RESEARCH ARTICLE OPEN ACCESS

Evolving Stream Data Clustering and TCV Rank Summarization

P.Damodharan[1], Dr.C.S.Ravichandran[2]

1 (Department of Computer Science and Engineering, Akshaya College of Engineering and Technology, Coimbatore)

2 (Department of Electrical and Electronics Engineering, Sri Ramakrishna Engineering College, Coimbatore)

Abstract:

Tweet are being created short text message and shared for both users and data analysts. Twitter which receives over 400 million tweets per day has emerged as an invaluable source of news, blogs, opinions and more. Our proposed work consists three components tweet stream clustering to cluster tweet using k-means cluster algorithm and second tweet cluster vector technique to generate rank summarization using greedy algorithm, therefore requires functionality which significantly differ from traditional summarization . in general, tweet summarization and third to detect and monitors the summary-based and volume based variation to produce timeline automatically from tweet stream. Implementing continuous tweet stream reducing a text document is however not an simple task, since a huge number of tweets are worthless, unrelated and raucous in nature, due to the social nature of tweeting. Further, tweets are strongly correlated with their posted instance and up-to-the-minute tweets tend to arrive at a very fast rate. Efficiency—tweet streams are always very big in level, hence the summarization algorithm should be greatly capable; Flexibility—it should provide tweet summaries of random moment durations. (3) Topic evolution—it should routinely detect sub-topic changes and the moments that they happen.

Keywords — **Tweet Stream, summarization, Timeline, Topic evolution, summary**

I. INTRODUCTION

Growing attractiveness of micro blogging services such as Twitter, Weibo, and Tumblr has resulted in the explosion of the amount of short-text messages. Twitter, for instance, which receives over 400 million tweets per day1, has emerged as an invaluable source of news, blogs, opinions, and more.

Tweets, in their raw form, while being informative, can also be overwhelming. For instance, search for a hot topic in Twitter may yield millions of tweets, spanning weeks. Even if filtering is allowed, plowing through so many tweets for important contents would be a nightmare, not to mention the enormous amount of noise and redundancy that one might encounter. To make things worse, new tweets satisfying the filtering criteria may arrive continuously, at an unpredictable rate. One possible solution to information overload problem is summarization. Summarization represents restating of the main ideas of the text in as few words as possible Intuitively, a good summary should cover the main topics (or subtopics) and have diversity among the sentences to reduce redundancy. Summarization is widely used in comfortable arrangement, especially when users surf the internet with their mobile devices which have much lesser screens than PCs. Traditional document summarization approaches, however, are not as effective in the situation of tweets given both the big size of tweets as well as the fast and continuous nature of their arrival. Tweet summarization, therefore, requires functionalities which significantly differ from traditional summarization. In general, tweet summarization has to take into consideration the temporal feature of the arriving tweets. Consider a user interested in a topic-related tweet stream, for example, tweets about “Apple”. A

tweet summarization system will continuously monitor "Apple" related tweets producing a real-time timeline of the tweet stream. a user may explore tweets based on a timeline (e.g., "Apple" tweets posted between October to November). Given a timeline range, the document system may generate a series of current time summaries to highlight points where the topic/subtopics evolved in the stream. Such a system will effectively enable the user to learn major news/ discussion related to "Apple" without having to read through the entire tweet stream. Given the big picture about topic evolution about "Apple", a user may decide to zoom in to get a more detailed report for a smaller duration (e.g., from three hour) system may provide a drill-down summary of the duration that enables the user to get additional details for that duration. Such application would not only facilitate easy navigation in topic-relevant tweets, but also support a range of data analysis tasks such as instant reports or historical survey.

II. EXPERIMENTAL SETUP AND PROCEDURE

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 a scalable clustering framework which selectively stores important portions of the data, and compresses or discards other portions. CluStream is one of the most classic stream clustering methods. It consists of an online micro-clustering component and an offline macro-clustering component. A variety of services on the Web such as news filtering, text crawling, and topic detecting etc. have posed requirements for text stream clustering CluStream to generate duration- based clustering results for text and categorical data streams. However, this algorithm relies on an online phase to generate a large number of "micro-clusters" and an offline phase to re-cluster them. In contrast, our tweet stream clustering algorithm is an online procedure without extra offline clustering. And in the context of tweet summarization, we adapt the online clustering phase by incorporating the new structure TCV, and restricting the number of clusters to guarantee efficiency and the quality of TCVs.

2.1 Tweet Stream Initialization

At the start of the stream, we 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.

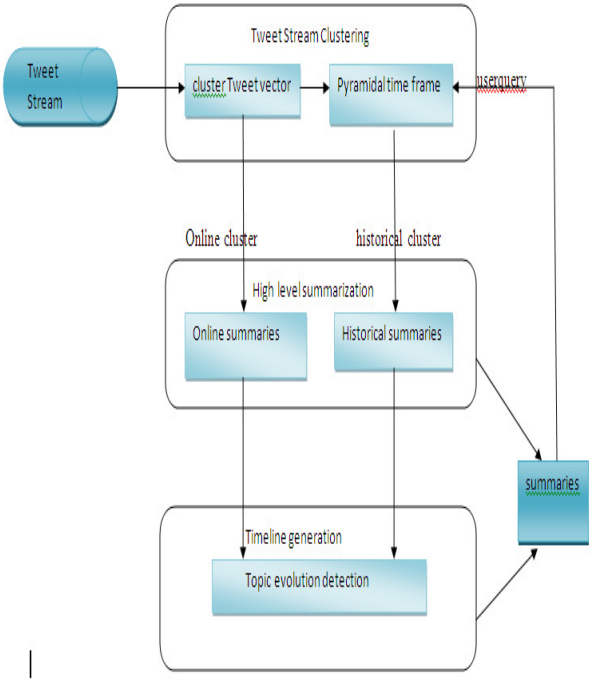

Fig. 1 Tweet Stream Initialization

2.2 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 attract into one of the in progress clusters or advance t as a new cluster. We first find the cluster whose centroid is the closest to t. specifically; we get the centroid of each cluster based on Equation (1), compute its cosine similarity to t, and find the cluster Cp with the largest similarity.

2.3 Deleting Outdated Clusters

For most events (such as news, football matches and concerts) 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, we employ an approximate method to get Avgp.

2.4 Merging Clusters

If the number of clusters keeps increasing with few deletions, system memory will be exhausted. To avoid this, we 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, we sort all cluster pairs by their centroid similarities in a descending order. Then, starting with the most similar pair, we 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).

III. THEORETICAL ANALYSIS

The demand for analyzing massive contents in social medias fuels the developments in visualization techniques. Timeline is one of these techniques which can make analysis tasks easier and faster. presented a timeline-based backchannel for conversations around events. proposed the evolutionary timeline summarization (ETS) to compute evolution timelines similar to ours, which consists of a series of time-stamped summaries. the dates of summaries are determined by a pre-defined timestamp set. In contrast, our method discovers the changing dates and generates timelines dynamically during the process of continuous summarization. Moreover, ETS does not focus on efficiency and scalability issues, which are very important in our streaming context. Several systems detect important moments when rapid increases or “spikes” in status update volume happen. Developed an algorithm based on TCP congestion detection, employed a slope-based method to find spikes. After that, tweets from each moment are identified, and word clouds or summaries are selected. Different from this two-step approach, our method detects topic evolution and produces summaries/timelines in an online fashion.

3.1 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. To this end, we consider the use of rapid increases (or “spikes”) in the volume of tweets over time, which is a common technique in existing online event detection systems . A spike suggests that something essential in a minute happened because a lot of people found the need to comment on it. In this part, we develop a spike-finding method. As the input, the binning process in Algorithm needs to count the tweet arrival volume in each time unit

IV. RESULTS AND DISCUSSION

We construct five data sets to evaluate summarization. One is obtained by conducting keyword filtering on a large Twitter data set. The other four include tweets acquired during one month in 2015 via Twitter’s keyword tracking.

Baseline methods: In Existing summarization methods have not been designed to handle

continuous summarization. In this way, here implement the sliding window version of the above three algorithms, namely Cluster Sum, LexRank, and DSDR.

In our experiments, we find similar trends in the comparison of precision, recall and F-score between the proposed approach and the baseline methods.

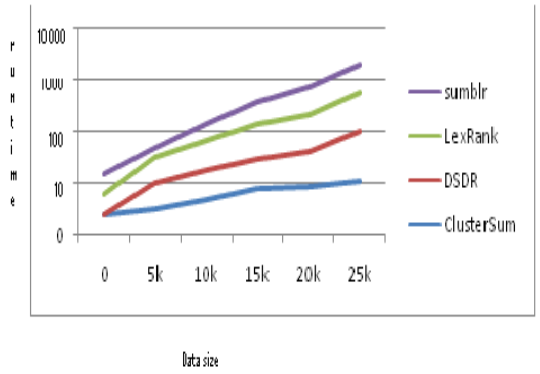

Fig. 2. F-Score

The demand for analyzing massive contents in social medias fuels the developments in visualization techniques. Timeline is one of these techniques which can make analysis tasks easier and faster. presented a timeline-based backchannel for conversations around events. proposed the evolutionary timeline summarization (ETS) to compute evolution timelines similar to ours, which consists of a series of time-stamped summaries. the dates of summaries are determined by a pre-defined timestamp set. In contrast, our method discovers the changing dates and generates timelines dynamically during the process of continuous summarization. Moreover, ETS does not focus on efficiency and scalability issues, which are very important in our streaming context. Several systems detect important moments when rapid increases or "spikes" in status update volume happen. Developed an algorithm based on TCP congestion detection, employed a slope-based method to find spikes. After that, tweets from each moment are identified, and word clouds or summaries are selected. Different from this two-step approach, our method detects topic evolution and produces summaries/timelines in an online fashion.

Table. 1 Basic Information of Dataset

Topics (filtering keywords)	#Tweets	Time Span
Obama	95,055	2014.07 - 2014.10
Chelsea	438,884	2015.07 - 2015.08
Arsenal Arsene Wenger	323,555	2015.07 - 2015.08
Tablet Smartphone Cellphone	231,011	2015.07 - 2015.08
Black Friday	124,684	2015.07 - 2015.08

V. CONCLUSION

We proposed 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. The experimental results make obvious the competence and success of our method. For future work, we aim to develop a multi-topic version of Sumblr in a spread system, and estimate it on more complete and large-scale data sets.

REFERENCES

1. *C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," in Proc. 29th Int. Conf. VeryLarge Data Bases, 2003, pp. 81–92.*
2. *T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An efficient data clustering method for very large databases," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 1996, pp. 103–114.*
3. *P. S. Bradley, U. M. Fayyad, and C. Reina, "Scaling clustering algorithms to large databases," in Proc. Knowl. Discovery Data Mining, 1998, pp. 9–15.*
4. *L. Gong, J. Zeng, and S. Zhang, "Text stream clustering algorithm based on adaptive feature selection," Expert Syst. Appl., vol. 38, no. 3, pp. 1393–1399, 2011.*

5. *Q. He, K. Chang, E.-P. Lim, and J. Zhang, "Bursty feature representation for clustering text streams," in Proc. SIAM Int. Conf. Data Mining, 2007, pp. 491–496.*
6. *J. Zhang, Z. Ghahramani, and Y. Yang, "A probabilistic model for online document clustering with application to novelty detection," in Proc. Adv. Neural Inf. Process. Syst., 2004, pp. 1617–1624.*
7. *S. Zhong, "Efficient streaming text clustering," Neural Netw., vol. 18, nos. 5/6, pp. 790–798, 2005.*
8. *C. C. Aggarwal and P. S. Yu, "On clustering massive text and categorical data streams," Knowl. Inf. Syst., vol. 24, no. 2, pp. 171–196, 2010.*
9. *R. Barzilay and M. Elhadad, "Using lexical chains for text summarization," in Proc. ACL Workshop Intell. Scalable Text Summarization, 1997, pp. 10–17.*
10. *W.-T. Yih, J. Goodman, L. Vanderwende, and H. Suzuki, "Multidocument summarization by maximizing informative contentwords," in Proc. 20th Int. Joint Conf. Artif. Intell., 2007, pp. 1776–1782.*