International Journal of Advancements in Technology

Selvaraj and Balaji, Int J Adv Technol 2016, 7:3
DOI: 10.4172/0976-4860.1000162

Research Article Open Access

Topic Evolutionary Tweet Stream Clustering Algorithm and TCV Rank Summarization

Selvaraj K* and Balaji S

Department of Computer Science and Engineering, Akshaya College of Engineering and Technology, India

***Corresponding author:** Selvaraj K, Department of Computer Science and Engineering, Akshaya College of engineering and technology, Tamil Nadu, India 641032, Tel: +91 9578817220; E-mail: Selvacse07@gmail.com

Received date: Nov 04, 2015; **Accepted date:** May 12, 2016; **Published date:** May 29, 2016

Abstract

Twitter which receives over 400 million tweets per day has emerged as an invaluable source of news, blogs, opinions and more. Our proposed work consist 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 a 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. Topic evolution-it should routinely detect sub-topic changes and the moments that they happen.

Keywords: Tweet stream; Summarization; Timeline; Topic evolution; Summary

Introduction

Growing attractiveness of microblogging 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 day, 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 and 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.

Twitter Summarization

Stream data 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 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 [1]. 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].

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. Next, the stream clustering process starts to incrementally update the TCVs whenever a new tweet arrives.

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, compute its cosine similarity to t, and find the cluster Cp with the largest similarity.

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.

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) Figure 1.

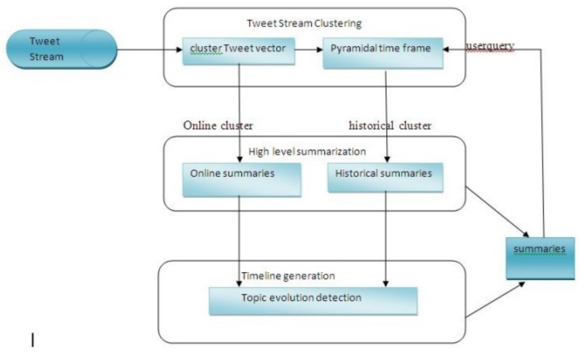

Figure1: The frame work of sumblr.

Related Work

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 we 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 our focus in the following discussion. Suppose the length of user-defined time duration is H, and the ending timestamp of the duration is tse.

Document/Microblog Summarization

Document summarization can be categorized as extractive and abstractive. The former selects sentences from the documents, while the latter may generate phrases and sentences that do not appear in the original documents. In this paper, we focus on extractive summarization. Extractive document summarization has received a lot of recent attention. Most of them assign salient scores to sentences of the documents, and select the top-ranked sentences. Some works try to extract summaries without such salient scores the symmetric non-negative matrix factorization to cluster sentences and choose sentences in each cluster for summarization. Proposed to summarize documents from the perspective of data reconstruction, and select sentences that can best reconstruct the original documents. In modeled documents (hotel reviews) as multi-attribute uncertain data and optimized a probabilistic coverage problem of the summary [3]. There have also been studies on summarizing microblogs for some specific types of events, e.g., sports events. proposed to identify the participants of events, and generate summaries based on sub-events detected from each participant introduced a solution by learning the underlying hidden state representation of the event, which needs to learn from previous events (football games) with similar structure. In summarized events by exploiting "good reporters", depending on event-specific keywords which need to be given in advance. In contrast, we aim to deal with general topic-relevant tweet streams without such prior knowledge [4]. Moreover, their method stores all the tweets in each segment and selects a single tweet as the summary, while our method maintains distilled information in TCVs to reduce storage/computation cost, and generates multiple tweet summaries in terms of content coverage and novelty. In addition to online summarization, our method also supports historical summarization by maintaining TCV snapshots.

Timeline Detection

The demand for analyzing massive contents in social media 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).

Topics (Filtering keywords)	#Tweets	Time span
Obama	95,055	2014.07-2014.10
Chelsea	433,884	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

Table1: Basic information of dataset.

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), we can expect a change of sub-topic (i.e., a time node on the timeline). To quantify the variation, we use the divergence to measure the distance between two word distributions in two successive summaries Sc and Sp (Sc is the distribution of the current summary and Sp is that of the previous one).

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.

Experiments for Summarization

Overall performance comparison

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 (Figure 2). In our experiments, we find similar trends in the comparison of precision, recall and F-score between the proposed approach and the baseline methods [5,6].

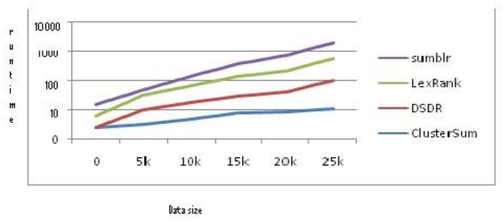

Figure 2: Scalability on data size.

Conclusion

We proposed a prototype called Sumblr which supported continuous tweet stream summarization [7-9]. 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 [10]. 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. Aggarwal CC, Han J, Wang J, Yu PS (2003) A framework for clustering evolving data streams. Proceedings of the 29th VLDB Conference, Germany.
2. Zhang T, Ramakrishnan R, Livny M (1996) BIRCH: An efficient data clustering method for very large databases. Acm sigmod International Conference Manage, Canada.
3. Zhang J, Ghahramani Z, Yang Y (2004) A probabilistic model for online document clustering with application to novelty detection. Adv Neural Inf Process Syst 1617-1624.
4. He Q, Chang K, Lim EP, Zhang J (2007) Bursty feature representation for clustering text streams. 2007 SIAM international conference on data mining, Singapore.
5. Bradley PS, Fayyad UM, Reina C (1998) Scaling clustering algorithms to large databases. Knoledge Discovery Data -98, USA.

6. Gong L, Zeng J, Zhang S (2011) Text stream clustering algorithm based on adaptive feature selection. Expert Syst Appl 38: 1393-1399.

7. Zhong S (2005) Efficient streaming text clustering. Neural Netw 18: 790-798.

8. Aggarwal CC, Yu PS (2010) On clustering massive text and categorical data streams. Knowl Inf Syst 24: 171-196.

9. Barzilay R, Elhadad M (1997) Using lexical chains for text summarization. Advances in automatic text summarization. MIT press Cambridge, London.

10. Yih WT, Goodman J, Vanderwende L, Suzuki H (2007) Multidocument summarization by maximizing informative contentwords. Int Joint Conf Artif Intell 7: 1776-1782.