International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 03 Issue 18
December 2016

Summarization and Generation of Air Time on the Evolutionary Tweet

[1]P ARJUN [2]B SUMAN

[1]PG SCHOLAR, DEPARTMENT OF CSE, VIJAY KRISHNA INSTITUTE OF TECHNOLOGY,PALAMAKULA,SHAMSHABAD,R.R. DIST, HYDERABAD,TELANGANA-501 325

[2] ASSOCIATE PROFESSOR, DEPARTMENT OF CSE, VIJAY KRISHNA INSTITUTE OF TECHNOLOGY,PALAMAKULA,SHAMSHABAD,R.R. DIST, HYDERABAD,TELANGANA-501 325

Abstract— *Short-text messages such as tweets are being created and shared at an unprecedented rate. Tweets, in their raw form, while being informative, can also be overwhelming. For both end-users and data analysts, it is a nightmare to plow through millions of tweets which contain enormous amount of noise and redundancy. In this paper, we propose a novel continuous summarization framework called Sumblr to alleviate the problem. In contrast to the traditional document summarization methods which focus on static and small-scale data set, Sumblr is designed to deal with dynamic, fast arriving, and large-scale tweet streams. Our proposed framework consists of three major components. First, we propose an online tweet stream clustering algorithm to cluster tweets and maintain distilled statistics in a data structure called tweet cluster vector (TCV). Second, we develop a TCV-Rank summarization technique for generating online summaries and historical summaries of arbitrary time durations. Third, we design an effective topic evolution detection method, which monitors summary-based/volume-based variations to produce timelines automatically from tweet streams. Our experiments on large-scale real tweets demonstrate the efficiency and effectiveness of our framework.*

1INTRODUCTION

Increasing popularity 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 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 a set of documents by a summary consisting of several sentences. Intuitively, a good summary should cover the main topics (or subtopics) and have diversity among the sentences to reduce redundancy. Summarization is extensively used in content presentation, specially when users surf the internet with their mobile devices which have much smaller screens than PCs. Traditional document summarization approaches, however, are not as effective in the context of tweets given both thelarge volume 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 thet emporal feature of the arriving tweets. Let us illustrate the desired properties of a tweet summarization system using an illustrative example of a usage of such a system. 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. As illustrated in Fig. 1, a user may explore tweets based on a timeline (e.g., "Apple" tweets posted between October 22nd, 2012 to November 11th, 2012). Given a timeline range, the summarization system may produce a sequence of timestamped 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 8 am to 11 pm on November 5th). The system may provide a drill-down summary of the duration that enables the user to get additional details for that duration. A user,

perusing a drill-down summary, may alternatively zoom out to a coarser range (e.g., October 21st to October 30th) to obtain a roll-up summary of tweets. To be able to support such drill-down and roll-up operations, the summarization system must support the following two queries: summaries of arbitrary time durations and real-time/range timelines. 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. To this end, in this paper, we propose a new summarization method, continuous summarization, for tweet streams.

2 RELATED WORK

In this section, we review the related work including stream data clustering, document/microblog summarization, timeline detection, and other microblog mining tasks.

2.1 Stream Data Clustering Stream data

clustering has been widely studied in the literature. BIRCH clusters the data based on an in-memory structure called CF-tree instead of the original large data set. Bradley et alproposed 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. The pyramidal time frame was also proposed in to recall historical microclusters for different time durations. A variety of services on the Web such as news filtering, text crawling, and topic detecting etc. have posed requirements for text stream clustering. A few algorithms have been proposed to tackle the problem Most of these techniques adopt partition-based approaches to enable online clustering of stream data. As a consequence, these techniques fail to provide effective analysis on clusters formed over different time durations.

2.2 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. Wang et al. used the symmetric non-negative matrix factorization to cluster sentences and choose sentences in each cluster for summarization. He et al. proposed to summarize documents from the perspective of data reconstruction, and select sentences that can best reconstruct the original documents. Xu et al. modeled documents (hotel reviews) as multi-attribute

uncertain data and optimized a probabilistic coverage problem of the summary. While document summarization has been studied for years, microblog summarization is still in its infancy. Sharifi et al. proposed the Phrase Reinforcement algorithm to summarize tweet posts using a single

2.3 Timeline Detection

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. Diakopoulos and Shamma made early efforts in this area, using timelines to explore the 2008 Presidential Debates by Twitter sentiment. Dork et al. presented a timeline-based backchannel for conversations around events. In Yan et al. proposed the evolutionary timeline summarization (ETS) to compute evolution timelines similar to ours, which consists of a series of time-stamped summaries. However, in 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. TwitInfo developed an algorithm based on TCP congestion detection, while Nichols et al. 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.

2.4 Other Microblog Mining Tasks

The emergence of microblogs has engendered researches on many other mining tasks, including topic modeling [27], storyline generation [28] and event exploration [25]. Most of these researches focus on static data sets instead of data streams. For twitter stream analysis, Yang et al. [29] studied frequent pattern mining and compression. In [30], Van Durme aimed at discourse participants classification and used gender prediction as the example task, which is also a different problem from ours. To sum up, in this work, we propose a new problem called continuous tweet summarization. Different from previous studies, we aim to summarize large-scale and evolutionary tweet streams, producing summaries and timelines in an online fashion.

3 PRELIMINARIES

In this section, we first present a data model for tweets, then introduce two important data structures: the tweet cluster vector and the 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, we build a matrix based on social relationships among users, and compute the UserRank as in [31]. As a result, we define a tweet ti as a tuple: ðtvi;ts i;w iÞ, where tvi is the textual vector, tsi is the posted timestamp and wi is the UserRank 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. In this section, we propose 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: TCVðCÞ¼ ðsum v;wsum v;ts1;ts2;n;ftsetÞ, where □ sum v ¼Pn i¼1

tvi=jjtvijj is the sum of normalized textual vectors, □ wsum v ¼Pn i¼1 wi □tvi is the sum of weighted textual vectors, □ ts1 ¼Pn i¼1 tsi is the sum of timestamps, □ ts2 ¼Pn i¼1ðtsiÞ2 is the quadratic sum of timestamps, □ n is the number of tweets in the cluster, and □ 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. In fact, we only store 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, we use cosine similarity as the distance metric. From the definition, we can derive the vector of cluster centroid (denoted as cv) cv ¼ X n i¼1 wi □tvi ! □n ¼ wsum v=n: (1) The definition of TCV is an extension of the cluster feature vector proposed in [2]. Besides information of data points (textual vectors), TCV includes temporal information and representative original tweets. As in [2], our TCV structure can also be updated in an incremental manner when new tweets arrive. We shall discuss details on TCV updating in Section 4.1.2.

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 [1]:

4 THE SUMBLR FRAMEWORK

our framework consists of three main modules: the tweet stream clustering module, the high-level summarization module and the timeline generation module. In this section, we shall present each of them 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, 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.

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. 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 (denoted as MaxSimðtÞ). Note that although Cp is the closest to t, it does not mean t naturally belongs to Cp. The reason is that t may still be very distant from Cp. In such case, a new cluster should be created . The decision of whether to create a new cluster can be made with the following heuristic.

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 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 a user-defined time duration is H, and the ending timestamp of the duration is tse.

.4.2.1 TCV-Rank Summarization

Algorithm Given an input cluster set, we denote its corresponding TCV set as DðcÞ. A tweet set T consists of all the tweets in the ft sets inDðcÞ. 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 F¼f T1;T2;...;T tg as a

collection of non-empty subsets ofT , where a subset Ti represents a sub-topic and jTij means the number of its related tweets. Suppose for each Ti, there is a tweet which represents the content of Ti's sub-topic. Then, selecting k tweets is equivalent to selecting k subsets. Now, the problem can be defined as: given a number k and a collection of sets F, find a subset F0 □F, such that jF0j¼k and jSTi2F0Tij is maximized (i.e., F0 contains as many tweets as possible). We note that this is the Max-kCover problem, which is NP-hard. Thus, our summarization problem is also NP-hard. From the geometric interpretation, our summarization tends to select tweets that span the intrinsic subspace of candidate tweet space, such that it can cover most information of the whole tweet set.

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. We shall only describe the real-time case here. 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. We 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, we loop through the bins and append new timeline nodes whenever large variations are detected (lines 4-5).

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), we can expect a change of sub-topic (i.e., a time node on the timeline). To quantify the variation, we use the Jensen-Shannon 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) DJSðSc;SpÞ¼ 1 2ðDKLðScjjMÞþDKLðSpjjMÞÞ: (3) M is

the average of the two word distributions, i.e., M ¼ 1 2ðSc þSpÞ. DKL is the Kullback-Leibler divergence (KLD) which defines the divergence of distribution M from S DKLðSjjMÞ¼X w2V pðwjSÞlog pðwjSÞ pðwjMÞ : (4) We apply DJSðSc;SpÞ to measure variation instead of directly using DKLðScjjSpÞ, because the JSD is a symmetrical and smoothed (the use of pðwjMÞ avoids zero values of the denominator in Equation (4)) version of the KLD. For ease of comparison, we use the base 2 logarithm in Equation (4), so that the JSD is bounded by½0;1 [34]. According to this summary-based variation, we determine whether the current time is a sub-topic changing node by Dcur Davg > ts; (5) where Dcur is the distance between the current summary and its previous neighboring summary, Davg is the average distance of all the previous successive summary pairs which do not produce time nodes, and ts (ts 1) is the decision threshold. That is, we detect sub-topic changes whenever there is a burst in distances between successive summaries.

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.

5 EXPERIMENTS

In this section, we evaluate the performance of Sumblr. We present the experiments for summarization and timeline generation respectively.

5.1 Experiments for Summarization

5.1.1 Setup Data Sets.

We construct five data sets to evaluate summarization. One is obtained by conducting keyword filtering on a large Twitter data set used by . The other four include tweets acquired during one month in 2012 via Twitter's keyword tracking API.4 As we do not have access to the respective users' social networks for these four, we set their weights of tweets wi to the default value of 1. Details of the data sets are listed in Table 2. Ground truth for summaries. As no previous work has conducted similar study on continuous summarization, we

have to build our own ground truth (reference summaries). However, manual creation of these summaries is apparently impractical due to the large size of the data sets. Thus, we employ a two-step method to obtain fair-quality reference summaries: 1) Given a time duration, we 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 first builds a sentence similarity graph, and then selects important sentences based on the concept of eigenvector 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.

5.1.2 Overall Performance Comparison

In this section, we compare the F-scores and runtime costs between Sumblr and three baseline algorithms (sliding window version). As tweets are often produced very quickly and reach a huge volume in a short while, it is hardly meaningful to summarize a small number of tweets. Thus the window size should be a relatively large one. In this experiment, we set window size to 20,000 and sampling interval to 2,000. The step size varies from 4,000 to 20,000. The metrics are averaged over the whole stream. Figs. 6 and 7 present the results for different step sizes.

5.1.3 Parameter Tuning

In this section, we tune the parameters in our approach. In each of the following experiments, we vary one parameter and keep the others fixed. Effect of b. In Heuristic 1 we use b to determine whether to create a new cluster. Figs. 9a and 9b show its effect on summary quality and efficiency. When b is small, tweets related to different sub-topics may be absorbed into the same clusters, so the input of our summarization component is of low quality. At the same time, there are many focus tweets in each cluster, thus the time cost of cluster updating and summarization is high. When b increases, too many clusters are created, causing damage to both quality and efficiency. A good choice is b ¼ 0:07 as it gives more balanced results. Effect of Nmax. Figs. 9c and 9d depict the performance of Nmax. For small Nmaxs,

many merging operations are conducted, which are time-consuming and produce lots of low-quality clusters. For large values, stream clustering is slow due to large number of clusters. Note that the storage overhead (both in memory and disk) is also higher for larger Nmaxs. A balanced value for Nmax is 150. Effect of mc. Another parameter in cluster merging is mc (0 < mc <1). It does not have significant impact on efficiency, so we only present its quality results (Fig. 9e). Small values of mc result in low-quality clusters, while large ones lead to many merging operations, which in turn reduce the quality of clusters. An ideal value for mc is 0:7.

6 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 demonstrate the efficiency and effectiveness of our method. For future work, we aim to develop a multi-topic version of Sumblr in a distributed system, and evaluate 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. Very Large 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," inProc.Adv.NeuralInf.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,"inProc.20thInt.JointConf.Arti f.Intell.,2007,pp.1776–1782.

[11] G. Erkan and D. R. Radev, "LexRank: Graph-based lexical centrality as salience in text summarization," J. Artif. Int. Res., vol. 22, no. 1, pp. 457–479, 2004.

[12] D. Wang, T. Li, S. Zhu, and C. Ding, "Multi-document summarization via sentence-level semantic analysis and symmetric matrix factorization," in Proc. 31st Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval, 2008, pp. 307–314

.