International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2015): 78.96 | Impact Factor (2015): 6.391

K Prototype Clustering with Efficient Summarization for Topic Evolutionary Tweet Stream Clustering

Swapnil Rajaram Ahire[1], Lakshita Landge[2]

[1, 2] Astral Institute of Technology and Research, RGPV University, Indore, Madya Pradesh, India

Abstract: ***Social media data reflects the lot of interests of virtual communities in a spontaneous and timely manner. Tweets are created as short text message and shared for each users and information analysts. Twitter that receives over four hundred million tweets per day has emerged as a useful supply of reports, blogs, opinions and additional. Our projected work consists of 3 parts. Tweet stream clustering to cluster tweets with k-prototype cluster algorithm (Existing paper shows, k-means clustering algorithm wont to produce the initial clusters. With globular clusters, it did not work well. Therefore in our projected work, we are using k-prototype algorithm to get tighter clusters than k-means algorithm, particularly if the clusters area globular). Second is tweet cluster vector technique to get rank summarization with greedy algorithm. Third to notice and monitors the outline – summary based and volume based variation to provide timeline from tweet stream. Implementing continuous tweet stream reducing a text document is but not an easy task. Since a large variety of tweets area trashy, unrelated and raucous in nature, owing to the social nature of tweeting. Further, tweets area powerfully related with their announce instance and current tweets tend to reach an awfully quick rate. Efficiency - tweet streams are huge, thus the summarization algorithm ought to be greatly capable and time efficient. Flexibility - it ought to offer tweet summaries of random moment durations. Topic evolution - it ought to habitually notice sub - topic changes and therefore the moments that they happen.***

Keywords: Tweet stream, summarization, timeline, summary, continuous tweet stream

1. Introduction

Twitter, Weibo, Tumblr such micro-blogging services has resulted in the huge amount of short-text messages. Twitter, which receives over millions tweets per day has emerged as an invaluable source of blogs, news, opinions, and more. Tweets, in their raw form, while more informative. For an instance, search for a hot topic in Twitter may yield millions of tweet. Even if filtering is allowed, so many tweets for important contents are available. To make things worse, new tweets satisfying the filtering criteria may arrive very fast, continuously, at an unpredictable rate. One solution to information overloading problem is summarization. The summarization is restating of the main ideas of the text in as few words as possible. A good summary should cover the main topics also the subtopics and have diversity among the sentences to reduce redundancy. Summarization is widely used, especially when users surf the internet with their mobile devices which have much smaller screens than PCs. The Traditional document summarization approaches are not as effective in the situation of tweets like the big size of tweets and the fast and continuous nature of their arrival.

In this project, we propose continuous tweet summarization as a solution to address this problem. Traditional documents focus on static and small-scale data. Here we aim to deal with dynamic, tweet streams and produce novel tweet streams. We propose a novel prototype called Sumblr (SUMmarization By stream cLusteRing) for tweet streams. We first use an online tweet stream clustering algorithm to cluster tweets and maintain distilled statistics called Tweet Cluster Vectors. In existing base paper, k-means clustering algorithm was used to create the initial clusters but with global cluster, it didn't work well. In our proposed work, we use k-prototype clustering to produce tighter clusters than k-means clustering, especially if the clusters are globular. Then the TCV-Rank summarization technique for generating online summaries and historical summaries of arbitrary time durations. Finally, described a topic evolution detection method, which consumes online and historical summaries to produce timelines automatically from tweet streams [1].

2. Literature Review

Television broadcasters are commencing to mix social micro-blogging systems like Twitter with television to make social video experiences around events. Here checked out one such event, the primary U.S. presidential discussion in 2008, in conjunction with aggregate ratings of message sentiment from Twitter. Then start to develop an analytical methodology and visual representations that would facilitate a journalist or public affairs person higher perceive the temporal dynamics of sentiment in reaction to the controversy video. The demonstration of visuals and metrics which will be wont to find sentiment pulse, anomalies in this pulse and indications of debatable topics. which will not inform the planning of visual analytic systems for social media events[2].

Classic news summarization plays a very important role with the exponential document growth on the net. Several approaches are proposed to get summaries however rarely at the same time consider evolutionary characteristics of news and to traditional summary. Therefore, shown a unique framework for the web mining problem named evolutionary Timeline summarization (ETS). Given the huge collection of time-stamped web documents associated with a general news query. ETS aims to come back the evolution trajectory with the timeline, consisting of individual however correlate summaries of each date, relevancy, coverage, coherence and

cross-date diversity. ETS greatly facilitates quick news browsing and knowledge comprehension and therefore could be a necessity. They formulate the task as an improvement problem via repetitive substitution from a collection of sentences to a subset of sentences that satisfies the above necessities, balancing coherence/diversity measuring and local/global summary quality. The optimized substitution is iteratively conducted by incorporating many constraints till convergence. Here newly developed experimental systems to gauge on six instinctively totally different datasets which has amounted to 10251 documents [3].

In this paper, study of data stream cluster problem within the context of text and categorical data domains is shown. Whereas the cluster problem has been studied recently for numeric data streams. The issues of text and categorical data present completely different challenges because of the big and un-ordered nature of the corresponding attributes. Therefore, proposed an algorithms for text and categorical data stream clustering. An approach for stream clustering that summarizes the stream into variety of fine grained cluster droplets. These summarized droplets are utilized with a variety of user queries to form the clusters for various input parameters. Thus, this provides an online analytical process approach to stream clustering. [4].

Here investigated one technique to provide a summary of an original text without requiring its full linguistics interpretation, however instead relying on a model of the topic progression within the text derived from lexical chains. Also present a new algorithm to form lexical chains during a text, merging many robust data sources, the WordNet thesaurus, shallow parser, a part-of-speech tagger for the identification of nominal groups, and a segmentation algorithm. Summarization occurs in four steps: the first text is segmental, lexical chains are made, strong chains are identified and important sentences are extracted. During this paper presented empirical results on the identification of robust chains and of significant sentences [5].

In this paper show that an easy procedure supported increasing the number of informative content-words can produce a number of the best reportable results for multi-document summarization. Initially assign a score to every term within the document cluster, using only frequency and position information and then discover the set of sentences within the document cluster that maximizes the total of those scores, subject to length constraints. The overall results are the best reportable on the DUC-2004 summarization task for the ROUGE-1 score and are the most effective, but not statistically significantly different from the best system in MSE-2005[6].

We introduce a random graph-based methodology to compute relative importance of textual units for natural language process. We test the technique on the problem of Text summarization (TS). Extractive TS depends on the concept of sentence saliency to identify the foremost important sentences in an exceedingly document or set of documents. Saliency is usually defined in terms of the presence of specific important words or in terms of similarity to a centroid pseudo-sentence [10].

We consider a new approach, LexRank, which is used for computing sentence importance supported the conception of eigenvector centrality in a graph illustration of sentences. During this model, a connectivity matrix based on intra-sentence cosine similarity is employed as the adjacency matrix of the graph illustration of sentences. Our system, based on LexRank ranked in 1st place in more than one task within the recent DUC 2004 analysis. in this paper we present a close analysis of our approach and apply it to a larger data set as well as data from earlier DUC evaluations. We discuss many methods to compute centrality using the similarity graph. The result shows that the degree-based methods (including LexRank) outperform each centroid-based methods and different systems taking part in DUC in most of the cases. Furthermore, the LexRank with threshold methodology outperforms the other degree-based techniques together with continuous LexRank.[7]

3. Problem Definition

The implementation of continuous tweet stream summarization is not an simple task, since a large variety of tweets are not important, irrelevant and noisy in nature, due to the social nature of tweeting. Further, tweets are fully related with their posted time. New tweets tend to arrive at a very quick rate. Consequently, a good solution for continuous summarization needs to address the subsequent 3 issues:

1) Efficiency —tweet streams are forever very large in scale, hence the summarization algorithm need to be extremely efficient.
2) Flexibility —it should give tweet summaries of arbitrary time durations.
3) (3)Topic evolution — it automatically finds such sub-topic changes and the moments that they happen. But, the existing methods cannot satisfy the above 3 requirements because:

- They highly focus on static and small-sized data sets, and hence are not that much efficient and scalable for huge data sets and data streams.
- For providing summaries of discretional durations, they will need to perform iterative/recursive summarization for each possible time length, which is unacceptable.
- Summary results are insensitive to time. Thus it is tough for them to discover topic evolution.

In this project, we introduce a novel summarization framework called as Sumblr (continuous SUMmarization By stream cLusteRing) with k-prototype clustering. The framework consists of three main elements, namely the Tweet Stream clustering module, High-level Summarization module and the Timeline Generation module for deal with dynamic, fast incoming, and large-scale tweet streams [10].

4. Proposed Methodology

Our framework consists of three main modules: the tweet stream clustering module, the high-level summarization module and the timeline generation module.

4.1 Tweet Stream Clustering

The tweet stream clustering module maintains the on-line statistical data. Given a topic - based tweet stream able to efficiently cluster the tweets and maintain compact cluster information a scalable cluster framework that by selection stores necessary portions of the data, and compresses or discards other parts. CluStream is one of the foremost classic streams clustering ways. It consists of an on-line micro-clustering element and an offline macro - clustering element. A variety of services on the online like news filtering, text crawling, and topic detecting etc. have posed needs for text stream clustering CluStream to generate duration - based cluster results for text and categorical data streams. However, this algorithm depends on an on-line phase to generate a large range of micro - clusters and an offline section to re - cluster them. In contrast, our tweet stream cluster algorithm is an on-line procedure with no extra offline clustering. And in the context of tweet summarization, we adapt the on-line cluster phase by incorporating the new structure TCV, restricting the variety of clusters to ensure efficiency and therefore the quality of TCVs.

1) Tweet Stream Initialization
2) Incremental cluster
3) Deleting Outdated Clusters
4) Merging Clusters
Tweet Stream initialization

4.1.1 Tweet Stream Initialization

At the beginning of the stream, we collect a small range of tweets and use a k-prototype clustering algorithm (instead of k-means) to form the initial clusters. Next, the stream clustering process starts to update the TCVs incrementally 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 available. The key problem is to make a decision whether to attract into one of the current in progress clusters or make t as a new cluster. We initially find the cluster whose centroid is the closest to t. Specifically, we get the centroid of every cluster, compute its cosine similarity to t, and find out the cluster Cp with the largest similarity.

4.1.3 Deleting outdated Clusters

For most events (such as news, football matches and concerts) in tweet streams, timeliness is very important because they usually don't last for a long time. thus it is safe to delete the clusters representing these sub - topics once they are rarely discussed to search out out such clusters, an intuitive approach 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 each cluster can increase memory costs, particularly when clusters grow big. Thus, we use an approximate methodology to get Avgp.

4.1.4 Merging Clusters

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

4.2 High - Level Summarization

The module provides two kinds of summaries: on-line and historical summaries. An online summary describes what is presently discussed among the public. Thus, the input for generating on-line summaries is retrieved directly from the current clusters maintained in memory. On the other hand, a historical summary helps people to understand the main happenings in a specific period, which means to eliminate the influence of tweet contents from the outside of the period. As a result, retrieval of the required information for generating historical summaries is very much complicated, and this shall be our focus within the following discussion. Suppose the length of a user - defined time period is H, and the ending timestamp of the period is tse.

4.2.1 Document/Microblog Summarization

The Document summarization is categorized as extractive and abstractive. The previous selects sentences from the documents, whereas the latter may generate phrases and sentences that don't appear in the original documents. Now we are focusing extractive summarization. Extractive document summarization has received lots of recent attention. Most of assign salient scores to sentences of the documents, and select the top - ranked sentences. Some works attempt to extract summaries without such salient scores. The symmetric non - negative matrix factorization for cluster sentences and select sentences in each cluster for summarization. Projected to summarize documents for the perspective of data reconstruction, and choose sentences that can best reconstruct the original documents. In modeled documents (restaurant reviews) as multi - attribute uncertain data problem and optimized a probabilistic coverage of the summary. There have also been studies on microblogs summarizing for some specific forms of events, e.g., sports events are taken to identify the participants of events and generate summaries based on sub - events detected from each participant introduced by learning the underlying hidden state representation of the event, which has to learn from previous events (soccer game) with similar structure. As in the summarized events by exploiting "good reporters", depends on event - specific keywords which requires being in advance. In distinction, we aim to deal with general topic - relevant tweet streams without such previous knowledge. Moreover, their method stores all the tweets in every segment and selects one tweet as the summary, whereas our method

maintains distilled information in TCVs to reduce storage/computation cost, and generates many tweet summaries in terms of content coverage and novelty additionally to online summarization, our method supports historical summarization by maintaining TCV snapshots.

4.3 Timeline Detection

The demand for analyzing huge contents in social medias fuels the developments in visualization techniques. Timeline is one of these techniques which may make analysis tasks easier and faster. We have presented a timeline supported backchannel for conversations around events and proposed the evolutionary timeline summarization (ETS) to compute evolution timelines same to ours, that consists of a series of time stamped summaries. By a predefined timestamp set the dates of summaries are determined. In contrast, our methodology discovers the changing dates and generates timelines dynamically throughout the process of continuous summarization. Moreover, ETS does not focus on efficiency and scalability problems, which are important in our streaming context. Several systems detect important moments when fast increases or "spikes" in status update volume occurs. Developed an algorithm based on tcp congestion detection, employed a slope based method to find out spikes. After that, tweets from every moment are identified, and word clouds or summaries are selected which is different from this two step approach. Our method detects topic evolution and produces summaries/timelines in an online fashion.

5. System Architecture

- In this paper, we introduce a novel summarization framework called Sumblr (continuouSsUMmarizationBy stream cLusteRing).
- The framework consists of three main components, specifically the Tweet Stream clustering module, the High-level summarization module and Timeline Generation module.
- We design an efficient tweet stream cluster algorithm in the tweet stream clustering module. It is an online algorithm allows effective clustering of tweets in only one pass over the data. At the beginning of the stream, we use k-prototype clustering that produce tighter clusters than k-means clustering, especially if the clusters are globular.
- The high-level summarization module supports generation of two forms of summaries: on-line and historical summaries.
- A topic evolution detection algorithm is base of the timeline generation module that consumes online and historical summaries to produce real-time and range timelines. The algorithm monitors quantified variation during the course of stream process.
- We load the Twitter data sets. as a result of tweets are being created and shared at an unprecedented rate.
- Tweets, in their raw form, whereas being informative also can be overwhelming.
- Through millions of tweets that contain huge amount of noise and redundancy.

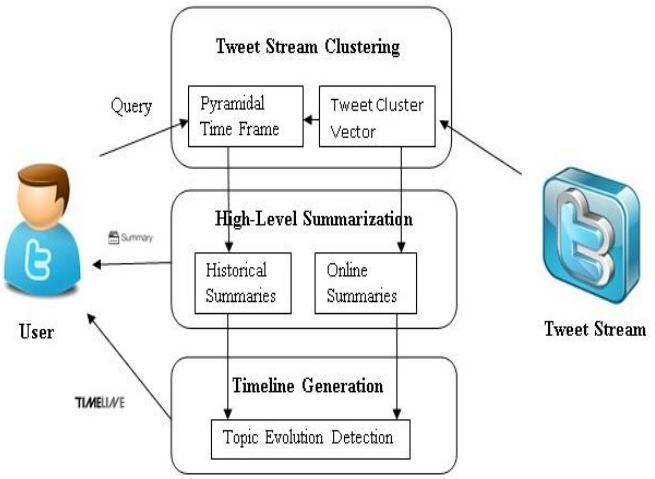

Fig 1. System Architecture[1]

- In this project, we propose a novel continuous summarization framework called as Sumblr to all eviate the problem. Therefore we load the dataset for continuous summarization and timeline generation.

5.1 Tweet Stream Clustering

This module maintains the online statistical data. With a given topic - based tweet stream, it is ready to efficiently cluster the tweets and maintain compact cluster information a scalable cluster framework which selectively stores necessary parts of the data, and compresses or discards different portions.

It consists of four phases like
1) Tweet Stream initialization
2) Incremental cluster
3) Deleting outdated Clusters
4) Merging Clusters

5.2 High-Level Summarization

The summarization module provides two types of summaries: historical and online summaries.

- The summarization module provides two types of summaries: historical and online summaries.
- A historical summary helps people to understand the main happening during a specific period, means from the outside of that period we need to eliminate the influence of tweet contents.
- On the other hand, an online summary describes what is currently discussed among the public. The input for generating online summaries is retrieved directly from the current clusters maintained in memory.
- As a result, retrieval of the needed information for generating the historical summaries is more complicated, and this is the focus in this discussion. Suppose ending timestamp of the duration is tse and the length of a user - defined time duration is H.

5.3 Timeline Detection

- The increasing demand for analyzing huge contents in social media fuels the developments in visualization techniques. Timeline is one among these techniques can make analysis tasks easier and faster.
- It has been presented a timeline - based mostly backchannel for conversations around events. It proposed the evolutionary timeline summarization (ETS) to compute evolution timelines similar to ours that consists of a series of time - stamped summaries.
- The pre - defined timestamp set are used to determine the dates of summaries. In distinction, our methodology discovers the changing dates and generates timelines dynamically throughout the process of continuous summarization. Moreover, ETS does not specialize in efficiency and scalability problems that are important in our streaming context.
- Several systems find important moments when fast increases or "spikes" in status update volume happen. Developed an algorithm based on tcp congestion detection, utilized a slope - based methodology to find spikes.
- After that, tweets from each moment are known, and word clouds or summaries are selected completely different from this two - step approach, our methodologies detect topic evolution and generate summaries/timelines in an online fashion.

6. Algorithms

6.1 K-Prototype Clustering:

1) Select k number of initial prototypes from a data set X and assign one for each cluster.
2) Allocate every object in X to a cluster which has nearest prototype. After each allocation update the prototype of the cluster.
3) After all the objects have been allocated to a cluster, test again the similarity of objects for the current prototype. If any of the objects found such that its nearest prototype belongs to the other cluster than its current cluster, then reallocate the object to the same cluster and update prototypes for both of the clusters.
4) Repeat step 3) yet no object change clusters after testing full cycle of X.

6.2 Incremental Tweet Stream Clustering

Input: a cluster set C set
Output: store C set in PTF
1) While! stream.end () do
2) Tweet t=stream. Next ();
3) Choose Cp in C set whose centre is the
Closest to t;
4) If MaxSim (t) < MBS then
5) Create a new cluster Cnew=ftg;
6) Cset.add (Cnew);
7) Else
8) update Cp with t;
9) If TScurrent %(ai) ==0 then
10) Store C set into PTF;

6.3 TCV-Rank Summarization

Input: a cluster set D(c)
Output: a summary set S
1) S=0, T=fall the tweets in ft set of D(c)g;
2) build a similarity graph on T;
3) Compute LexRank score LR;
4) Tc =ftweets with the highest LR in each cluster g;
5) While jSj<L do
6) For each tweet ti inTc□S do
7) calculate vi according to Equation (2);
8) select tmax with the highest vi ;
9) S.add (tmax);
10) While jSj< L do
11) For each tweet i in Tc -S do
12) Calculate v' i according to Equation (2);
13) select tmax with the highest v' i ;
14) S.add (t'max);
15) Return S;

6.4 Topic Evolution Detection

Input: a tweet stream binned by time units
Output: a timeline node set TN
1) TN=0;
2) While! stream.end () do
3) Bin Ci=stream. Next ();
4) If hasLargeVariation ( ) then
5) TN.add (i);
6) return TN;

7. Result and Analysis

Input:
1) We choose a twitter dataset, timelines for sport topics are relatively easier to build. The reference timelines can be manually produced.
2) We search the particular topic i.e. query, which is entered by user in twitter dataset.

Output:
1) Give the summarization of respective input query.

System Boundaries:
Deleting the old data which is rarely visited or followed by the twitter and the average timestamp of the latest 10 percent tweets is more than three days old, are considered as outdated and removed.

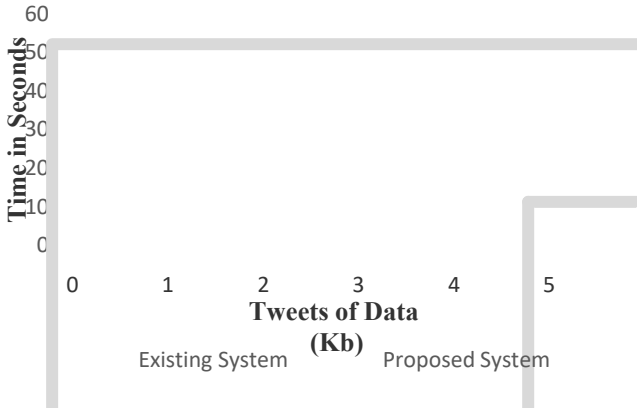

Figure 2: Scalability of data with time

For evaluating the performance of proposed system, we consider here the time required to produce the summarization of tweets & tweets of data size. For existing system, producing the summarization on tweets of data size 5Kb require 60 secs. The recommendation proposed system for producing the summarization on tweets of data size 5Kb is requires 20 secs.

As the experimental result shows optimum performance of proposed system than that of existing system. The proposed system is protocol oriented and more time efficient, scalable than existing system.

K Prototype clustering has more stability of clusters for tweets in it as compared to K means algorithm. Stability of K Means is reduced because, at the start it selects a random tweet in a cluster and make it as centroid.

8. Conclusion

In this paper, we proposed a tweet stream summarization framework, called Sumblr, to generate summaries and timelines in the context of streams. Sumblr employs a tweet stream clustering algorithm for the purpose of compressing tweets into TCVs and also maintains them in an online fashion. Our proposed k-prototype clustering algorithm produced tighter clusters than k- means clustering, if the clusters are globular. We designed a novel data structure called TCV for stream processing, and proposed the TCV-Rank algorithm for online and historical summarization. The topic evolution can be produced automatically, allowing Sumblr to produce dynamic timelines for tweet streams. The proposed K-prototype clustering algorithm can produce tighter clusters and time efficiently than K-Means clustering algorithm.

9. Future Work

Future work is to develop a multi-topic version of summarization in a distributed system, and evaluate same on more complete and large-scale data sets.

References

[1] Zhenhua Wang, Lidan Shou, Ke Chen, Gang Chen, and Sharad Mehrotra, “On Summarization and Timeline Generation for Evolutionary Tweet Streams,” IEEE Transactions On Knowledge And Data Engineering, vol. 27, no. 5, May 2015 pp. 1301-1315.
[2] N. A. Diakopoulos and D. A. Shamma, “Characterizing debate performance via aggregated twitter sentiment,” in Proc. SIGCHI Conf. Human Factors Comput. Syst., 2010, pp. 1195–1198.
[3] R. Yan, X. Wan, J. Otterbacher, L. Kong, X. Li, and Y. Zhang, “Evolutionary timeline summarization: A balanced optimization framework via iterative substitution,” in Proc. 34th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval, 2011, pp. 745–754.
[4] C. C. Aggarwal and P. S. Yu, “On clustering massive text and cate-gorical data streams,” Knowl. Inf. Syst., vol. 24, no. 2, pp. 171–196, 2010.
[5] R. Barzilay and M. Elhadad, “Using lexical chains for text summa-rization,” in Proc. ACL Workshop Intell. Scalable Text Summarization , 1997, pp. 10–17.
[6] W.-T. Yih, J. Goodman, L. Vanderwende, and H. Suzuki, “Multi-document summarization by maximizing informative content-words,” in Proc. 20th Int. Joint Conf. Artif. Intell., 2007, pp. 1776–1782.
[7] G. Erkan and D. R. Radev, “LexRank: Graph-based lexical central-ity as salience in text summarization,” J. Artif. Int. Res., vol. 22, no. 1, pp. 457–479, 2004.