# TIMELINE GENERATION AND RECAPITULATION OF PROGRESSIVE TWEET STREAMS IN A DISTRIBUTED SYSTEM

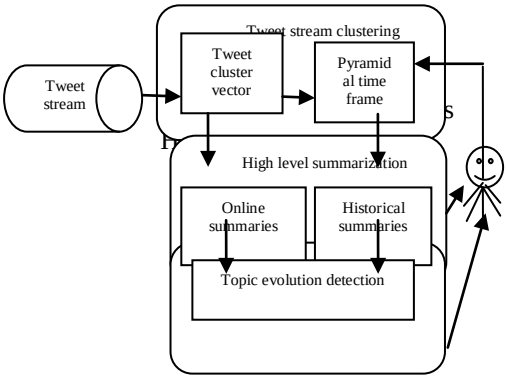
Amila.H[1] | Kirthana.S[2]| Nithra.G[3]| Neelamegam.G[4]
[1](Department of Computer science and Engineering, Chennai, India, amila7280@gmail.com)
[2](Department of Computer science and Engineering, Chennai, India, kirthanaviews@gmail.com)
[3](Department of Computer science and Engineering, Chennai, India, nithu.sekar635@gmail.com)
[4](Department of Computer science and Engineering, Chennai, India, gneelamegham@gmail.com)

***Abstract—*** *Short Message Services such as Tweets are created and shared in an outré rate. Raw form of tweets is newsy and also paralyzing. Tweets contain blimp of raspy and superfluity for both end users and data begetter. A novel continuous summarization skeleton called Sumblr (continuouS sUMmarization By stream cLusteRing) to overcome the problem. Here multi topic version of Sumblr is used for summarizing and clustering of large datasets in a Distributed System. Traditional summarization methods were focused on static and small scale datasets in a single system. But Sumblr is designed to deal with dynamic, fast arriving and large scale datasets. Three major components are proposed in the framework; first the tweets are clustered using tweet stream clustering algorithm and maintain clear statistics in an data structure called Tweet Cluster Vector (TCV), Second a TCV-Rank Summarization technique is developed for generating online and historical summaries of arbitrary time durations, Third an effective topic evolution detection method is designed for monitoring summary based variations to produce timeline automatically. Experiments on large scale real tweets demonstrate the efficiency and effectiveness of the framework.*

***Keywords— Tweet stream, continuous summarization, historical summary and online summary.***

## 1. INTRODUCTION (*HEADING 1*)

Social media services such as Twitter, Face book and Wechat resulted in the whump of short text messages. Twitter, for instance receives over 400 million tweets per day which involves invaluable source of news, blogs, opinions and more. Raw form of tweets is newsy and also paralyzing. For instance, search for a hot topic in Twitter may yield millions of tweets, spanning weeks. Many number of tweets are available, picking out the important contents would be ordeal. The emergence topics of tweets may be uncovered due to continuous arrival of new tweets at a difficult rate. A solution to information overload problem is summarization. Summarization represents a set of documents consisting of several sentences. Tweet summarization requires functionalities which differ from traditional summarization. Temporal feature of arriving tweets has also considered in tweet summarization. Let us illustrate using an example of usage of such a system. Consider a user interested in topic related tweet streams, for example, tweets about “Fastrack”. A tweet summarization system is continuously monitoring “Fastrack” related tweets producing a real time timeline of the tweet stream. As illustrated in Fig. Fig. 1, a user may explore tweets based on a timeline. Given a timeline range, the summarization system may produce a sequence of time stamped summaries to highlight points in the stream. The system will enable the user to learn major news/ discussion related to “Fastrack” without having entire tweet stream. Given the big picture about topic evolution about “Fastrack”. Drilldown and rollup summary is used. For this summarization the system supports the following two queries: summaries of arbitrary time duration and real time/range timelines. A new summarization method called continuous summarization is used. A good solution for continuous summarization has to address some problems (1) Efficiency, (2) Flexibility, (3) Topic evolution. Existing summarization method cannot satisfy the above three requirements because, they mainly focus on static and small sized datasets and hence are not efficient and scalable for large datasets and data streams.




*A Framework of Sumblr*

A novel summarization framework called Sumblr is used. The framework consists of three main components, they are the Stream Clustering module, the Large level Recapitulation module and the Timeline Compeers module. In tweet stream clustering module, a tweet stream clustering algorithm is designed, an online algorithm allowing for effective clustering of tweets. This algorithm employs two data structures. The first one is tweet cluster vector and the next one is pyramidal time frame. The high

level summarization model supports two kinds of summaries: online and previous summaries. (1)To produce online summaries, a TCV-Rank summarization algorithm is computed. (2)To compute historical summary, first retrieve two historical cluster snapshots from the PTF.
The main contribution of this work is as follows:

- A continuous tweet stream summarization framework, namely Sumblr.
- A novel data structure called PCV for stream processing and TCV-Rank algorithm, different summarization.
- A topic evolution detection algorithm is proposed which produces timelines.

## 2 RELATED WORKS

In review related works it includes stream data clustering, document summarization, timeline detection, and other mining tasks.

### *2.1 Stream Data Clustering*

Stream data clustering has been widely studied in the literature. In BIRCH clusters, data is based on an in memory structure called CF tree. A growable clustering framework which chooses and stores important parts. CluStream is the most classical clustering methods. It consists of an online small clustering component and an offline large clustering component. The pyramidal time frame was also proposed to recall historical micro clusters for different time summaries.

A variety of services on the Web have posed requirements for text clustering. A few algorithms are proposed to handle the problem. some techniques adopt separation based techniques. These approach lose to provide efficient analysis on clusters formed at different time summaries.

CluStream is used to generate time based clustering, results for text and categorical data streams. This algorithm relies on an online phase to produce a large number of "micro clusters" and an offline phase to recluster them. In contrast, tweet clustering algorithm is an online method without clustering. And in the context of tweet summarization, adapt the online clustering phase by introducing the new structure TCV, and discarding the number of clusters to guarantee efficiency and quality TCVs.

### *2.2 Document Summarization*

Document summarization can be categorized in two summarization methods. The former selects sentences from the documents, while in turn generate phrases and sentences that may not present in the real documents. Extractive document summarization has gained many people attention. Most of them assign scores to sentences.
Document summarization has been studied for years summarization is still in its infancy. All existing document summarization methods mainly deal with small and static data sets, and rarely pay attention to efficiency issues.

### *2.3 Timeline Detection*

The demand for analyzing massive contents in social media, fuels the development in visualization techniques. Timeline is a techniques which can perform analysis tasks easier and faster which made early efforts in this area, using timelines to explore the 2008 Presidential Debates by Twitter sentiment. The evolutionary timeline summarization (ETS) is used to compute evolution timelines similar to the timeline, which consists of a series of time stamped summaries. However, the dates of summaries are determined by a predefined timestamp set.

## 3 PRELIMINARIES

A data model for tweets, introduces two important data structures: the tweet cluster vector and pyramidal time frame.

### *3.1 Tweet Representation*

Generally, a document is represented as a textual vector, where the value of each dimension is the TF-IDF score of a word. However, tweets are not only textual, but also have temporal nature; a tweet is strongly correlated with its posted time. In addition, the importance of a tweet is affected by the author's social influence. To estimate the user influence, build a matrix based on social relationships among users, and compute the User Rank. A definition for a tweet $t_i$ as a tuple :( $tv_i$, $ts_i$, $w_i$), where $tv_i$ is the textual vector, $ts_i$ is the posted time stamp and $w_i$ is the User Rank value of the tweet's author

### *3.2 Tweet Cluster Vector*

During tweet stream clustering, it is necessary to maintain statistics for tweets to facilitate summary generation. A new data structure called tweet cluster vector, which keeps information of tweet cluster.
**Definition 1.** For a cluster C containing tweets t1, t2, . . . , tn, its tweet cluster vector is defined as a tuple:
TCV (C) = (sum _v, wsum _v, ts1, ts2, n, ft _set), where

- sum _v=$\sum_{i=1}^{n} tv_i/||tv_i||$ is the sum of normalized textual vectors.
- wsum_v=$\sum_{i=1}^{n} w_i.tv_i$ is the sum of weighted textual vector.
- ts1=$\sum_{i=1}^{n} ts_i$ is the sum of timestamps.
- ts2= $\sum_{i=1}^{n} (ts_i)^2$ is the quadratic sum of timestamps,
- n is the number of tweets in the cluster,
- ft_set is a focus tweet set of size m, consisting of the closest m tweets to the cluster centroid.

The form of sum_v is used for ease of presentation. It only stores the identifiers and sums of values of the words occurring in the cluster. The same convention is used for wsum_v. To select tweets into ft_set, use cosine similarity as the distance metric.

From the definition, derivation for the vector of cluster centroid (denoted as cv)

$$cv= (\sum_{i=1}^{n} w_i.tv_i )/n=wsum\_v/n$$

The definition of TCV is an extension of the cluster feature vector. Besides information of data points (textual vectors), TCV includes temporal information and representative original tweets. TCV structure can also be updated in an incremental manner when new tweets arrive.

*3.3 Pyramidal Time Frame*

To support summarization over user defined time durations, it is crucial to store the maintained TCVs at particular moments, which are called snapshots. While storing snapshots at every moment is impractical due to huge storage overhead, insufficient snapshots make it hard to recall historical information for different durations. This dilemma leads to the incorporation of the pyramidal time frame.

**Definition 2.** A pyramidal time frame stores snapshots at differing levels of granularity depending on the recency. Snapshots are stored into different orders varying from 0 to [$\log_\alpha$ (T)], where T is the time elapsed since the beginning of the stream and a is an integer ($\alpha$> 1). A particular order of snapshots defines the level of granularity in time at which the snapshots are maintained. The snapshots of different orders are maintained as follows:

- Snapshots of the $i^{th}$ order occur at time intervals of $\alpha^i$ each snapshot of the $i^{th}$ order is taken at a moment in time when the timestamp from the beginning of the stream is exactly divisible by $\alpha^i$.
- At any given moment in time, only the last $\alpha^l$ +1(l>=1) snapshots of order i are stored.

According to the definition, PTF has two properties: (1)The maximum order of any snapshot stored at timestamp T since the beginning of the stream is $\log_\alpha$(T); (2) The maximum number of snapshots maintained at T is ($\alpha^l$+1)*$\log_\alpha$(T). These properties are crucial for system performance. Taking more snapshots (by using a larger $\alpha$ or l) offers better accuracy of time duration approximation, but mean while causes larger storage overhead. A need to strike a balance between duration accuracy and storage space. It is only maintain the current clusters in main memory, and store all historical snapshots in the PTF on disk.

## 4 THE SUMBLR FRAMEWORK

Framework consists of three main modules: the tweet stream clustering module, the high level summarization module and the timeline generation module. Each of them are given in detail.

*4.1 Tweet Stream Clustering*

The tweet stream clustering module maintains the online statistical data. Given a topic based tweet stream, it is able to efficiently cluster the tweets and maintain compact cluster information.

*4.1.1 Initialization*

At the start of the stream, collect a small number of tweets and use a k-means clustering algorithm to create the initial clusters. The corresponding TCVs are initialized according to Definition 1. Next, the stream clustering process starts to incrementally update the TCVs whenever a new tweet arrives.

*4.1.2 Incremental Clustering*

Suppose a tweet t arrives at time ts, and there are N active clusters at that time. The key problem is to decide whether to absorb t into one of the current clusters or upgrade t as a new cluster. First find the cluster whose centroid is the closest to t. specifically, we get the centroid of each cluster, compute its cosine similarity to t, and find the cluster $C_p$ with the largest similarity (denoted as MaxSim (t)). Note that although $C_p$ is the closest to t, it does not meant naturally belongs to $C_p$. The reason is that t may still be very distant from $C_p$. In such case, a new cluster should be created. Note that although $C_p$ is the closest to t, it does not meant naturally belongs to $C_p$. The reason is that t may still be very distant from $C_p$. In such case, a new cluster should be created.

Algorithm 1 describes the overview of incremental clustering procedure.

**Algorithm 1**

```
Input: a cluster set C_set
1 while! stream.end() do
2   Tweet t= stream.next();
3   choose Cp in C_set whose centroid is
    closest to it;
4   if MaxSim (t) < MBS then
5    create a new cluster Cnew = {t};
6    C_set.add (Cnew);
7   else
8    update Cp with t;
9   if TScurrent %( αi) ==0 then
10  store C_set into PTF;
```

During incremental clustering, assume there are N active clusters, the computational cost of finding the closest cluster for every new tweet is O (Nd), where d is the vocabulary size. In addition, the complexity of computing and updating TCV is O (d) and O (md) respectively, where m is the size of focus set. Then the total cost is O ((N+m) d). Because m and d are static, the computational cost depends on N. Similarly, the storage costs in disk (TCV snapshots) and memory (current TCVs) also depend on N.

Given the above analysis, g outdated clusters and merging similar clusters. Since the computational complexity of deletion is O (N) and that of merging is O ($N^2$), use the former method for periodical examination and use the latter method only when memory limit is reached.

*4.1.3 Deleting Outdated Clusters*

For most events in tweet streams, timeliness is important because they usually do not last for a long time. Therefore it is safe to delete the clusters representing these sub topics when they are rarely discussed. To find out such clusters, an intuitive way is to estimate the average arrival time (denoted as Avgp) of the last p percent of tweets in a cluster. However, storing p percent of tweets for every cluster will increase memory costs, especially when clusters grow big. Thus, employ an approximate method to get Avg.

*4.1.4 Merging Clusters*

If the number of clusters keeps increasing with few deletions, system memory will be exhausted. To avoid this, specify an upper limit for the number of clusters as Nmax. When the limit is reached, a merging process starts. The process merges clusters in a greedy way. First, sort all cluster pairs by their centroid similarities in a descending order. Then, starting with the most similar pair, try to merge two clusters in it. When both clusters are single clusters which have not been merged with other clusters, they are merged into a new composite cluster. When one of them belongs to a composite cluster (it has been merged with others before), the other is also merged into that composite cluster. When both of them have been merged, if they belong to the same composite cluster, this pair is skipped; otherwise, the two composite clusters are merged together. This process continues until there are only mc percentage of the original clusters left (mc is a merging coefficient which provides a balance between available memory space and the quality of remaining clusters).

**Definition 3 (Aggregation Operation).** Let C1 and C2 be two clusters and their TCVs be TCV (C1) and TCV (C2). When C1and C2 are merged together, the composite cluster's TCV (C1 Ụ C2) is given by

- $sum\_v = sum\_v_1 + sum\_v_2$
- $wsum\_v = wsum\_v_1 + wsum\_v_2$
- $ts1 = ts1_1 + ts1_2$
- $ts2 = ts2_1 + ts2_2$
- $n = n_1 + n_2$
- ft_set consists of the first m tweets in $ft\_set_1 U ft\_set_2$ , sorted by distance to the centroid.

*4.2 High-Level Summarization*

The high-level summarization module provides two types of summaries: online and historical summaries. An online summary describes what is currently discussed among the public. Thus, the input for generating online summaries is retrieved directly from the current clusters maintained in memory. On the other hand, a historical summary helps people understand the main happenings during a specific period, which means the need to eliminate the influence of tweet contents from the outside of that period. As a result, retrieval of the required information for generating historical summaries is more complicated, and this shall be on focusing on the following discussion. Suppose the length of user defined time duration is H, and the ending timestamp of the duration is $ts_e$. From PTF, it can retrieve two snapshots whose timestamps are either equal to or right before $ts_e$ and $ts_e$ _ H, respectively. Denote the timestamps by $ts_1$ and $ts_2$, and their cluster sets by S ($ts_1$) and S($ts_2$). Now the original duration [$ts_e$ _ H, $ts_e$] is approximated by [$ts_2$, $ts_1$]

**Definition 4 (Subtraction Operation).** Given a cluster $C_1$ in S ($ts_1$) and its corresponding cluster $C_2$ in S ($ts_2$), when $C_2$ is subtracted from $C_1$, their difference TCV ($C_1$ - $C_2$) is given by

- $sum\_v_1 = sum\_v_1 - sum\_v_2$
- $wsum\_v = wsum\_v_1 - wsum\_v$
- $ts1 = ts1_1 - ts1_2$
- $ts2 = ts2_1 - ts2_2$
- $n = n1 - n2$
- ft_set consists of tweets which exist in $ft\_set_1$ but not in $ft\_set_2$.

The above process eliminates the influence of clusters created before $ts_2$ on summary results. The final set of clusters after this process is the input for historical summarization.

**TCV-Rank Summarization Algorithm**

Given an input cluster set, to denote its corresponding TCV set as D(c). A tweet set T consists of all the tweets in the ft_sets in D(c). The tweet summarization problem is to extract k tweets from T, so that they can cover as many tweet contents as possible. Let us first describe this problem formally. Denote F = {$T_1$, T2,.., $T_t$} as a collection of non empty subsets of T, where a subset $T_i$ represents a sub topic and $|T_i|$ means the number of its related tweets. Suppose for each $T_i$, there is a tweet which represents the content of $T_i$'s sub topic. Then, selecting k tweets is equivalent to selecting k subsets.

**Algorithm 2**. TCV-Rank Summarization

**Input:** a cluster set D(c)
**Output:** a summary set S

```
1   S =ǿ , T={all the tweets in ft_sets of D(c)};
2   Build a similarity graph on T;
3   Compute LexRank scores LR;
4   Tc = {tweets with the highest LR in each cluster};
5   while |S| < L do
6   foreach tweet ti in Tc _ S do
7   calculate vi according to Equation
      (2);
8   select tmax with the highest vi;
9   S.add (tmax);
10  while |S| < L do
11  foreach tweet t'i in T _ S do
12  calculate v'i according to Equation
13  select t'max with the highest v'i;
14  S.add (t'max);
15   returns S;
```

To design a greedy algorithm to select representative tweets to form summaries (Algorithm 2). First, build a

cosine similarity graph for all the tweets in T. The maximum size of T is N _ m, where N is the number of clusters in D(c) and m is the size of ft_set. It is the upper bound because ft sets of some clusters (e.g., small clusters or clusters newly created) may not be full. Next, apply the LexRank method to compute centrality scores for tweets. LexRank is an effective static summarization method and is efficient for small sized data sets. But when data sets become large, its efficiency drops quickly. The tweet set T has at most Nm tweets (usually hundreds or thousands), so LexRank is suitable for the situation.

*4.3 Timeline Generation*

The core of the timeline generation module is a topic evolution detection algorithm which produces real time and range timelines in a similar way. The algorithm discovers sub topic changes by monitoring quantified variations during the course of stream processing. A large variation at a particular moment implies a sub topic change, which is a new node on the timeline. The main process is described in Algorithm 3. First bin the tweets by time (e.g., by day) as the stream proceeds. This sequenced binning is used as input of the algorithm. Then, loop through the bins and. append new timeline nodes whenever large variations are detected. Key problem is how to define the variation and detect when it becomes large. It describes three different kinds of variations and their detection methods.

**Algorithm 3**. Topic Evolution Detection

**Input**: a tweet stream binned by time units
**Output**: a timeline node set TN
1: TN=ǿ;
2: **while** !stream.end () **do**
3: Bin $C_i$= stream.next ();
4: **if** hasLargeVariation () **then**
5: TN.add (i);
6: **return** TN;

In this algorithm, the key problem is how to define the variation and detect when it becomes large. In what follows, will describe three different kinds of variations and their detection methods respectively.

*4.3.1 Summary-Based Variation*

As tweets arrive from the stream, online summaries are produced continuously by utilizing online cluster statistics in TCVs. This allows for generation of a real time timeline. Generally, when an obvious variation occurs in the main contents discussed in tweets (in the form of summary), can expect a change of sub topic (i.e., a time node on the timeline).

*4.3.2 Volume-Based Variation*

Though the summary-based variation can reflect sub-topic changes, some of them may not be influential enough. Since many tweets are related to users' daily life or trivial events, a sub topic change detected from textual contents may not be significant enough. Consider the use of rapid increases (or "spikes") in the volume of tweets overtime, which is a common technique in existing online event detection systems. A spike suggests that something important just happened because many people found the need to comment on it. Develop a spike finding method. As the input, the binning process in Algorithm 3 needs to count the tweet arrival volume in each time unit.

## 5 EXPERIMENTS

To evaluate the performance of Sumblr, present the experiments for summarization and timeline generation respectively.

*5.1 Experiments for Summarization*

*5.1.1 Setup*

Data Sets: Construct five data sets to evaluate summarization. One is obtained by conducting keyword filtering on a large Twitter data set used. The other four include tweets acquired during one month in 2012 via Twitter's keyword tracking API.4 ,do not have access to the respective users' social networks for these four, set their weights of tweets $w_i$ to the default value of 1.

1. Given time duration, first retrieve the corresponding tweet subset, and use the following three well recognized summarization algorithms to get three candidate summaries. ClusterSum clusters the tweets and picks the most weighted tweet from each cluster to form summary. LexRank [11] first builds a sentence similarity graph, and then selects important sentences based on the concept of Eigen vector centrality. DSDR models the relationship among sentences using linear reconstruction, and finds an optimal set of sentences to approximate the original documents, by minimizing the reconstruction error.
2. Next, for each subset, the final reference summary is extracted from three candidate summaries by using a voting scheme. The intuition is that if a specific tweet and its similar tweets appear many times in the candidate summaries, they can represent important content and should be chosen into the final summary. Specifically, for each tweet in each candidate summary, compute its similarities to the tweets from the other two candidate summaries. Then, the tweet votes to its most similar one and these two tweets form a "pair". After processing all the tweets in candidate summaries, sort them in descending order of their total votes. Delete the tweets whose pair members already exist at higher ranks. In each window contains a certain number (window size) of tweets which are summarized as a document. After that, the window moves forward by a step size, so that the oldest tweets are discarded and the new ones are added into the window. In this way, implement the sliding window version of the above three algorithms, namely ClusterSum, LexRank, and DSDR. The windows are dimensioned by number of tweets instead of time duration, because the number of tweets may

vary dramatically across fixed length durations, leading to very poor performance of the baseline algorithms.

*5.1.2 Flexibility*

Feature of Sumblr is the flexibility to summarize tweets over arbitrary time durations. This feature is provided by incorporating the PTF. The effectiveness of PTF depends on a and l. Fix a at 2 and show the results varying l. For consistency, extract a subset of one month period from each data set as the input stream. The interval between two successive snapshots (timestamp unit) is one hour. For a timestamp ts, evaluate the results for different durations with length len varying from 1 to 10 days. Report the average F-score score (ts) by interval of 48 hours.

## 6 CONCLUSION

A prototype called Sumblr which supported continuous tweet stream summarization. Sumblr employs a tweet stream clustering algorithm to compress tweets into TCVs and maintains them in an online fashion. Then, it uses a TCV-Rank summarization algorithm for generating online summaries and historical summaries with arbitrary time durations. The topic evolution can be detected automatically, allowing Sumblr to produce dynamic timelines for tweet streams.

## REFERENCES

[1]. Zhenhua Wang,Lidan Shou,Ke Chen,Gang Chen, and Sharad Mehrotra, "On Summarization and Timeline Generation for Evolutionary Tweet Streams " by IEEE ,2015

[2]. Yang Gao,Yue Xu,and Yuefeng Li, "Pattern-Based Topics for Document Modeling in Information Filtering" by IEEE,2015

[3]. Maryam Habibi and Andrei Popescu-Belis,"Keyword Extraction and Clustering for Document Recommendation in Conversations" by IEEE,2015.

[4]. AltugAkay,AndreiDragomir,Erlandson,"Network-Based Modeling and Intelligent Data Mining of Social Media for Improving Care" by IEEE,2015.