. The localness is defined as

$$localness(e) = \frac{p_r - \mu}{\sigma} \tag{6}$$

where μ and σ are the mean and the standard deviation of (p_1, p_2, ..., p_n), respectively. As shown in Figure 5b, the localness is defined in a similar way with burstiness.

B. Learning Ranking Functions

Given all the ranking factors, we adapt a classification-based approach to learn a ranking function. Particularly, we choose the logistic regression model, which is a linear combination of all the ranking factors.

Definition 3 (Ranking Score)**.** *Given a region r in time frame t, the ranking score for an event $e = \{h_1, h_2, ..., h_k\}$ is the weighted averaged score of $score(h)$.*

$$\begin{aligned} score(e) &= \sum_{i=1}^{k} w_i score(h_i) \\ &= \alpha \sum_{i=1}^{k} w_i pop(h_k) + \beta \sum_{i=1}^{k} w_i burst(h_k) + \gamma \sum_{i=1}^{k} w_i local(h_k) \\ &= \alpha \cdot pop(e) + \beta \cdot burst(e) + \gamma \cdot local(e) \end{aligned} \tag{7}$$

where

- *w_i is the weight of hashtag h_i and is defined as $\frac{n_i}{n}$, where n_i represents the size of hashtag h_i and n represents the size of the event.*
- *α, β, and γ are linear weights that represent the preference for each factors (i.e., popularity, burstiness, and localness).*
- *$pop(\cdot)$, $burst(\cdot)$, and $local(\cdot)$ represent the popularity, burstiness, and localness for a hashtag or an event, respectively.*

Learning α, β, and γ. We treat $popularity$, $burstiness$, and $localness$ as features in the logistic regression model. Then we manually labeled top-10 important hashtags for two weeks in the USA, which are used as positive examples. Then we trained the logistic regression model to fit the manually labeled data to get the parameters.

Algorithm 5: `Top-k Event Ranking`

```
Input: $\alpha$, $\beta$, $\gamma$: user-specified weights
$k$: number of events to retrieve
$L_{pop}$, $L_{burst}$, $L_{local}$: sorted event lists according to popularity, burstiness, and localness
Output: Top-k events $E_k$
1  while $i$ from $1$ to $|E|$ do
2      $e_{pop} \leftarrow L_{pop}[i]$
3      $e_{burst} \leftarrow L_{burst}[i]$
4      $e_{local} \leftarrow L_{local}[i]$
5      $score(e_{pop}) \leftarrow$ COMPUTE-SCORE($e_{pop}$)
6      $score(e_{burst}) \leftarrow$ COMPUTE-SCORE($e_{burst}$)
7      $score(e_{local}) \leftarrow$ COMPUTE-SCORE($e_{local}$)
8      update top-k events $E_k$ with $score(e_{pop})$, $score(e_{burst})$, and $score(e_{local})$
9      initialize an event $e_{max}$ with $e_{pop}$.popularity, $e_{burst}$.burstiness, $e_{local}$.localness
10     $threshold$ = COMPUTE-SCORE($e_{max}$)
11     if $|E_k|.min > threshold$ then
12         return $E_k$
```

C. *Efficient Top-k Ranking*

Although we can learn the weights for each ranking factors according to human labeled data, different applications may have different preferences for each ranking factors. Some users may prefer popular events, while others may prefer local events or burst events. Therefore, we designed a more flexible way to organize the events which can quickly retrieve the top-k events given a set of ranking factor weights.

Inspired by the TA algorithm [33], we organize events in three sorted lists: (1) events sorted by popularity, (2) events sorted by burstiness, and (3) events sorted by localness. Given the weight for each ranking factor, we can compute the top-k events without scanning the entire event list. As shown in Algorithm 5, the TA algorithm contains the following steps: (1) Retrieve events at the i-th position from each sorted list ($lines$ 2 to 4). (2) Perform random access to get the popularity, localness, and burstiness of the current event, and compute the ranking scores. (3) Insert the current events to the top-k event set E_k. (4) Compute the maximum possible scores for the unseen events ($lines$ 9 to 10) and terminate if all the scores of the current top-k events are higher than the threshold.

VI. Experimental Study

A. *Dataset*

We crawled 9 million tweets with 2 million hashtags using Twitter's streaming API from December, 2013 to January, 2014 . We pre-processed the data with the following steps: (1) For geo-tagged tweets, geographical coordinates can be directly obtained. (2) For tweets without latitude and longitude coordinates, we roughly infer the coordinates according to users' location, since users may provide their countries/cities in the profile. Specifically, given all the <location, geographical coordinates> pairs extracted from geo-tagged tweets, we learn a density distribution for location. For a tweet without geographical coordinates, we randomly select a coordinate according to the density distribution of the location. Since we aim to find aggregated results (what is happening in a city), coordinates do not need to be exactly accurate. (3) All the tweets are stemmed using the Porter Stemmer.

B. *Evaluation Methodology*

Clustering and Ranking Quality. In Section VI-D, we first present the results of qualitative evaluation by analyzing some real cases, which gives us the concept of good clustering and ranking. Then we show our quantitative evaluation result in Section VI-E based on a manually labeled data.

Scalability. Since we are constructing data cubes in real-time, it is important to make sure our algorithm can handle tweets in a highly efficient way with a high throughput. Also it should scale well to large dataset with limited memory. The results will be discussed in Section VI-F and Section VI-G.

C. *Baselines*

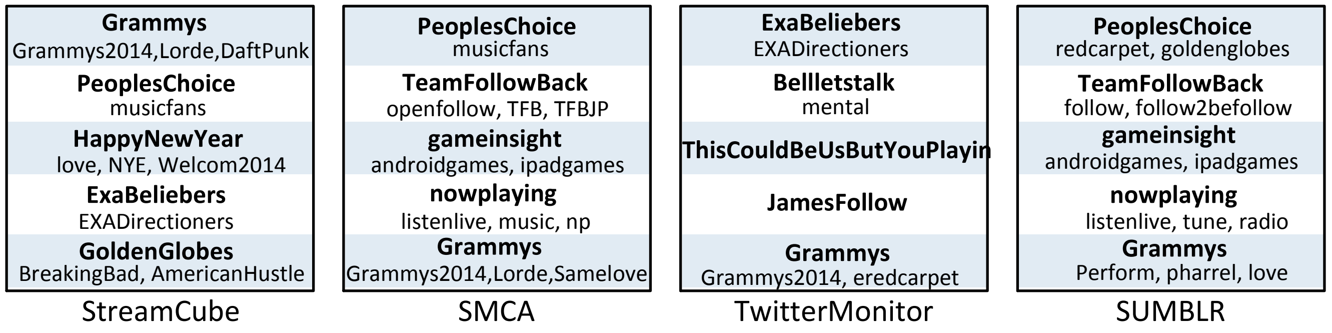
To the best of our knowledge, StreamCube is the first framework which aims to study hierarchical spatio-temporal hashtag clustering, which makes it possible to explore Twitter data with different space and time granularity in real-time. Thus we adapt some recent work on tweet clustering and burst event detection as our baselines.

SUMBLR. SUMBLR (SUMmarization By stream cLusteRing) [7] is an algorithm for efficient online tweet clustering. It has two stages: (1) Batch clustering of tweets as initial clusters, where each tweet is treated as a data point. (2) Incremental clustering, where each tweet is either assigned to its nearest neighbor or become a new cluster.

SMCA. SMCA (Scalable Multi-stage Clustering Algorithm) [30] is an algorithm for clustering tweet with the enhancement of hashtags. It has two stages: (1) Batch clustering of hashtags as the initial clusters. This is an offline stage, where each hashtag is an aggregation of tweets. K-means is used for batch clustering. (2) Online clustering of tweets, this is an online stage, which each tweet in the stream is assigned to the nearest clusters. If the tweet contains hashtags, then it is assigned directly to clusters based on its hashtags, where nearest neighbor search is avoided.

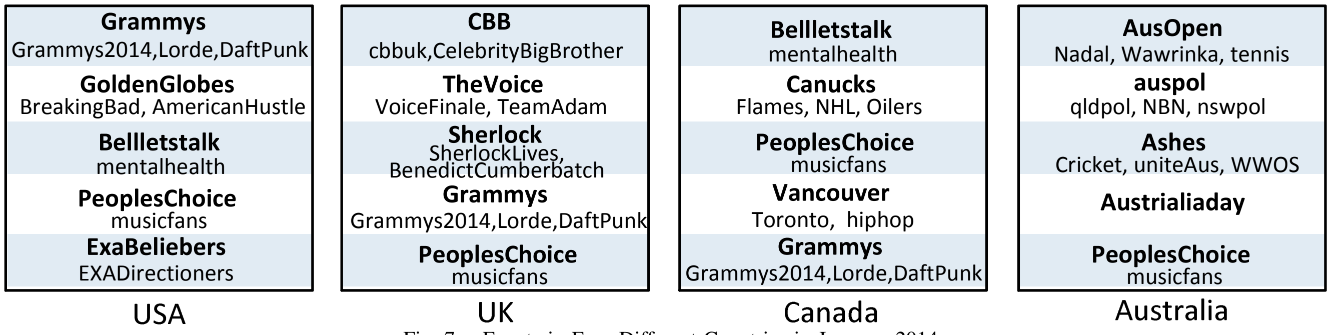
Both the above two baselines focus on clustering instead of event ranking. Thus we have another baseline that supports finding burst events.

TwitterMonitor. TwitterMonitor [10] is the state-of-the-art event identification algorithm for Twitter stream. It consists of two components: (1) Burst keyword identification, which is a real-time one-pass algorithm to detect burst keywords. (2) Event identification, which further uses the keyword co-occurrences information to cluster keywords into groups.



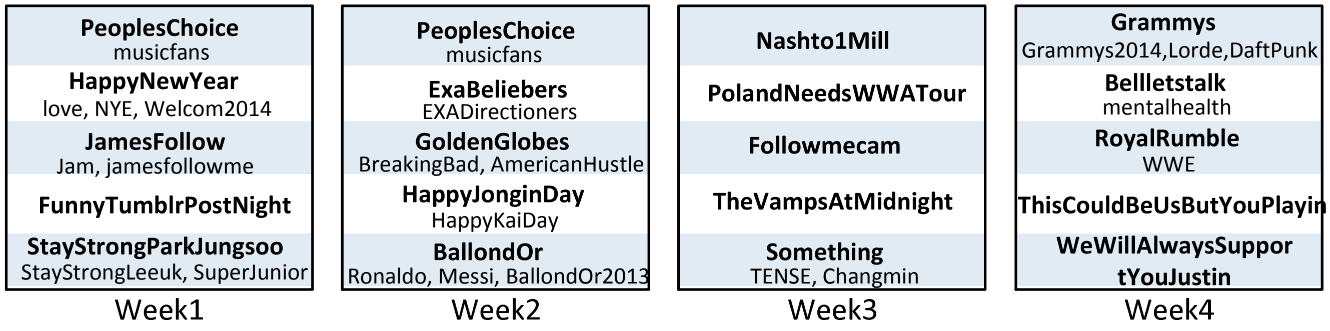
StreamCube	SMCA	TwitterMonitor	SUMBLR
Grammys Grammys2014,Lorde,DaftPunk	**PeoplesChoice** musicfans	**ExaBeliebers** EXADirectioners	**PeoplesChoice** redcarpet, goldenglobes
PeoplesChoice musicfans	**TeamFollowBack** openfollow, TFB, TFBJP	**Bellletstalk** mental	**TeamFollowBack** follow, follow2befollow
HappyNewYear love, NYE, Welcom2014	**gameinsight** androidgames, ipadgames	**ThisCouldBeUsButYouPlayin**	**gameinsight** androidgames, ipadgames
ExaBeliebers EXADirectioners	**nowplaying** listenlive, music, np	**JamesFollow**	**nowplaying** listenlive, tune, radio
GoldenGlobes BreakingBad, AmericanHustle	**Grammys** Grammys2014,Lorde,Samelove	**Grammys** Grammys2014, eredcarpet	**Grammys** Perform, pharrel, love

Fig. 6. Detected Events by Different Methods in January, 2014



USA	UK	Canada	Australia
Grammys Grammys2014,Lorde,DaftPunk	**CBB** cbbuk,CelebrityBigBrother	**Bellletstalk** mentalhealth	**AusOpen** Nadal, Wawrinka, tennis
GoldenGlobes BreakingBad, AmericanHustle	**TheVoice** VoiceFinale, TeamAdam	**Canucks** Flames, NHL, Oilers	**auspol** qldpol, NBN, nswpol
Bellletstalk mentalhealth	**Sherlock** SherlockLives, BenedictCumberbatch	**PeoplesChoice** musicfans	**Ashes** Cricket, uniteAus, WWOS
PeoplesChoice musicfans	**Grammys** Grammys2014,Lorde,DaftPunk	**Vancouver** Toronto, hiphop	**Austrialiaday**
ExaBeliebers EXADirectioners	**PeoplesChoice** musicfans	**Grammys** Grammys2014,Lorde,DaftPunk	**PeoplesChoice** musicfans

Fig. 7. Events in Four Different Countries in January, 2014



Week1	Week2	Week3	Week4
PeoplesChoice musicfans	**PeoplesChoice** musicfans	**Nashto1Mill**	**Grammys** Grammys2014,Lorde,DaftPunk
HappyNewYear love, NYE, Welcom2014	**ExaBeliebers** EXADirectioners	**PolandNeedsWWATour**	**Bellletstalk** mentalhealth
JamesFollow Jam, jamesfollowme	**GoldenGlobes** BreakingBad, AmericanHustle	**Followmecam**	**RoyalRumble** WWE
FunnyTumblrPostNight	**HappyJonginDay** HappyKaiDay	**TheVampsAtMidnight**	**ThisCouldBeUsButYouPlayin**
StayStrongParkJungsoo StayStrongLeeuk, SuperJunior	**BallondOr** Ronaldo, Messi, BallondOr2013	**Something** TENSE, Changmin	**WeWillAlwaysSupportYouJustin**

Fig. 8. Events in four weeks of January, 2014

D. Qualitative Evaluation

We first illustrate the top-5 global events in January detected by StreamCube and other baselines in Section VI-D1. Since the StreamCube support zoom in with different time and space granularity. We analyze the detected local events in four countries (USA, UK, Australia, and Canada) in Section VI-D2 and burst events in four weeks of January in Section VI-D3.

1) Global Events in January: The results of four algorithms are shown in Figure 6.

StreamCube. The top-5 events detected by StreamCube are: (1) 'Grammys', which are famous music awards for recognizing outstanding musicians. The ceremony is broadcast on January 26. (2) 'PeoplesChoices', which is an award for honoring the best in popular culture and is presented on January 8. (3) 'HappyNewYear', which is a hashtag used for celebrating the new year of 2014. (4) 'EXABeliebers', and 'EXADirectioners', which are two hashtags used for voting musician Justin Bieber, and music band One Direction on a show on a radio station called EXA during the second week of January. (5) 'GoldenGlobe', which is an award in film and television industry and is presented on January 12. The bold hashtag represents the one with the biggest size in the cluster. As we can see, all these events are quite human readable and unique to the time frame of January. Also we can see that hashtag clusters are self-explainable in summarizing events. For example, from the cluster {'Grammys', 'Grammys2014', 'Lorde', 'DaftPunk'} we can infer that the Lorde (a singer) and Daft Punk (musical group) have performed in the Grammys ceremony and have attracted much attention from the users in Twitter. Another example is {'GoldenGlobes', 'BreakingBad', 'AmericanHustle'}, where we can infer that Breaking Bad (a TV show) and American Hustle (a movie) have won the Golden Globes awards.

SMCA. As Figure 6 shows, the top-5 events detected by SMCA are: (1) 'PeoplesChoices'. (2) 'TeamFollowBack', which is a long last popular hashtags used for gain followers. (3) 'gameinsight', which is a hashtag for discussing ipad and android games. (4) 'nowplaying', which is used for denoting what the user is currently listening to, watching or playing. (5) 'Grammys'. The main problem of SMCA is that popular events may not carry valuable information. For example, 'nowplaying' is popular in every month while 'GoldenGlobes' detected by STREAMCUBE only happens in January. When we ranking the events in January, it is more reasonable to rank 'GoldenGlobes' higher than 'nowplaying'. The same problem exists for 'TeamFollowBack', which happens in every month and carries less information. The main reason for such ranking is that SMCA is designed to cluster hashtags instead of find burst events and local events. Thus the final ranking is mostly dominated by popular events.

TWITTERMONITOR. TWITTERMONITOR is designed to detect hot events base on burst keywords. The top-5 events detected are: (1) 'ExaBeliebers', which is a hashtag to support Justin Bieber (a singer) for a ratio session. (2) 'Belletstalk', which represents a fund for supporting mental health organizations. (3) 'ThisCouldBeUsButYouPlayin', which is used to highlight awkward photographs of couples, (4) 'JamesFollow', which is used to support James and is heavily used his fans, and (5) 'Grammys'. TWITTERMONITOR mainly has two drawbacks: (1) Burst keywords have lower quality and carries less information compared with burst hashtags. For example, TWITTERMONITOR clusters 'Belllettalk' with 'mental'. In contrast, STREAMCUBE clusters 'Belllettalk' with hashtag 'mentalhealth', which make it easier for users to understand that 'Bellelettalk' is an event related to 'mentalhealth'. (2) Considering burstiness is not enough for a good event ranking. For example, TwitterMoniter ranked 'JamesFollow' among the top-5 events in January, which is a hashtag for encouraging people to follow a user called James. However, there may be other hashtags that only have high burstiness but also keep popular during the whole January, such as 'peoplechoice' and 'Grammys'. It is more reasonable to rank these hashtags higher than 'JamesFollow'.

SUMBLR. All the top-5 hashtags have been introduced before. The first problem of SUMBLR is that the events does not have clear boundaries and it is easy to mix different events. Since tweets are extremely short, the clustering is easy to be affected by noisy tweets. For example, 'PeoplesChoice' and 'GlodenGlobes' are clustered into one cluster, but they are two independent events. The second problems is words are not easy to understand. For example, 'Grammys' are clustered with 'perform' and 'love'. However, the 'love' here refers to a song performed in Grammys called 'the same love'. Finally, SUMBLR has problem in finding valuable events that are particular in January, which is similar to SMCA.

2) Local Events in January: Different from existing algorithms, STREAMCUBE enables us to zoom in with finer space granularity. In this study, we show the detected events in four countries: United States, United Kingdom, Canada, and Australia. We can find many local events in each country as shown in Figure 7. Since most of the events in US has been introduced. We will focus on the other countries.

United Kingdom. We can find three localized events: (1) 'CBB', which is short for Celebrity Big Brother, a popular TV show in UK. (2) 'TheVoice', which is a talent show. (3) 'Sherlock', which is a crime drama and is broadcast in January. Although 'Grammys' and 'PeoplesChoice' are not localized events, they have a high burst during January and are very popular.

Canada. We can find three localized events: (1) 'BellLetsTalk' is a charitable program dedicated to mental health in Canada. (2) 'Canucks' is an ice hockey team in Canada. (3) 'Vancouver', is a city of Canada. Also we find that 'PeoplesChoice' and 'Grammys' quite popular.

Australia. We can find four local events: (1) 'AusOpen' (Australian Open), which is a major tennis tournament held in Melbourne, Australia. (2) 'auspol' (Australian politics) is a hashtag used for discussing political issues. (3) 'Ashes', which is the notional prize in a Test cricket series played between England and Australia. (4) 'Australiaday', which is the official national day of Australia.

From the results above, we can see that STREAMCUBE can successfully detect local events given a particular region.

3) Events in Each Week of January: Besides zoom in with finer space granularity, we can also zoom in with finer time granularity. As shown in Figure 8, we listed top-5 events for each of the four weeks in January in global space. As we can see, different weeks have different popular events. For example, 'HappyNewYear' only occurs in the first week. In the second week, we find 'BallondOr' quite popular, which stands for an award given annually to the best football player. Users can use 'BallondOr' to support their favorite players. We also find that 'RoyalRumble', which is a professional wrestling event, attracts a lot of users in the fourth week. From the results we can see that STREAMCUBE can provide meaningful rankings for different time granularity.

To summarize, we can see from the examples that STREAMCUBE can not only find high quality hashtag clusters but also detect local events and burst events.

E. Quantitative Evaluation

In the previous section, we have listed some examples on hashtag clustering and ranking. In this section, we introduce the quantitative evaluation results. Since it would be impossible for human to rank all hashtag clusters manually. The ground truth is constructed according to the following steps: (1) Given a particular time frame and region, we merge the top-50 ranked hashtag clusters from SUMBLR, TWITTERMONITOR, SMCA and STREAMCUBE as the initial candidate set. (2) We asked 10 volunteers to rank the top-10 events from the initial candidate set. (3) Since different users may have different ranking, the final top-10 ranking are decided by majority voting, assuming all the volunteers are equally weighted. Finally, the ground truth consists of four parts: (1) top-10 events for global space in January, which consists of 10 ranked clusters, (2) top-10 events for four countries (USA,

TABLE III
CLUSTERING QUALITY

Metric	SUMBLR	TMONITOR	SMCA	STREAMCUBE
NMI	0.312	0.336	0.361	**0.381**
RI	0.609	0.642	0.684	**0.717**

TABLE IV
RANKING QUALITY

Metric	SUMBLR	SMCA	TMONITOR	STREAMCUBE
MAP	0.511	0.523	0.608	**0.634**

UK, Canada, and Austrilia), which are 40 ranked clusters, (3) top-10 events for global space in each week in January, which are 40 ranked clusters, and (4) top-10 events for four countries in each week of January, which are 160 ranked clusters. In total, we have 250 human annotated events as ground truth.

1) Clustering Quality: To evaluate how well our clustering results match the gold standard classes, we use two widely used metrics in clustering evaluation[1]: NMI (Normalized Mutual Information) and RI (Rand Index).

As shown in Table III. STREAMCUBE outperforms other algorithms in terms of NMI and RI. SUMBLR, which incrementally absorbs tweets to existing clusters, has the worst performance. This implies a single tweet is too short for computers to measure its semantic relatedness with existing clusters. TWITTERMONITOR has a better performance than SUMBLR because it uses the co-occurrences of burst keywords for clustering, instead of treating tweets as plain documents. SMCA achieves the best clustering qualities among the baselines. The main reason is that it treat hashtags as basic data points in the initialization stage. However, it does not consider the fact that hashtag content envolves over time, thus does not provide any clustering adjustments as new tweets emerge.

2) Ranking Quality: We use Mean Average Precision (MAP) to measure the ranking quality, which is a popular metric in ranking problems. Given a ranked event list $E = (e_1, e_2, ..., e_n)$, the Average Precision (AP) is defined as

$$AP = \frac{\sum_{k=1}^{n} precision@k \times isTopEvent(e)}{the\ number\ of\ positive\ events} \quad (8)$$

where $isTopEvent(e)$ is an indicator function and equals 1 only when event e is labeled among the top-n events, otherwise it equals 0. The number of positive events are 10 since we selected top-10 events. With the knowledge of AP, MAP is defined as the mean of all APs:

$$MAP = \frac{\sum_{i=1}^{N} AP_i}{N} \quad (9)$$

As shown in Table IV, STREAMCUBE achieves the best performance in terms of MAP. SUMBLR and SMCA does not perform well on this task because they simply use popularity-based ranking. TWITTERMONITOR achieves better performance because it explicitly models burstiness into the clustering algorithm. However, it does not take localized events into consideration. Thus some localized events cannot get high scores. Finally, by combining popularity, burstiness, and localness into ranking function, STREAMCUBE successfully outperforms others.

[1]http://www-nlp.stanford.edu/IR-book

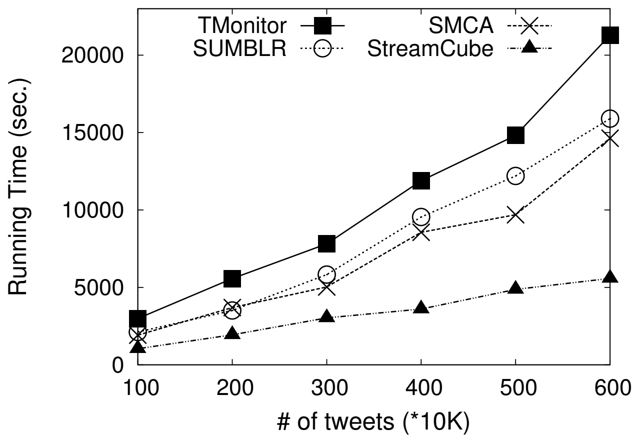


Fig. 9. Scalability

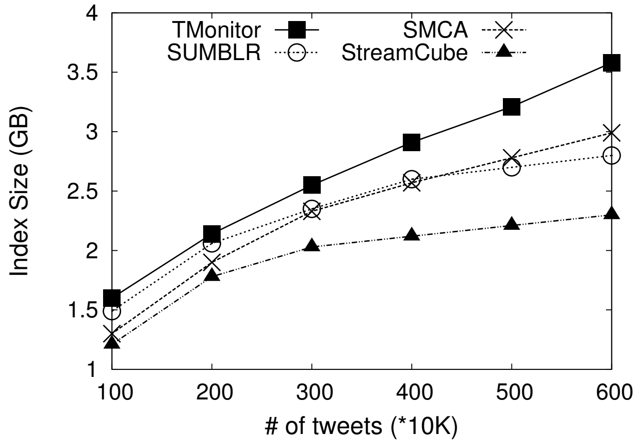


Fig. 10. Memory Usage

F. Scalability

Scalability is tested by measuring the running time for different data size. As shown in Figure 9, TWITTERMONITOR is the slowest algorithm. The main reason is that the set of candidate of words is much larger than the size of hashtags. Thus it takes more time to compute the similarities between keywords. Both SMCA and SUMBLR achieve better performance than TWITTERMONITOR since they are designed for incremental clustering, and it takes less time to assign a new tweet to an existing cluster. STREAMCUBE has the shortest running time and scales well with different data size. The main reason is that tweets are processed in a divide-and-conquer fashion, in which case the clustering results are merged according to the time and space hierarchy. Another reason is that STREAMCUBE can efficiently find the nearest neighbor given a new hashtag to be clustered.

G. Memory Usage

Memory usage is measured by the index size with different data size. As shown in Figure 10, the memory usage of TWITTERMONITOR is larger than other algorithms. The main reason is that TWITTERMONITOR needs to maintain a similarity matrix for all pairs of keywords. SUMBLR takes more memory at the early stage but remains stable for large datasets. This is mainly because SUMBLR can detect and remove out-dated clusters periodicity. SMCA takes less memory than SUMBLR because the tweets are first aggregated by hashtags to save the space. However, the memory usage of SMCA continues to grow as new hashtags keep emerging. Finally, STREAMCUBE has the best performance among all the algorithms. Similar to SMCA, STREAMCUBE takes less space due to hashtag-based aggregation in the early stage. Similar to SUMBLR, STREAMCUBE flushs out-dated clusters to disk. So the memory usage remains stable for large datasets.

VII. Conclusion and Future Work

In this paper, we proposed STREAMCUBE to support hierarchical spatio-temporal hashtag clustering, in which case users can explore twitter data interactively with different time and space granularity. To the best of our knowledge, this is the first framework to support such application. Our approach contains three components: (1) A spatio-temporal hierarchy inspired by the quad-tree and the data cube. Hashtag clustering is performed according to a divide-and-conquer strategy at the lowest level of the hierarchy. Then clustering results are merged incrementally in a bottom-up manner. (2) A single-pass hashtag clustering algorithm. Different from existing clustering techniques, we are dealing with content-evolving hashtags. (3) Event ranking, which is designed to help users identify local events and burst events.

STREAMCUBE can be extended in many directions. Firstly, we can extend STREAMCUBE to support topic-based exploration. For example, users explore events in a specific domain such as politics, music, travel, breaking news, etc. This can be considered as a four-dimension data cube (i.e., time, latitude, longitude, and topic). Secondly, it is also important to develop an alert mechanism to push information to users from the server instead of waiting for user-initiated queries since users cannot keep monitoring what is happening.

Acknowledgment

This work was supported in part by National Basic Research Program of China (973 Program) under Grant No. 2014CB340505, National Natural Science Foundation of China under Grant No. 61272088, Tsinghua University Initiative Scientific Research Program, the US Army Research Lab. under Cooperative Agreement No. W911NF-09-2-0053 (NSCTA), the US Army Research Office under Cooperative Agreement No. W911NF-13-1-0193, US National Science Foundation IIS-1017362, IIS-1320617, and IIS-1354329, HDTRA1-10-1-0120, and MIAS, a DHS-IDS Center for Multimodal Information Access and Synthesis at UIUC.

References

[1] T. Sakaki, M. Okazaki, and Y. Matsuo, "Earthquake shakes twitter users: real-time event detection by social sensors," in *WWW*, 2010.
[2] J. Bollen, H. Mao, and X. Zeng, "Twitter mood predicts the stock market," *Journal of Computational Science*, 2011.
[3] J. Han, *Data Mining: Concepts and Techniques*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005.
[4] C. C. Aggarwal, "Mining text and social streams: A review," *SIGKDD Explor. Newsl.*, vol. 15, no. 2, pp. 9–19, Jun. 2014.
[5] Y. Song, H. Wang, Z. Wang, H. Li, and W. Chen, "Short text conceptualization using a probabilistic knowledgebase," in *Proc. of IJCAI*, 2011.
[6] O. Ozdikis, P. Senkul, and H. Oguztuzun, "Semantic expansion of tweet contents for enhanced event detection in twitter," *ASONAM 2012*, vol. 0.
[7] L. Shou, Z. Wang, K. Chen, and G. Chen, "Sumblr: continuous summarization of evolving tweet streams," in *SIGIR*, 2013, pp. 533–542.
[8] H. Becker, M. Naaman, and L. Gravano, "Beyond trending topics: Real-world event identification on twitter," in *ICWSM*, 2011.
[9] C. C. Aggarwal and K. Subbian, "Event detection in social streams." in *SDM*. SIAM / Omnipress, 2012, pp. 624–635.
[10] M. Mathioudakis and N. Koudas, "Twittermonitor: trend detection over the twitter stream," in *SIGMOD Conference*, 2010, pp. 1155–1158.
[11] C. I. Muntean, G. A. Morar, and D. Moldovan, "Exploring the meaning behind twitter hashtags through clustering," in *BIS (Workshops)*, 2012.
[12] S. Carter, M. Tsagkias, and W. Weerkamp, "Twitter hashtags: Joint translation and clustering," in *Web Science 2011*. ACM, 2011.
[13] C. Budak, T. Georgiou, D. Agrawal, and A. El Abbadi, "Geoscope: Online detection of geo-correlated information trends in social networks," *PVLDB*, vol. 7, no. 4, pp. 229–240, 2013.
[14] W. Xie, F. Zhu, J. Jiang, E.-P. Lim, and K. Wang, "Topicsketch: Real-time bursty topic detection from twitter," in *ICDM*, 2013, pp. 837–846.
[15] T. Lappas, M. R. Vieira, D. Gunopulos, and V. J. Tsotras, "On the spatiotemporal burstiness of terms," *PVLDB*, vol. 5, no. 9, 2012.
[16] T. Lappas, M. R. Vieira, D. Gunopulos, and V. Tsotras, "Stem: a spatio-temporal miner for bursty activity," *SIGMOD Conference*, 2013.
[17] M. Sarwat, J. Bao, A. Eldawy, J. J. Levandoski, A. Magdy, and M. F. Mokbel, "Sindbad: a location-based social networking system," in *SIGMOD Conference*, 2012, pp. 649–652.
[18] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: user movement in location-based social networks," in *KDD*, 2011.
[19] G. Li, Y. Wang, T. Wang, and J. Feng, "Location-aware publish/subscribe," in *KDD*, 2013, pp. 802–810.
[20] A. Marcus, M. S. Bernstein, O. Badar, D. R. Karger, S. Madden, and R. C. Miller, "Twitinfo: aggregating and visualizing microblogs for event exploration," in *CHI*, 2011, pp. 227–236.
[21] D. Shahaf, J. Yang, C. Suen, J. Jacobs, H. Wang, and J. Leskovec, "Information cartography: creating zoomable, large-scale maps of information," in *KDD*, 2013, pp. 1097–1105.
[22] A. Marcus, M. S. Bernstein, O. Badar, D. R. Karger, S. Madden, and R. C. Miller, "Processing and visualizing the data in tweets," *SIGMOD Record*, vol. 40, no. 4, pp. 21–27, 2011.
[23] M. Busch, K. Gade, B. Larson, P. Lok, S. Luckenbill, and J. Lin, "Earlybird: Real-time search at twitter," in *ICDE*, 2012, pp. 1360–1369.
[24] J. Yao, B. Cui, Z. Xue, and Q. Liu, "Provenance-based indexing support in micro-blog platforms," in *ICDE*, 2012, pp. 558–569.
[25] L. Wu, W. Lin, X. Xiao, and Y. Xu, "Lsii: An indexing structure for exact real-time search on microblogs," in *ICDE*, 2013, pp. 482–493.
[26] A. Magdy, M. F. Mokbel, S. Elnikety, S. Nath, and Y. He, "Mercury: A memory-constrained spatio-temporal real-time search on microblogs," in *ICDE*, 2014, pp. 172–183.
[27] L. Hong, A. Ahmed, S. Gurumurthy, A. J. Smola, and K. Tsioutsiouliklis, "Discovering geographical topics in the twitter stream," in *WWW*, 2012, pp. 769–778.
[28] Q. Yuan, G. Cong, Z. Ma, A. Sun, and N. Magnenat-Thalmann, "Who, where, when and what: discover spatio-temporal topics for twitter users," in *KDD*, 2013, pp. 605–613.
[29] H. Yin, B. Cui, H. Lu, Y. Huang, and J. Yao, "A unified model for stable and temporal topic detection from social media data," in *ICDE*, 2013.
[30] O. Tsur, A. Littman, and A. Rappoport, "Efficient clustering of short messages into general domains." in *ICWSM*, E. Kiciman, N. B. Ellison, B. Hogan, P. Resnick, and I. Soboroff, Eds., 2013.
[31] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," in *Proc. of VLDB*, 2003.
[32] C. Buckley and A. F. Lewit, "Optimization of inverted vector searches," in *Proc.*, ser. SIGIR '85, pp. 97–110.
[33] R. Fagin, A. Lotem, and M. Naor, "Optimal aggregation algorithms for middleware," in *In Proc.*, ser. PODS '01, 2001, pp. 102–113.