International Journal of Research in Advent Technology (E-ISSN: 2321-9637) Special Issue
National Conference "NCPCI-2016", 19 March 2016
Available online at www.ijrat.org

Continuous Summarization and Timeline Generation for Topic Evolutionary Tweet Streams

Mr.Amol Dhepe[1], Bharat Burghate[2]
Department of Computer Engineering
Bhivrabai Sawant Institute of Technology & Research
Wagholi, Pune
Email: dhepeamol4@gmail.com[1], bharatburghate@gmail.com [2]

Abstract- Tweets are being created short text message and shared for both users and data analysts. Twitter which receive over 400 million tweets per day has emerged as an invaluable source of news, blogs, opinions and more. Our proposed work consists three components tweet stream clustering to cluster tweet using k-prototype cluster algorithm (In existing base paper, k-means clustering algorithm used to create the initial clusters. with global cluster, it didn't work well. So in our proposed work, we use k-prototype clustering produce tighter clusters than k-means clustering, especially if the clusters are globular) 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, continuous summarization, timeline, summary.*

1. INTRODUCTION

Growing attractiveness of micro-blogging services such as Twitter, Weibo, and Tumblr has resulted in the explosion of the amount of short-text messages. Twitter, for instance, which receives over 400 million tweets per day1 has emerged as an invaluable source of news, blogs, opinions, and more. Tweets, in their raw form, while being informative, can also be overwhelming. For instance, search for a hot topic in Twitter may yield millions of tweets, spanning weeks. Even if filtering is allowed, plowing through so many tweets for important contents would be a nightmare, not to mention the enormous amount of noise and redundancy that one might encounter. To make things worse, new tweets satisfying the filtering criteria may arrive continuously, at an unpredictable rate. One possible solution to information overload problem is summarization. Summarization represents restating of the main ideas of the text in as few words as possible intuitively, a good summary should cover the main topics (or subtopics) and have diversity among the sentences to reduce redundancy. Summarization is widely used in comfortable arrangement, especially when users surf the internet with their mobile devices which have much lesser screens than PCs. Traditional document summarization approaches, however, are not as effective in the situation of tweets given both the big size of tweets as well as the fast and continuous nature of their arrival.

Tweet summarization, therefore, requires functionalities which significantly differ. From traditional summarization. In general, tweet summarization has to take into consideration the temporal feature of the arriving tweets. Consider a user interested in a topic - related tweet stream, for example, tweets about "Apple". A tweet summarization system will continuously monitor "Apple" related tweets producing a real time timeline of the tweet stream. A user may explore tweets based on a timeline (e.g. "Apple" tweets posted between Oct. to Nov.). 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 in topic - relevant tweets, but also

support a range of data analysis tasks such as instant reports or historical survey.

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

2. LITERATURE SURVEY

2.1Characterizing debate performance via aggregated twitter sentiment

Television broadcasters are beginning to combine social micro-blogging systems such as Twitter with television to create social video experiences around events. We looked at one such event, the first U.S. presidential debate in 2008, in conjunction with aggregated ratings of message sentiment from Twitter. We begin to develop an analytical methodology and visual representations that could help a journalist or public affairs person better understand the temporal dynamics of sentiment in reaction to the debate video. We demonstrate visuals and metrics that can be used to detect sentiment pulse, anomalies in that pulse, and indications of controversial topics that can be used to inform the design of visual analytic systems for social media events.

2.2 A Visual Backchannel for Large-Scale Events

We introduce the concept of a Visual Backchannel as a novel way of following and exploring online conversations about large-scale events. Micro blogging communities, such as Twitter, are increasingly used as digital backchannels for timely exchange of brief comments and impressions during political speeches, sport competitions, natural disasters, and other large events. Currently, shared updates are typically displayed in the form of a simple list, making it difficult to get an overview of the fast-paced discussions as it happens in the moment and how it evolves over time. In contrast, our Visual Backchannel design provides an evolving, interactive, and multi-faceted visual overview of large-scale ongoing conversations on Twitter. To visualize a continuously updating information stream, we include visual saliency for what is happening now and what has just happened, set in the context of the evolving conversation. As part of a fully web-based coordinated-view system we introduce Topic Streams, a temporally adjustable stacked graph visualizing topics over time, a People Spiral representing participants and their activity, and an Image Cloud encoding the popularity of event photos by size. Together with a post listing, these mutually linked views support cross-filtering along topics, participants, and time ranges. We discuss our design considerations, in particular with respect to evolving visualizations of dynamically changing data. Initial feedback indicates significant interest and suggests several unanticipated uses.

2.3 Evolutionary timeline summarization: a balanced optimization framework via iterative substitution

Classic news summarization plays an important role with the exponential document growth on the Web. Many approaches are proposed to generate summaries but seldom simultaneously consider evolutionary characteristics of news plus to traditional summary elements. Therefore, we present a novel framework for the web mining problem named Evolutionary Timeline Summarization (ETS). Given the massive collection of time-stamped web documents related to a general news query, ETS aims to return the evolution trajectory along the timeline, consisting of individual but correlated summaries of each date, emphasizing relevance, coverage, coherence and cross-date diversity. ETS greatly facilitates fast news browsing and knowledge comprehension and hence is a necessity. We formally formulate the task as an optimization problem via iterative substitution from a set of sentences to a subset of sentences that satisfies the above requirements, balancing coherence/diversity measurement and local/global summary quality. The optimized substitution is iteratively conducted by incorporating several constraints until convergence. We develop experimental systems to evaluate on 6 instinctively different datasets which amount to 10251 documents.

2.4 Twit-info: aggregating and visualizing microblogs for event exploration

Microblogs are a tremendous repository of user-generated content about world events. However, for people trying to understand events by querying services like Twitter, a chronological log of posts makes it very difficult to get a detailed understanding of an event. In this paper, we present Twit-Info, a

system for visualizing and summarizing events on Twitter. Twit-Info allows users to browse a large collection of tweets using a timeline-based display that highlights peaks of high tweet activity. A novel streaming algorithm automatically discovers these peaks and labels them meaningfully using text from the tweets. Users can drill down to sub-events, and explore further via geo-location, sentiment, and popular URLs. We contribute a recall-normalized aggregate sentiment visualization to produce more honest sentiment overviews. An evaluation of the system revealed that users were able to reconstruct meaningful summaries of events in a small amount of time. An interview with a Pulitzer Prize-winning journalist suggested that the system would be especially useful for understanding a long-running event and for identifying eyewitnesses.

2.5 Summarizing sporting events using twitter

The status updates posted to social networks, such as Twitter and Facebook, contain a myriad of information about what people are doing and watching. During events, such as sports games, many updates are sent describing and expressing opinions about the event. In this paper, we describe an algorithm that generates a journalistic summary of an event using only status updates from Twitter as a source. Temporal cues, such as spikes in the volume of status updates, are used to identify the important moments within an event, and a sentence ranking method is used to extract relevant sentences from the corpus of status updates describing each important moment within an event.

2.6 A Probabilistic Model for Online Document Clustering with Application to Novelty Detection

In this paper we propose a probabilistic model for online document clustering. We use non-parametric Dirichlet process prior to model the growing number of clusters, and use a prior of general English language model as the base distribution to handle the generation of novel clusters. Furthermore, cluster uncertainty is modeled with a Bayesian Dirichlet-multinomial distribution. We use empirical Bayes method to estimate hyper parameters based on a historical dataset.

2.7 On clustering massive text and categorical data streams

In this paper, we will study the data stream clustering problem in the context of text and categorical data domains. While the clustering problem has been studied recently for numeric data streams, the problems of text and categorical data present different challenges because of the large and un-ordered nature of the corresponding attributes. Therefore, we will propose algorithms for text and categorical data stream clustering. We will propose a condensation based approach for stream clustering which summarizes the stream into a number of fine grained cluster droplets. These summarized droplets can be used in conjunction with a variety of user queries to construct the clusters for different input parameters. Thus, this provides an online analytical processing approach to stream clustering. We also study the problem of detecting noisy and outlier records in real time.

2.8 Using Lexical Chains for Text Summarization

We investigate one technique to produce a summary of an original text without requiring its full semantic interpretation, but instead relying on a model of the topic progression in the text derived from lexical chains. We present a new algorithm to compute lexical chains in a text, merging several robust knowledge sources: the Word Net thesaurus, a part-of-speech tagger, shallow parser for the identification of nominal groups, and a segmentation algorithm. Summarization proceeds in four steps: the original text is segmented, lexical chains are constructed, strong chains are identified and significant sentences are extracted. We present in this paper empirical results on the identification of strong chains and of significant sentences. Preliminary results indicate that quality indicative summaries are produced.

2.9 Multi-document summarization by maximizing informative content-words

We show that a simple procedure based on maximizing the number of informative content-words can produce some of the best reported results for multi-document summarization. We first assign a score to each term in the document cluster, using only frequency and position information, and then we find the set of sentences in the document cluster that maximizes the sum of these scores, subject to length constraints. Our overall results are the best reported on the DUC-2004 summarization task for the ROUGE-1 score, and are the best, but not statistically significantly different from the best system in MSE-2005.

2.10 LexRank: graph-based lexical centrality as salience in text summarization

We introduce a stochastic graph-based method for computing relative importance of textual units for Natural Language Processing. We test the technique on the problem of Text Summarization (TS). Extractive TS relies on the concept of sentence salience to identify the most important sentences in a document or set of documents. Salience is typically defined in terms of the presence of particular important words or in terms of similarity to a centroid pseudo-sentence. We consider a new approach, LexRank, for computing sentence importance based on the concept of eigenvector centrality in a graph

representation of sentences. In this model, a connectivity matrix based on intra-sentence cosine similarity is used as the adjacency matrix of the graph representation of sentences. Our system, based on LexRank ranked in first place in more than one task in the recent DUC 2004 evaluation. In this paper we present a detailed analysis of our approach and apply it to a larger data set including data from earlier DUC evaluations. We discuss several methods to compute centrality using the similarity graph. The results show that degree-based methods (including LexRank) outperform both centroid-based methods and other systems participating in DUC in most of the cases. Furthermore, the LexRank with threshold method outperforms the other degree-based techniques including continuous LexRank.

3. PROPOSED SYSTEM

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

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

1. Tweet Stream Initialization
2. Incremental Clustering
3. Deleting Outdated Clusters
4. Merging Clusters

Tweet Stream Initialization

At the start of the stream, we collect a small number of tweets and use a k-prototype clustering algorithm (instead of k-means) to create the initial clusters. 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, 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).

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.

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

4. MODULE DESCRIPTION

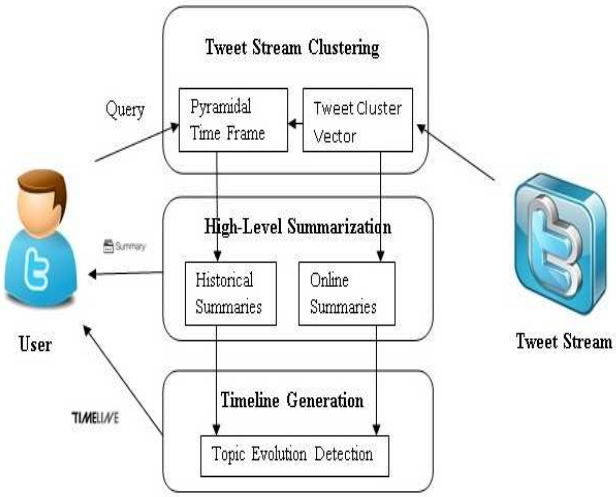

Fig. 1 System Architecture

Load Dataset

- In this module, we load the Twitter data sets. Because 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 project, we propose a novel continuous summarization framework called slumber to alleviate the problem. So we load the dataset for continuous summarization and timeline generation.

Tweet Stream Clustering

- In this 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.
- It consists of 4 phases such as
  1) Tweet Stream Initialization
  2) Incremental Clustering
  3) Deleting Outdated Clusters
  4) Merging Clusters
- Tweet Stream Initialization: At the start of the stream, we collect a small number of tweets and use a k-prototype clustering algorithm (instead of k-means) to create the initial clusters. 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, 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).

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.

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.
- It presented a timeline - based backchannel for conversations around events. It 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.

5. CONCLUSION

In this project, we proposed a continuous tweet stream summarization framework, namely slumber, to generated summaries and timelines in the context of streams. Slumber employs a tweet stream clustering algorithm to compress tweets into TCVs and maintains them in an online fashion. Our proposed k-prototype clustering algorithm produced tighter clusters than k- means clustering, especially 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 detected automatically, allowing slumber to produce dynamic timelines for tweet streams.

REFERENCES

1] 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.

[2] M. Dork, D. Gruen, C. Williamson, and S. Carpendale, "A visual backchannel for large- scale events," IEEE Trans. Vis. Comput. Graph. vol. 16, no. 6, pp. 1129–1138, Nov. 2010.

[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] A. Marcus, M. S. Bernstein, O. Badar, D. R. Karger, S. Madden, and R. C. Miller, "Twitinfo: Aggregating and visualizing micro-blogs for event exploration," in Proc. SIGCHI Conf. Human Factors Comput. Syst. , 2011, pp. 227–236

[5] J. Nichols, J. Mahmud, and C. Drews, "Summarizing sporting events using twitter," in Proc. ACM Int. Conf. Intell. User Interfaces, 2012, pp. 189–198.

[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] 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.

[8] R. Barzilay and M. Elhadad, "Using lexical chains for text summa-rization," in Proc. ACL Workshop Intell. Scalable Text Summarization, 1997, pp. 10–17.

[9] 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.

[10] 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.