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

Vaibhav Kasturia
L3S Research Center
Leibniz Universität Hannover
Hannover, Germany
+49-1628626978
vbh18kas@gmail.com

Eirini Ntoutsi
L3S Research Center
Leibniz Universität Hannover
Hannover, Germany
+49-511762-17756
ntoutsi@l3s.de

ABSTRACT
Data from Microblogging websites, such as Twitter, has gained considerable interest over the past few years due to its content and growing popularity. Twitter users post many updates about the on-going events. Identifying, analyzing and grouping such updates may help us to detect events in real time before they get published as news articles, or analyze long-term events, for example, the Refugee Crisis in Europe or the United States Presidential Election. Several platforms have been developed for event detection and analysis. Existing systems usually allow users to explore events along different granularities of space, time or topic. Data storage and management is another important feature of such systems as these factors extensively affect the performance of these systems. This paper provides a critical review of one such system called STREAMCUBE and compares it with other systems such as EvenTweet, TwitterMonitor, SUMBLR and SMCA. It highlights the strengths and the shortcomings of these systems and suggest possible extensions.

CCS Concepts
• **Information Storage Systems → Information Storage Management → Hierarchical Storage Management**

• **Information Systems Applications → Data Mining → Clustering, Stream Data Mining**

Keywords
Event Detection, Stream Mining, Clustering, Burstiness

1. INTRODUCTION
Twitter has become the most famous microblogging websites, with users posting over a billion tweets every week. Twitter allows users to share happening events in their life with their followers in the form of tweets. Each tweet has a word limit of 140 characters. This restriction on the word limit has led users to describe each event in short and has led to many users adding links in their tweets to other external sites to allow their followers to follow the event in detail. Users also add hashtags, which are phrases or words preceded with a '#', and this is used to identify messages related to a particular event to topic. Followers are further allowed to retweet or re-post a particular tweet, to spread the topic further and to create more interest for that topic.

The data on Twitter has numerous applications these days. Apart from event detection, it is used for sentiment analysis in which the positive and negative opinions in social media discussion are used as data for analysis tasks, such as reviewing a product or election candidate surveying. Tools such as Political Hashtag Trends(PHT)[1], have been developed for finding political polarization of Twitter users. In the US Presidential Elections of 2016, parties usually monitor the polarization of Twitter users in the United States, seeing which users retweet the tweets posted by their political candidates to find out whether a user is pro-Democrat or pro-Republican or whether he favors both. Then using the profile information and the previous tweets by the users, parties can actually see the other interests of their followers and shape their policies and make poll predictions [1]. There are systems which use Twitter stream data to find out disasters and monitor crime in real-time. One such system proposed allows to detect earthquakes in real time using tweets related to earthquakes in Japan, a country where there are a lot of Twitter users and earthquakes are common [2]. Tweets can also be used to trace the evolution of events, for example, twitter data was used to trace the Egyptian Revolution of February 2011 and the Syrian Uprising in March 2012 in the paper by SalahEldeen and Nelson[3].

As described before, a tweet is typically composed of text, hashtags, user tags and URIs or embedded resources all spanning a maximum of 140 characters. Figure 1 shows a typical example of a tweet record from the Stanford Snap Project Dataset[2]. In this figure, the line beginning with T refers to the timestamp, the line beginning with U shows a link to the user who authored this particular tweet and the line beginning with W the content of the tweet. Because more and more users access Twitter from mobile devices with GPS, more and more tweets are getting geo-tagged as well. The authors of the assigned paper [4] try to use this geo-information for mapping events to a location and assume that using hashtags enables us to find the content of a tweet more accurately.

The assigned paper considers each event as a cluster of hashtags. For example, the Syrian Uprising of 2012 can be represented using the hashtag cluster {#AssadCrime, #Bashar, #Assad, #RiseDamascus, #GenocideInSyria} where #Bashar and #Assad represents the Syrian president Bashar al-Assad, #AssadCrime and #GenocideInSyria refer to the crimes committed by him and #RiseDamascus refers to rise of people protesting against Bashar al-Assad in Damascus, the capital city of Syria. STREAMCUBE performs spatio-temporal hashtag clustering by constructing a data cube from the twitter stream in real time and this data cube structure is an extension of the data cube structure from database literature [5]. Figure 2 shows how the hashtag clusters are organized into data cubes according to their timestamps and geographical information. From this structure, STREAMCUBE allows users to do explore events along different time and space granularity. If the

[1]http://politicalhashtagtrends.sandbox.yahoo.com/

[2]http://snap.stanford.edu/

```
T   2009-07-31 23:57:18
U   http://Twitter.com/nickgotch
W   RT @rockingjude: December 21, 2009 Depopulation by Food
      WHOA..BETTER WATCH RT plz #pwa #tcot
```

Figure 1. Sample Tweet Record

user wants to finds the top 10 events of a month, then the time granularity should be chosen as a month or if the user wants to find breaking news, then the time granularity chosen should be finer, like a day. Similarly, for finding local news and global news, the user may choose a district or the global space as the basic space granularity. The system also allows to detect burst events and localized events and a combination of both called burst localized events. Burst events are events which exhibit high popularity for a very short period of time. Local events are events which are restricted only to a particular region.

There are several challenges faced when doing spatio-temporal hashtag clustering and the authors for the challenges devised the following solutions:

i. Iterative computation should be avoided by the clustering algorithm if it is to detect events in real time. The authors of the assigned paper developed a single-pass hashtag clustering algorithm which merges newly generated results with the earlier results in an incremental fashion. For example, having identified events for the current day and in order to get the events for the current month, we need to make sure that the events for the current day are merged with little cost to the events of the rest of the days of the current month.

ii. While clustering, data points cannot be assumed to be static. This is because contents of the hashtags keep evolving and hashtags which are similar at one point in time may become dissimilar at a later point in time. For example, #merkel might be closely related to #hannovermesse during the time Angela Merkel visits the Hannover Messe but it might be related to other events such as #eureferendum at a time when Angela Merkel is attending a meeting related to the EU Referendum. The clustering algorithm devised by the authors handles content evolving hashtags in an online fashion.

iii. The algorithm needed to have a clear understanding of the localness and burstiness in reference to the data cube. This is because for a user from a region, events from that region during the time frame may be more valuable than the events from another region during the same time frame. Keeping the time frame very short in this example leads to the problem of detecting burst localized events. The algorithm devised by the authors is able to adapt between situations where the users want to find burst localized events as compared to situations where the user wants to find global events during a year by varying the scores of events by popularity, burstiness and localness.

2. RELATED WORK

2.1 Data Cube

The technique used by the algorithm for Data Warehousing is an extension of the traditional data cube structure. Data Cube is structured in such a way that it allows to look up facts along different dimensions. In the context of the assigned paper, the system designed allows users to explore events along the time and

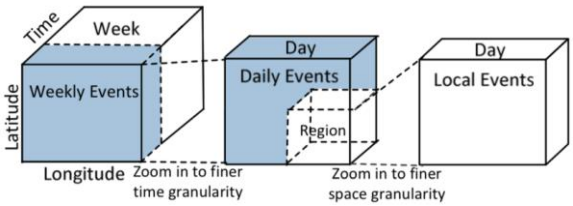


Figure 2. Zoomable Event Cube

space dimension. The cube structure used in the system differs from the traditional structure as the data cube structure arranges data arriving in real time along the time and space dimension and is able to rank events given a time frame and a region using burstiness and localness measures which is not possible to achieve with the traditional data cube structure.

2.2 Clustering Techniques in Social Media

The clustering techniques for social media and traditional text media differ because traditional text media like news articles are structured and of good quality whereas the quality of tweets is most of the times not as good and many of the tweets contain spam as well. As described earlier, the length of each tweet is very short as well. Therefore, a single tweet can never be used for the complete description of an event but a group of tweets must be used in order to accurately describe an event. Many methods have been developed which classify tweets on the basis of keywords. TwitterMonitor [6] is a system which find out events by clustering keywords with high burstiness. However, this method fails to take other measures like localness and popularity into account and thereby might not be able to provide best localized event results to users interested in regional news. EvenTweet[7] is another system which clusters keywords with high localness and thereby provides very good localized results but has the drawback that it cannot detect events with high burstiness. SMCA[8] is a system which also does clustering on the basis of keywords but in a fashion that it initially performs a batch clustering of keywords. In batch clustering, clustering is performed on the entire set in offline mode. After batch clustering of keywords, each new tweet that comes in is clustered in an incremental fashion based on keyword to its nearest cluster or it may form its own new cluster. SUMBLR[9] is another system which works in a similar way to SMCA, the only difference being that it performs batch clustering of tweets instead of batch clustering of keywords. And similar to SMCA, in the second step whenever a new tweet comes in it is clustered in an incremental way to its nearest cluster or may form its own new cluster. Clustering of tweets has the disadvantage that the tweets are static data points in the vector space and are quite noisy most of the times. The STREAMCUBE system [4] differs from all these systems as it uses hashtags to perform clustering instead of using keywords or complete tweets. Table 1 shows a comparison of STREAMCUBE with existing work. Hashtags are nothing but keywords provided by user only related to the event. The keywords extracted from tweets using some systems will not in most cases be able to express an event better than the hashtags of the tweet itself. Each event in STREAMCUBE is considered as a combination of a bag of words and bag of hashtags as we will see in later sections. STREAMCUBE further differs from other systems which perform hashtag based clustering as those systems fail to consider hashtags as dynamic points continuously moving in vector space and instead treat them as static data points while performing clustering and hence the results provided by STREAMCUBE can be expected to

Table 1. Comparison with existing work

Method	SUMBLR [6]	TMONITOR [8]	SMCA [28]	Ours
Tweet Level	√	√		
Hashtag Level			√	√
Zoomable	√			√
Interpretable		Partial	√	√
Ranking		Partial		√

be much better than any of the other hashtag based clustering systems.

2.3 Stream Granularity in Clustering

All the systems described from [6] to [9] and [4] make use of an aggregated stream of twitter data. Different from all these systems, Lappas et. al. developed a system for burst clustering [10] which allows users to choose the granularity of the twitter stream. The granularity of the twitter stream could range from a particular neighborhood to an entire city. Suppose, the granularity of the stream is a city. Then in the different streams coming into the system, each stream would correspond to tweets coming from a particular city. A temporal overlap between these streams could indicate that both the steams might be about the same topic. A maximal overall between the streams is found by treating each overlap as edges in a graph and then reducing the problem to find out the maximum clique number of the graph. In this way, the system developed also allows us to find combinatorial patterns in addition to bursty patterns and regionalized patterns as combinatorial patterns are patterns for which high frequency gets observed for the same term t in the same temporal interval I, but these patterns do not take any geographic proximity into account.

2.4 Event Ranking

In STREAMCUBE the data structure is a zoomable event cube which allows users to zoom in and out on maps to find out what is happening in a particular area. Users can also choose a particular time frame that they might be interested in. STREAMCUBE detects and ranks burst and local events not from the complete dataset but from the subset of the dataset chosen by the user by specifying time and space granularity. Similar to STREAMCUBE, EvenTweet [7] detects local events from the subset of the dataset chosen by the user by specifying time and space granularity. However, EvenTweet[7] provides the advantage that a user can specify the region he is interested in by specifying the coordinates of the bounding rectangle. In this way, in the experiments the authors found out localized events for just a football stadium during the 2012 UEFA European Football Championship by restricting the coordinates of the bounding rectangle to just the football stadium. STREAMCUBE, on the other hand, allows the user to choose only a minimum granularity of district. The method [10] described earlier has the drawback that it does not rank events from a subset of the dataset but from the whole dataset and thus is able to fetch burstiness events from the whole dataset only.

3. DATA WAREHOUSING STRUCTURE

STREAMCUBE has three main parts: (1) The event cube structure that it uses for spatio-temporal aggregation, (2) clustering that it performs on hashtags, (3) event ranking. This section provides a detailed description of the first part of the system, namely, the event cube structure that it uses for data warehousing. Figure 3 shows the organization of events according to space and

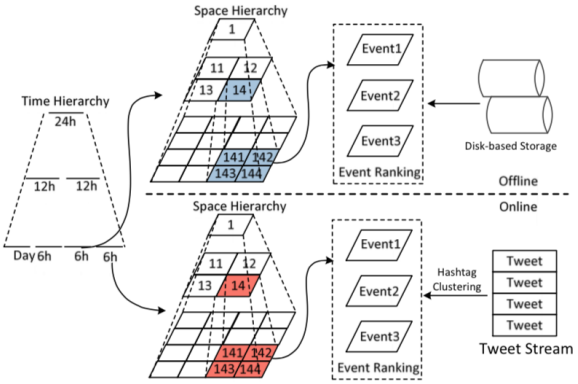


Figure 3. Framework

time hierarchy. Once the events are arranged according to space and time hierarchy, users can be immediately returned aggregated results based on the space and time granularity that they choose. An important thing to note in this figure is that it is the space that is embedded into the time and not the time that is embedded into the space.

3.1 Space Hierarchy

The space taken is the entire global space and the space is organized in a quad-tree like hierarchy. The top level of space represents the entire global space and is split into regions of equal size. The regions are then further split into sub-regions of equal size. The granularity of space allowed are country, city and district. Another thing is that all countries have varying sizes and varying number of cities. Each city in turn also has a varying size and varying number of districts. Taking these factors into account, the question that arises here is that how the authors claim to take all regions divided into the same number of sub-regions and each having the same size. Another possible improvement in this space hierarchy would be to consider not the entire global space, but initially begin with a country like the United States, where Twitter is very popular and expand the space and adding countries as new tweets come in from the other countries. That way countries where Twitter is not very popular could be not stored in each snapshot.

3.2 Time Hierarchy

Time has also been organized in a hierarchical fashion like the space hierarchy with the coarsest granularity of time being one day. The day in the second level is then split into two 12 hour frames and in the third level each 12 hour frame is further split into two 6 hour frames.

3.3 Connecting Space and Time Hierarchies

Each six hour time frame keeps a snapshot of the globe. The tweets that keep coming in are mapped to the snapshot of the globe in the current time frame. To avoid wastage of memory, only the current six hour time frame is kept in the memory and rest all previous time frames are stored in the disk.

Algorithm 1 gives the pseudo-code for the spatio-temporal aggregation algorithm. This algorithm has three main parts:

i.) **Creating new cubes:** For each tweet that comes in from the twitter stream in the current time frame, it first gets assigned to the lowest level of hierarchy, i.e., to its geo-location in the snapshot of the globe inside the current time frame. Different tweets coming in would have different regions. Thus, in order to speed up things, the

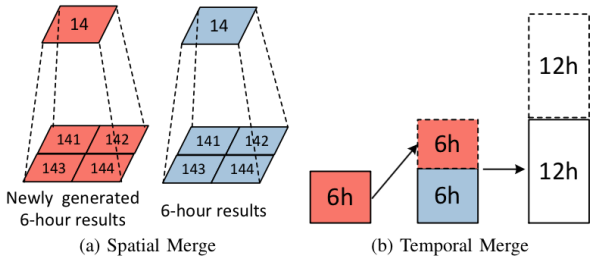


Figure 4. Spatio-temporal aggregation

mapping of the tweets to the snapshot of the globe can be done in a parallel way using multiple threads to perform this job. For the cube (line 2), the hashtag-clustering method (line 3) changes either the content of the cube or creates a new event. The update-event-ranking method updates the event ranking scores.

ii.) **Spatial merge:** The lines 5-7 of the algorithm describe the spatial merge process. The spatial merge process can be better visualized by the Figure 4a. In the Figure 4a. in both the cubes there is the same region #14 which has four sub-regions #141, #142, #143 and #144. Assuming these sub-regions as the lowest level of hierarchy here, the events would initially get mapped to these sub-regions. The spatial-merge method combines all the events in the sub-regions #141, #142, #143 and #144 into the region #14 for both the current six hour time frame and as well as the previous six hour time frame. The issue that still remains unclear here is that if we have event A in sub-region #143 and events B and C in sub-region #144, whether when calling spatial merge the event A is merged with B, merged with C or all are kept as separate events. This is because there could also be an overlap in the events from sub-regions or there could also be the possibility that the events A and B refer to the same event in different sub-regions.

iii.) **Temporal merge:** After having performed spatial merge on the small six hour time frame, it needs to be merged with the larger time frames. The lines 8-10 of the algorithm describe this process and it can be better visualized looking at the Figure 4b. The temporal aggregation will merge the red region #14 of the current six hour time frame with the blue region #14 of the previous six hour time frame, thus creating a current 12 hour time frame. This current 12 hour time frame can be again then merged with the previous 12 hour time frame to get a complete 24 hour cube.

4. HASHTAG CLUSTERING

Before we map events to the lowest level of hierarchy in the event cube and perform spatio-temporal aggregation, we need to detect the events. As we already know, STREAMCUBE performs clustering by treating hashtags as the basic data points. The concept of hashtags and events has been described earlier in the Introduction section. We now move on to formalize hashtag and event representation.

4.1 Hashtag Representation and Similarity

A hashtag can be represented in different forms in different contexts. In the context of the assigned paper, a hashtag has been described both as a bag of words and a bag of hashtags.

4.1.1 Hashtag as a bag of words

In this representation, we first find out all the tweets that contain the hashtag h and then find out all the words in the tweets which

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 ∈ D do
2      cube ← 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] ← 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] ←
           temporal-merge(cubes[t − 1][region],
           cubes[t][region])
```

are important for the hashtag. For example, if we begin with a hashtag such as '#GoldenGlobes', and collect all the tweets that contain this hashtag, then the most important words in the tweets describing the hashtag might be the actors and directors of the TV Series and Movies, the TV Series and Movies which got nominated and won this award and maybe the location of this event. Formally, if W denotes the total number of words, then the hashtag **h** can be represented as a normalized weighted vector as

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

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

4.1.2 Hashtag as a bag of hashtags

In this representation, we take into account the fact that the hashtags which describe an event co-occur with each other. For example, coming back to our example on the Syrian Uprising of 2012, tweets describing this event commonly contained the hashtags {#AssadCrime, #Bashar, #Assad, #RiseDamascus, #GenocideInSyria}. This shows that these hashtags are somehow related to each other in the context of this event. If H denotes the hashtag set and **h** is the hashtag then

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

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

4.1.3 Hashtag Representation and Similarity

A hashtag can be represented as a combination of both bag of words and bag of hashtags and then given two hashtags h1 and h2, their similarity can be calculated as follows

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

Here, $\alpha + \beta = 1$ and β is set to 0.7 for the system which means that the system gives more weight to hashtags than important words extracted from the tweets. Overall, what the system does is that it

Algorithm 2: 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 ← 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' ← e \ 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)
```

finds out the normalized word weighted vector and normalized hashtag weighted vector and again does re-normalization when using combination of both for finding out the similarity between two hashtags. The hashtag finally can be represented as the vector:

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

4.1.4 Event Representation and Similarity

Event is a group of hashtags and is represented in the same way as hashtags, i.e.,

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

4.2 Clustering Hashtags

STREAMCUBE is able to handle content-evolving hashtags. Hashtags are dynamic in nature as described with the #merkel example earlier. When performing clustering taking hashtag as data points, the problem that we face is that we can easily cluster static data points in the vector space but how do we cluster moving points in our vector space. For solving this problem, the authors of the assigned paper divided the clustering problem into two parts: clustering content-evolving hashtags and clustering static hashtags. In general the idea is that every time a new hashtag describing an event comes into our vector space, it is treated as a moving point in the vector space until it has enough tweets containing the same hashtag and related hashtags around it to be considered as an event. Once, the hashtag gains enough tweet support to be considered as an event, its status is changed from inactive to active and it starts getting treated as a static data point. Once it starts getting treated as a static data point in the vector space then it can be easily clustered with existing clusters or may form its own new cluster. The following sub-sections describe the two parts of the clustering problem in detail:

4.2.1 Clustering Content-Evolving Hashtags

The HASHTAG-CLUSTER-DYNAMIC algorithm has been designed to handle content-evolving hashtags. The Algorithm 2 gives the pseudo code of this algorithm. It has three main parts:

i.) **Handling new hashtags:** The lines 1-2 describe the situation as a new hashtag emerges. Whenever a new hashtag emerges, it is best to treat it as an outlier since there are many tweets around which are users' personal tweets or spamming tweets which are non-relevant. The status of the hashtag keeps inactive which means that it is free to move around and will not be clustered.

ii.) **Handling inactive events:** As new tweets keep coming in, the events keep on getting updated. Then the algorithm checks the status of each inactive event to see if its status can be changed to active and whether it can be clustered. The authors of the paper used the metric that a hashtag needs to occur in at least 30 tweets around it for it to be considered active. Once an event's status is changed to active, it starts getting treated as a static data point in vector space and the HASHTAG-CLUSTER-STATIC gets called to cluster it. The relevant lines of the pseudo-code for this part are the lines 4-7.

iii.) **Handling active events:** Once an event is active, the algorithm checks whether we can split or merge the event, meaning that in the split operation the algorithm checks whether the algorithm can be removed from a cluster and re-clustered or whether it can be merged with another existing cluster. This is important because as time passes, similar hashtags might become dissimilar and dissimilar hashtags could become similar. An example for this could be #obama and #romney, which represent the two politicians Barack Obama and Mitt Romney. During the US Presidential Elections of 2012, these two politicians were the main presidential candidates and hence their hashtags used to quite often co-occur together at that time in election tweets. But after the elections, once Barack Obama become the US President, #obama started getting mentioned in events related to the policies he made or for example during the time when he got the Nobel Peace Prize, #obama co-occurred more together with #nobel and #peace. The merging of events has been described later in the Spatio-temporal aggregation section.

4.2.2 Clustering Static Hashtags

The algorithm used for clustering static hashtags is the HASHTAG-CLUSTER-STATIC algorithm whose pseudo-code is given in the Algorithm 3. The algorithm has the following two steps:

i. In every update of the event set, the events for which the status has been changed to active are checked and the closest neighbor for the event is found out. This is because the nearest neighbor would be the one which would be the best candidate for absorbing the new event.

ii. After finding out the nearest neighbor, the similarity of the event with its nearest neighbor is computed and if the similarity exceeds a particular threshold then it gets absorbed to the existing event and otherwise it forms a new event.

The good thing about this algorithm is that the algorithm is a single pass algorithm, that is, every time a new event emerges, it is clustered in a single pass of data and not in an iterative fashion containing loops over the data.

Another good design by the authors of the assigned paper[4] is that of fast nearest neighbor computation in which the candidates for the nearest neighbor search are pruned apriori if they have no co-occurrence with the current hashtag. Thus, the candidate set for neighbors gets reduced drastically and hence this method would be

Algorithm 3: HASHTAG-CLUSTER-STATIC(E, h)

Input: Event set E={e_1, e_2, ..., e_k}
Hashtag h
Output: Updated event set E

```
e = nearest-neighbor(E, h)
if sim(e, h) > e.threshold then
    add h to E as a new event
else
    add h to the existing event e
```

Algorithm 4: 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

```
for each word w_i order by max_partial(w_i) do
    for each event e in the invert list of w_i do
        sim(h, e) += w_i^h * w_i^e
    Let e^1 denote the most similar event
    Let e^2 denote the second similar event
    if sim(h, e^1) > sim(h, e^2) +
    sum_{j=i+1}^{n} max_partial(w_j^e) · w_j^h then
        return e^1
```

faster than if the algorithm just simply computed for each event the closest neighbor from all the set of existing events. This is useful as well in the Twitter Context because many hashtags co-occur with more than three thousand hashtags. The pruning algorithm is NEAREST-NEIGHBOR algorithm whose pseudo-code is given in the Algorithm 4. The hashtags have already been described as a combination of the weighted normalized vectors of words and hashtags. All the hashtags are stored in an inverted index, which is common to use in information retrieval applications. Thus, if we have a current hashtag h = (0.8, 0.2) and two events e_1 = (0.8, 0.1) and e_2 = (0.1, 0.2) as illustrated in the Figure 4, then by scanning the first row of the inverted index we get sim(h, e_1) = 0.64 and sim(h,e_2) = 0.08. The maximum partial value for sim(h,e_2) can be 0.2 and even if we add this to the previous scan value we would get sim(h,e_2) = 0.28 which would still be less than sim(h,e_1) value obtained from the first scan which is 0.64. Since, we now know that even if in the second scan we get the maximum partial value still it can never exceed sim(h,e_2) and hence we conclude that e_1 is the nearest neighbor and this saves us from performing one database scan. This speeds up the clustering process.

4.3 Spatio-temporal Aggregation

By Spatial aggregation can be understood as zooming out on maps. An example of zooming out could be moving from events in a district to events in country. Temporal aggregation usually means moving from a week's event report to a month's event report. In STREAMCUBE, suppose that two events e_1 and e_2 need to be merged from the event sets E_1 and E_2 respectively. Then if the two events are represented by the same group of hashtags, meaning that they are probably equal, then they are merged. If the two events do not have any hashtags in common, then the events are probably different and they are kept separate. The only interesting case here is when the two events e_1 and e_2 have a partial overlap of hashtags. In this case, the system first matches the hashtags in common. Then it checks whether the rest of the hashtags can be absorbed to form

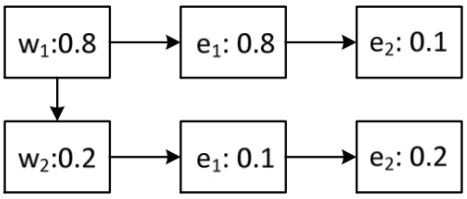


Figure 5. Nearest Neighbor Search

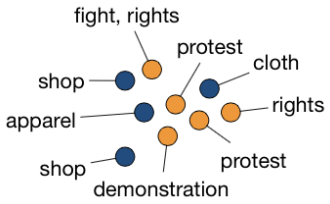


Figure 6. Tweets around the City Center

a single event as well. For example, given two events e_1 = (h_1, h_2, h_3, h_4) and e_2 = (h_3, h_4, h_5). First, the system merges h_3 and h_4. Then the system checks whether h_1, h_2 and h_5 are within the absorbance threshold of h_3 and h_4. If h_1, h_2 and h_5 cannot be merged, then they are re-clustered.

5. EVENT RANKING

The authors of the assigned paper [4] used the following measures to rank the events: Popularity, Burstiness and Localness. These three ranking factors are combined together to form the ranking function.

5.1 Ranking Factors

5.1.1 Popularity

Given a time frame t and a region r the popularity of an event, popularity(e) is defined as:

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

Here freq(e) represents the frequency of the event which is measured by the number of tweets describing this events and N represents the total number of tweets.

5.1.2 Burstiness

Burst events are events which exhibit a high popularity during a very short period of time. An event can be described as bursty only when it gets tweeted in a particular time frame way much higher than the historical frequency of tweets for that event. For example, the City Center of Hannover has a lot of shops and is sometimes the place chosen for a protest. So, in some day there might be tweets posted by people related to the shopping in the City Center and some protest going on in the City Center. This is illustrated in the Figure 6. But it might be that the number of tweets related to shopping is not unusually high when compared to the previous days since many people shop in the market and post shopping related tweets everyday whereas protests occur rarely in the City Center and hence the tweets related to protest are unusually high when compared to the history of protest related tweets in the city center.

In this case, the protest would get classified as a bursty event whereas shopping will not be classified as a bursty event. The common way to model historical popularity is using Gaussian distribution as shown in Figure 7a. Formally, Burstiness is defined as follows:

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

Here p_t is the popularity of the event e in the time frame t, μ is the mean and σ is the standard deviation.

5.1.3 Localness

Localness is defined in the same way as burstiness as in Figure 7b, the only difference being that it represents high popularity of an event in a particular region instead of a particular time frame. It can be formally given as:

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

Here p_r is the popularity of the event in the region r, μ is the mean and σ is the standard deviation.

5.2 Ranking Score

After finding the popularity, burstiness and localness measure of an event e, the total ranking score is computed as follows:

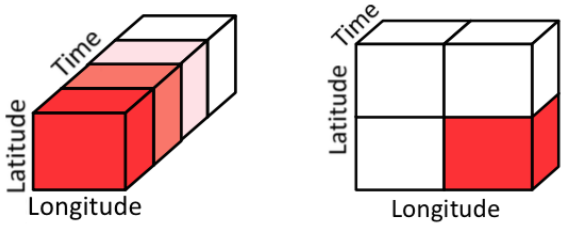
$$\begin{aligned} score(e) &= \sum_{i=1}^{k} w_i score(h_i) \\ &= \alpha \sum_{i=1}^{k} w_i pop(h_k) + \beta \sum_{i=1}^{k} w_i burst(h_k) + \gamma \sum_{i=1}^{k} w_i local(h_k) \\ &= \alpha \cdot pop(e) + \beta \cdot burst(e) + \gamma \cdot local(e) \end{aligned}$$

The values of α, β, and γ and can be varied according to the interest of the users. For example, for a user interested more in the local events, it makes sense to keep a higher value of γ when compared to β and α. For a user interested in most popular events for a year, α should be set to a higher value when compared to β and γ. It is also possible to train the system on the values to set for α, β and γ given a particular query by developing a classification model with the historical user preferences for types of query inputs as the training set. This could have been implemented in the system but the authors decided against it in order to provide a more flexible approach to the users to select for the same query the kind of output they wanted. This is because two users selecting the same region and the same time period may prefer different results. The first one may be more interested in burst events whereas the second may be more interested in local events. However, this approach leads to the system computing all the top-k events based on popularity, burstiness and localness for every query and this may not always be more efficient than by just fetching events for a query by setting α, β and γ according to the user preferences from the query logs.

6. EXPERIMENTAL STUDY

6.1 Dataset

The authors of the assigned paper [4] crawled 9 million tweets with 2 million hashtags from Twitter streaming API from December 2013 to January 2014 and stemmed all the tweets using Porter Stemmer method. The authors claim that since more and more people are accessing Twitter from mobile devices, more and more tweets are increasingly getting geo-tagged as well. The authors of the paper [11] claim that only 2 per cent of the tweets in Twitter are geo-tagged whereas the authors of the system EvenTweet [7] claim



(a) Burstiness (b) Burstiness

Figure 7. Localness

in their paper that only 1 per cent of the Tweets are geo-tagged. The authors of the assigned paper day that for users that still tweet from devices without GPS, their profile information can be used for mapping the tweets to a location as users indicate their country and city in their profiles. For users that don't even indicate that, the context information of the tweet must then be taken into account to approximate the location. For example, for a user tweeting about some strike in Hannover there is a high chance that the user lives in Hannover. For a user tweeting about the US Elections with the tweet not being geo-tagged as well as no location information existing in the profile the tweet gets mapped randomly based on density distribution of the tweets. Since, each tweet gets mapped at the lowest level of the hierarchy, which in our case is district, this might be problematic as this tweet would incorrectly contribute to the localized events query from that district since it was not originally from there. Further, for American users living in Germany tweeting about the US Elections, mapping them to USA knowing their location is not accurate. Also, if the tweet is geo-tagged, mapping it to Germany won't be of any help either since the event is irrelevant to Germany. Also, small cities in USA would have smaller Twitter users than big cities like New York or San Francisco, so for a more significant event from a small city there might be less number of tweets whereas for a less significant event from a big city there might be a large number of tweets. But in the overall ranking for the entire US, the more significant event should be displayed over the less significant event. So, it would be good if the authors made use of some kind of normalization measure based on the number of users from each city in the country when finding significant events for the entire country. Going by the studies from the other papers, since only 1 or 2 per cent of the tweets are geo-referenced the system might end up assigning locations to 98 or 99 per cent of tweets which might not be in most of the cases be accurate which means that the events coming in the result might not actually be from the location. That is why in the EvenTweet system[7] all the geo-tagged and non-geo-tagged tweets were used for finding words best describing events but only geo-tagged tweets were used for estimating the spatial distribution of words. This approach could be tried out by the STREAMCUBE system and then the changes in event ranking could be analyzed to assess whether it was actually useful to assign location to non-geo-tagged tweets.

6.2 Evaluation Methodology

The authors of the assigned paper [4] performed both Qualitative and Quantitative Evaluation of the Results to check the clustering and ranking quality of the results. Also, the scalability and the memory usage of the algorithm was evaluated to check whether the algorithm worked fine for large datasets when working with limited memory.

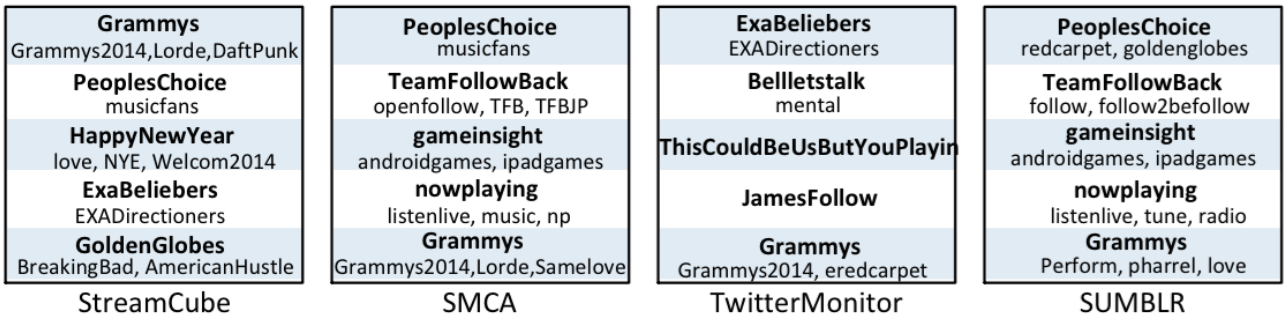


Figure 8. Detected Events by Different Methods in January 2014

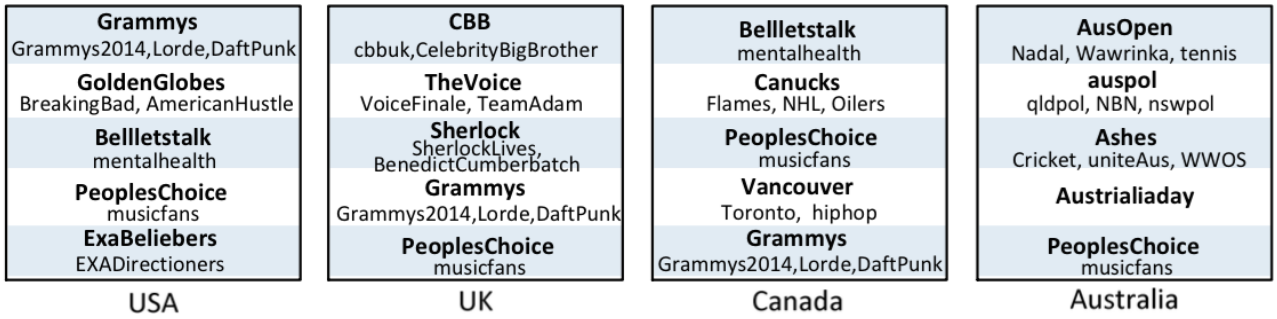


Figure 9. Events in Four Different Countries in January 2014

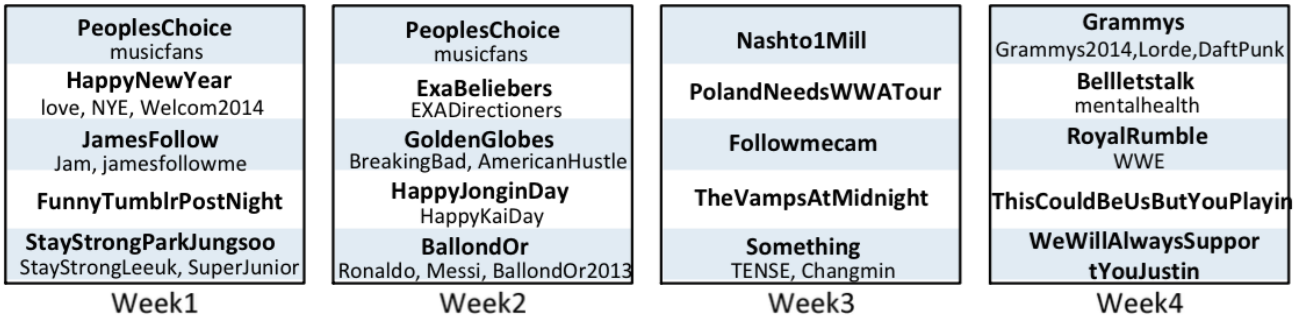


Figure 10. Events in four weeks of January 2014

6.3 Baselines

STREAMCUBE was compared against other tweet clustering and event detection systems, namely, SUMBLR [9], SMCA [8] and TwitterMonitor [6]. Since these systems were already described in the Related Work section, they will not be described here again.

6.4 Qualitative Evaluation

Figure 8 illustrates the detected events by the different systems in January 2014, Figure 9 the events in four different countries in January 2014 and Figure 10 the events in four weeks of January 2014.

Looking at Figure 8, we see that SMCA and SUMBLR both detect 'gameinsight' and 'nowplaying' which are popular in every month. This is because these systems just perform batch clustering instead of finding out burst events and local events. TwitterMonitor provides better results than SMCA and SUMBLR since it performs burst keyword identification, but it delivers results like 'Bellletstalk' with the most important keyword as 'mental'. 'Bellletstalk' is an event about mental health and here mental does not give a clear indication to the user about the topic of the event. This shows that burst keywords carry less information and have lower quality than burst hashtags. Further, burstiness may always not be the desired measure as users might be interested in finding out local events as well. In this regard, STREAMCUBE performs the best among all the four systems and seems to provide human readable and best results.

We can observe from Figure 10 that STREAMCUBE does indeed provide local events to that country as 'GoldenGlobes' which is an award event is quite popular in the USA, 'TheVoice' which is a singing competition is quite popular in UK, 'Canucks' is a popular ice hockey team from Canada and 'AusOpen' refers to the Australian Open Tennis Tournament in Australia.

Table 2. Ranking Quality

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

From the Figure 10, we can deduce that STREAMCUBE is able to provide results for different time granularities as in the first week of January, we always have the New Year event which is quite popular. FIFA Ballon d'Or is a Football Award Ceremony taking place in the second week of January and Royal Rumble a WWE Tournament taking place in the fourth week of January. These two events are detected in the second and fourth week as well by STREAMCUBE.

6.5 Quantitative Evaluation

The authors of the assigned paper crowdsourced the initial candidate set to 10 people to rank the top-10 events in each set. Based on the ranking done by the human evaluators, the authors then computed the Average Precision(AP) and the Mean Average Precision(MAP) for the results computed by all the four systems. Given a ranked event list E = (e_1, e_2, ..., e_n), the Average Precision (AP) is defined as

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

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. After finding out AP, we can find out MAP which is the mean of all APs:

$$MAP = \frac{\sum_{i=1}^{N} AP_i}{N}$$

STREAMCUBE had achieved the highest MAP among all the methods as is shown in Table 2.

6.6 Scalability

As we can observe from Figure 11, TwitterMonitor is the slowest among all the systems. This is because TwitterMonitor clusters keywords in Tweets and the set of keywords in tweets is much bigger than the set of hashtags. SMCA and SUMBLR have better performance since they initially perform batch clustering and later whenever tweets from a stream come in, these systems incrementally cluster them. STREAMCUBE has the shortest running time as it makes use of a single pass algorithm for clustering and uses a nice nearest neighbor search algorithm.

6.7 Memory Usage

The authors measured the memory usage by comparing the index size with the different data size. TwitterMonitor maintains a similarity matrix for all pair of keywords and hence, consumes most memory. Since SUMBLR performs batch clustering of tweets using tweets as data points, initially it consumes more memory when compared to SMCA as SMCA performs batch clustering of hashtags of tweets. But at a later stage, SMCA consumes more memory than SUMBLR as new hashtags keep emerging. Also, SUMBLR is able to periodically detect and remove outdated clusters. STREAMCUBE consumes the least amount of memory as it performs hashtag based clustering in the early stage and at a later stage only keeps the current six hour time frame in memory and flushes the outdated time frames to the disk. Hence, memory consumption remains stable for STREAMCUBE for large datasets.

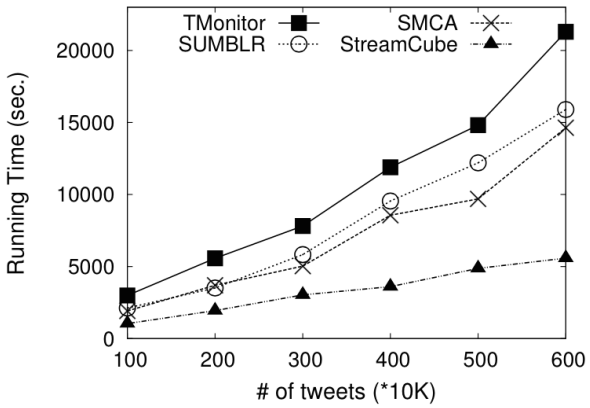


Figure 11. Scalability

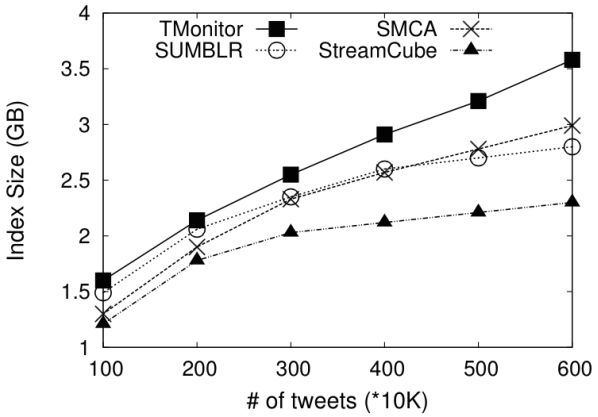


Figure 12. Memory Usage

7. CONCLUSION

The authors of the assigned paper have developed a truly state-of-the-art system which allows users to explore events with different time and space granularity according to their interests. The previous approaches had either allowed users to find burst events or local events but not both at the same time. It also makes uses of hashtag based clustering considering the dynamic nature of hashtags as well. Most of the previous approaches never made use of hashtags in tweets as the data points even though hashtags seem to provide more semantic expressivity than extracted keywords in tweets. Previous approaches which indeed did make use of hashtag based clustering failed to take their dynamic nature into account and treated hashtags as static data points. The design of the data warehousing structure for STREAMCUBE which is inspired by the data cube and quad tree structure along with the nearest neighbor search algorithm makes data storage and hashtag clustering very efficient. However, the drawbacks of the system could be that since less than 2 per cent of the tweets are geotagged, assigning location to all the tweets and then making use of all the tweets to identify tweets may not work well as location assigned to non-geotagged tweets may not always be accurate. It might be better to make use of geo-tagged as well as non-geo-tagged-tweets for hashtag clustering but make use of only geo-tagged tweets for event

ranking. Further, instead of using the complete globe in the space hierarchy for every six hour time frame, it might be better to begin with a local space, say a country like the USA, and then keep on adding new countries to this space as tweets emerge from new countries. It might also be useful to make a classifier model and train the results using query logs and the logs of the user preferences for particular regions and time frames to return the best possible events for a time frame instead of finding all the events according to popularity, burstiness and localness and then allowing the user choose according to his preferences. Possible extensions to this event cube could be allowing users to explore not just according to the space and time dimension but also the topic dimension. Thus, another topic dimension could be added and algorithm could be extended to identify combinatorial patterns so that users can explore events by topic. The authors have also proposed an alert mechanism as an extension to the STREAMCUBE system which would regularly push out event information to the users so that the users do not have to query and monitor the system from time to time.

8. REFERENCES

[1] M. Conover, J. Ratkiewicz, M. Francisco, B. Goncalves, A. Flammini, and F. Menczer, "Political polarization on twitter," in Proc. 5th Intl. Conference on Weblogs and Social Media, 2011.

[2] T.Sakaki, M.Okazaki, and Y.Matsuo ,"Earthquake shakes twitter users: real-time event detection by social sensors," in WWW, 2010.

[3] Salaheldeen, H.; Nelson, M. L.: "Losing My Revolution: How Many Resources Shared on Social Media Have Been Lost?" JCDL, Washington, USA, 2012.

[4] W. Feng et al., "STREAMCUBE: Hierarchical spatio-temporal hashtag clustering for event exploration over the Twitter stream," 2015 IEEE 31st International Conference on Data Engineering, Seoul, 2015, pp. 1561-1572.

[5] J. Han, Data Mining: Concepts and Techniques. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005.

[6] M. Mathioudakis and N. Koudas, "Twittermonitor: trend detection over the twitter stream," in SIGMOD Conference, 2010, pp. 1155–1158.

[7] Michael Gertz et. al. "EvenTweet: Online Localized Event Detection from Twitter" 39th International Conference on Very Large Data Bases, August 26-30th, Trento, Italy.

[8] 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.

[9] L. Shou, Z. Wang, K. Chen, and G. Chen, "Sumblr: continuous summarization of evolving tweet streams," in SIGIR, 2013, pp. 533–542.

[10] T. Lappas, M. R. Vieira, D. Gunopulos, and V. J. Tsotras, "On the spatiotemporal burstiness of terms," PVLDB, vol. 5, no. 9, 2012.

[11] Mohamed F. Mokbel and Amr Magdy. "Microblogs Data Management Systems: Querying, Analysis, and Visualization". In Proceedings of the ACM International Conference on Management of Data, SIGMOD 2016, San Francisco, CA, USA, June, 2016.