International Journal of Current Research and Modern Education (IJCRME)
ISSN (Online): 2455 – 5428 & Impact Factor: 3.165
Special Issue, NCFTCCPS - 2016

TWEET SUMMARIZATION IN REAL TIME EVENT ANALYSIS

K. Senthami*, S. Sinduja, D. Vanitha*** & C. Arivazhagan******
Department of Computer Science & Engineering, Jayam Engineering College, Dharmapuri, Tamilnadu

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.

Index Terms: Tweet Stream, Continuous Summarization, Summary & Timeline

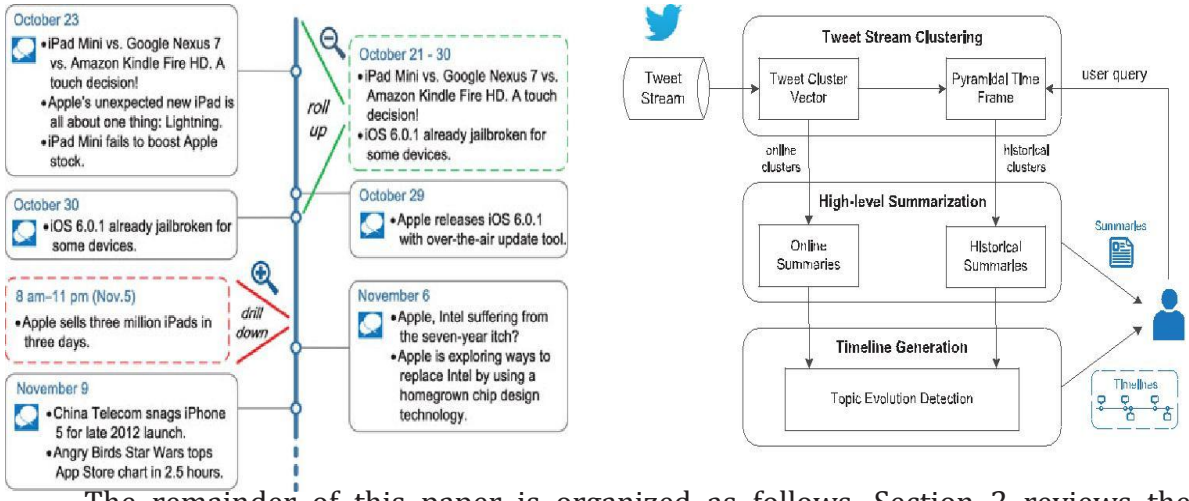
1. Introduction:

Increasing popularity 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 con-tenuously, 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 sub-topics) and have diversity among the sentences to reduce redundancy. Summarization is extensively used in content presentation, especially 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 the large 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 the temporal 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 time-line range, the summarization system may produce a sequence of time stamped 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 pro-vide 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.



The remainder of this paper is organized as follows. Section 2 reviews the related work. Section 3 explains concepts including TCV and PTF. Section 4 describes our Sumblr framework in detail. The experimental settings and results are presented in Section 5. In Section 6, we conclude the paper.

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 [2] clusters the data based on an in-memory structure called CF-tree instead of the original large data set. Bradley et al. [3] proposed a scalable clustering frame-work which selectively stores important portions of the data, and compresses or discards other portions. CluStream [1] 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 [1] to recall historical micro-clusters for different time durations.

A variety of services on the Web such as news filters, text crawling, and topic detecting etc. have posed requirements for text stream clustering. A few algorithms have been proposed to tackle the problem [4], [5], [6], [7]. 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.

In [8], the authors extended 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 off-line 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.

2.2 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 [9], [10], [11]. Some works try to extract summaries without such salient scores. Wang et al. [12] used the sym-metric non-negative matrix factorization to cluster sentences and choose sentences in each cluster for summarization. He et al. [13] proposed to summarize documents from the perspective of data reconstruction, and select sentences that can best reconstruct the original documents. In [14], 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 tweet [15]. Later, Inouye and Kalita proposed a Hybrid TF-IDF algorithm and a Cluster based algorithm to generate multiple post summaries [16]. In [17], Harabagiu and Hickl leveraged two relevance models for microblog summarization: an event structure model and a user behavior model. Takamura et al. [18] pro-posed a microblog summarization method based on the p-median problem, which takes posted time of microblogs into consideration. Unfortunately, almost all existing document/microblog summarization methods mainly deal with small and static data sets, and rarely pay attention to efficiency and evolution issues.

There have also been studies on summarizing microblogs for some specific types of events, e.g., sports events. Shen et al. [19] proposed to identify the participants of events, and generate summaries based on sub-events detected from each participant. Chakrabarti and Punera [20] 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 [21], Kubo et al. 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. More-over, 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.

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 [22] made early efforts in this area, using timelines to explore the 2008 Presidential Debates by Twitter sentiment. Dork et al. [23] presented a timeline-based backchannel for conversations around events.

In [24], 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 [24], 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. Twi-tInfo [25] developed an algorithm based on TCP congestion detection, while Nichols et al. [26] 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 relation-ships among users, and compute the UserRank as in [31]. As a result, we define a tweet t_i as a tuple: $\eth tv_i; ts_i; w_i Þ$, where tv_i is the textual vector, ts_i is the posted timestamp 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. 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 t_1; t_2; . . . ; t_n, its tweet cluster vector is defined as a tuple: TCV ðCÞ ¼ ð sum v; w sum v; ts1; ts2; n; ft set Þ, where 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 w sum 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). 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.

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 snap-shots 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]:

Definition 2: A pyramidal time frame stores snapshots at differing levels of granularity depending on the recency. Snap-shots are stored into different orders varying from 0 to blog $_a$ðT Þc , where T is the time elapsed since the beginning of the stream and a is an integer (a > 1Þ . 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 ith order occur at time intervals of a^i. Specifically, each snapshot of the ith order is taken at a moment in time when the timestamp from the beginning of the stream is exactly divisible by a^i. At any given moment in time, only the last a^l þ 1ðl _ 1Þ snapshots of order i are stored. Note that for more recent timestamps, the time interval between successive snapshots stored in PTF is smaller (finer granularity). This feature of PTF is consistent with the demand that recent summaries should be of higher quality because people usually care more about recent events.

4. Experiments:

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

4.1 Experiments for Summarization:

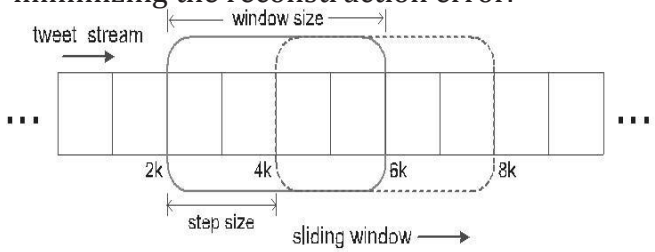
4.1.1 Setup:

We construct five data sets to evaluate summarization. One is obtained by conducting keyword filtering on a large Twitter data set used by [31]. The other four include tweets acquired during one month in 2012 via Twitter's key-word tracking API.[4] As we do not have access to the respective users' social networks for these four, we set their weights of tweets w_i to the default value of 1. Details of the data sets are listed in Table. Ground truth for summaries. As no previous work has conducted similar study on continuous summarization, we have to build our own ground truth (reference sum-maries). 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:

Given time duration, we first retrieve the corresponding tweet subset, and use the following three well recognized summarization algorithms to get three candidate summaries. Cluster Sum [16] clusters the tweets and picks the most weighted tweet from each cluster to form summary. Lex Rank [11] first builds a sentence similarity graph, and then selects important sentences based on the concept of eigenvector

centrality.

DSDR [13] 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.



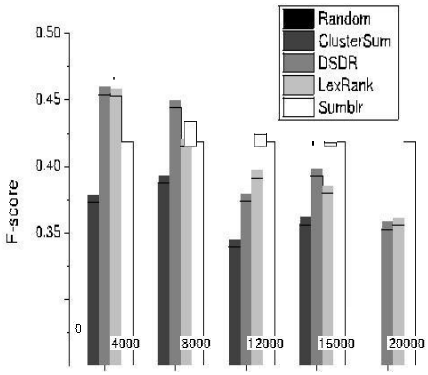


Figure: The sliding window mechanism

4.1.2 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 qualities Fig. 9. Performance of different parameters. (a) Effect of b on quality. (b) Effect of b on efficiency. (c) Effect of N_{max} on quality. (d) Effect of N_{max} on efficiency. (e) Effect of mc. (f) Effect of m. (g) Effect of _. Effect of N_{max}. Figs. 9c and 9d depict the performance of N_{max}. For small N_{max}s, 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 N_{max}s. A balanced value for N_{max} 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.

Effect of m. As shown in Fig. 9f, the summary quality improves when m increases. When m _ 40, the improve-ment is not obvious. Meanwhile, a larger m incurs more storage overhead. We choose m ¼ 40. Finally we check the effect of _ (0 _ 1), which is a tradeoff factor between content coverage and novelty. We gradually vary _ from 0 to 1 at the step of 0.1 to examine its effect, as shown in Fig. 9g. When _ _ 0:7, the extreme emphasis on coverage causes performance loss. Therefore, we set _ ¼ 0:4 as a balanced factor.

5. Flexibility:

One distinguishing 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 (Section 3.3). We fix a at 2 and show the results varying l. For consistency, we extract a sub-set 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, we evaluate the results for different durations with length

len varying from 1 to 10 days. We report the average F-score scoreðtsÞ by interval of 48 hours. Due to space limit, we on show the figure for "Arsenal", results for other data sets are available on the web.[5]

5.1 Observations:

- There exists a common trend: more recent time durations have higher summary quality. This is because PTF has finer granularity of snapshots for more recent moments. As a result, the queried durations can be better approximated.
- A larger l leads to higher overall quality. Due to larger capacity of each order, PTF with a larger l is able to maintain more snapshots, and thus produce more accurate approximation for the queried durations.
- Unfortunately, a larger l also requires more storage cost (the numbers in the parentheses represent the amounts of snapshots in PTF). This is obvious since it enables PTF to store more snapshots, which results in heavier storage burden.

For different applications, Sumblr can be customized with different l values. For example, for real-time summarization, a small l is enough; while for historical review, a large l is needed.

Granularity, to further evaluate the flexibility of Sumblr, we also conduct a granularity test. We partition the one-month data sets into time durations with fixed length (e.g., 24, 72, or 144 hours), then report the average F-scores for these durations under different levels of granularity. As shown in Table 3, summary quality does not have significant difference among different granularities. This is because the quality of a historical summary mainly depends on two end-points of the duration (i.e., the accuracy of duration approximation in PTF) rather than the length of the duration.

5.2 Experiments for Timeline Generation:

5.2.1 Data Sets and Ground Truth:

In this section, we evaluate the effectiveness of topic evolution detection, i.e., timeline generation. We use the "Arsenal" and "Chelsea" data sets in Section 5.1, and add two more recent data sets ("Arsenal2013" and "Chelsea2013"). We choose these data sets because reference timelines for sport topics are relatively easier to build. For this experiment, the reference timelines are manually produced. Specifically, we read through all the related news during those periods of the corresponding data sets from news websites (Yahoo!, ESPN, etc.), and select those dates as nodes on the reference timelines when something important happen, e.g., football matches, players' signing of new con-tracts, etc. The time unit of timelines is day. Statistics of the data sets and the reference timelines (numbers of timeline nodes) are listed in Table 4.

5.2.2 Results:

Our objective is to detect nodes in the reference timeline as the stream proceeds. We compare performance of the topic evolution detection algorithm using three different variations in Section 4.3, i.e., summary based variation, volume-based variation and sum volume variation. We present precision, recall, and F-score of the timeline nodes detected by these methods. Since similar trends are observed in all four data sets, here we only show results for the largest data set Chelsea2013 to save space. Results for other data sets are also available.[5]

Figs. 11 and 12 show the effects of the decision threshold for SUM (t_s) and VOL (t_v), respectively. In both figures, as the threshold increases, recall declines while precision increases. This is expected since higher threshold would exclude more promising candidate nodes, and those remaining nodes with larger variations are more likely to be the correct ones. We get the highest F-scores when t_s ¼ 1:10 for SUM and t_v ¼ 1:05 for VOL. Note that SUM has relatively higher recall while VOL has higher

precision. This is also consistent with our analysis in Section 4.3: SUM detects sub-topic changes based on content variation, but some of them may not be influential enough; VOL finds spikes when lots of people talk about something important, but reduces the chance for other sub-topic changes being detected. In other words, we can consider SUM as a "sensitive" method and VOL as a "conservative" method.

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.

7. 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 Min-ing, 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 repre-sentation 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 cate-gorical data streams," Knowl. Inf. Syst., vol. 24, no. 2, pp. 171–196, 2010.
9. R. Barzilay and M. Elhadad, "Using lexical chains for text summa-rization," in Proc. ACL Workshop Intell. Scalable Text Summarization, 1997, pp. 10–17.
10. 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.
11. 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.
12. D. Wang, T. Li, S. Zhu, and C. Ding, "Multi-document summari-zation 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.
13. Z. He, C. Chen, J. Bu, C. Wang, L. Zhang, D. Cai, and X. He, "Document summarization based on data reconstruction," in Proc. 26th AAAI Conf. Artif. Intell., 2012, pp. 620–626.