International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 6.887
Volume 6 Issue III, March 2018- Available at www.ijraset.com

RAMS: Real-time Anomaly Monitoring System

M. Vidhya Sri[1], Srinidhi Chelmeda[2], Ashwini S Tagadghar[3], S. Avinash[4]
[1, 2, 3] *B. Tech, CSE, TKREC, Hyderabad*
[4,] *M. Tech CSE, Assistant professor TKREC, Hyderabad*

Abstract: ***Microblog platforms have been exceptionally happening in the big data epoch due to its real-time dissipation of facts and figures. It's crucial to know what anomalous incidents are trending on the social network and be able to supervise their unfolding and find analogous abnormalities. In this project, we have established RAMS, an actual-time arising anomaly supervising scheme over microblog text contents. RAMS assimilates our attempt on both arising anomaly monitoring analysis and system investigation. From the arising supervising aspect, RAMS introduces a graph analytic method such that, RAMS can detect arising anomalies at a prior stage in comparison to the existing approaches, RAMS is among the first to design the arising anomalies interrelationships in a streaming manner, RAMS is able to monitor anomaly evolutions in real-time at diverse time ranges from minutes to months. From the system research aspect, RAMS enhances time-ranged keyword query execution of a full-text search engine to reform the capability of arising anomaly evolution, it also enhances the real-time graph developing performance of Spark and instruments our graph stream model on it, As an outcome, RAMS is able to process big data to the entire Weibo or Twitter text stream with continuous horizontal extensibility. The system distinctly presents its dominance over current existing systems and approaches from both the event monitoring aspect and the system perspective for the arising event monitoring project.***

Keywords: RAMS, monitoring, anomaly, detection, correlation, evolution, graph, query, keywords.

I. INTRODUCTION

Micro blogging is a combination of blogging and instant messaging that allows users to create short messages to be posted and shared with an audience online. Social platforms like Twitter and Weibo have become extremely popular forms of this new type of blogging, especially on the mobile web, making it much more convenient to communicate with people compared to the days when desktop web browsing and interaction was the norm. It creates a way that is easier and faster to communicate with people online and keep them informed about relevant, shareable information at the same time. The short and noisy nature of microblog text stream makes traditional topic detection methods and their derivatives inappropriate for the task. Latent space topic model takes a long time to train and the topics are fixed. While the online event monitoring scenario demonstrate fast changing set of topics and require high throughput of data processing. Nor are they able to perform early detection of emerging anomalous events in real-time. From an emerging anomaly monitoring perspective, we require early detection of emerging anomaly before they go viral, hierarchical view of correlated sub-events as different aspects of an event, monitoring and reveal the evolution of anomaly events. For instance, it can detect and respond to emergency events in a timely manner [1], [2], [3], [4], [5], track event evolution [6], [7], and summarize trending events [8], [9], [10]. Signitrend[5] could detect potential anomalous events at small scale but is confined to only detection, and could not provide tracking and correlation analysis. The method would also tend to generate anomalous events that only contain a single keyword as description, which is hard to comprehend for users.

II. EXISTING SYSTEM

Twitter Monitor provides online disclosure for general arising anomalous events but could not report the numerous aspects of the events nor track the evolution of them. Signitrend is restrained to only disclosure of potential abnormal events at small scale but could not represent tracking and correlation study. They consume a lot of time during the refurbishing of graph structure. They also cannot rebalance workload when some nodes in the graph are more often updated than others, which is a common case when processing trending words in microblog texts.

III. PROPOSED SYSTEM

Here, we propose RAMS, a real-time anomaly monitoring system over text streams from various microblogging sites such as Twitter, Weibo etc. In this paper, our main purpose is to supervise arising anomalous events on microblog platforms by using methods based on graph mining approaches, which grants unique opportunities to incorporate our arising anomaly monitoring exploration and system developments. RAMS system incorporates early disclosure, correlation study and temporal evolution tracking of abnormal events. Early disclosure would enable us to capture emerging events before they get popular and start trending.

Correlation study would automatically report numerous aspects of the anomalous event, or categorical structure of related anomalies.

IV. ARCHITECTURE

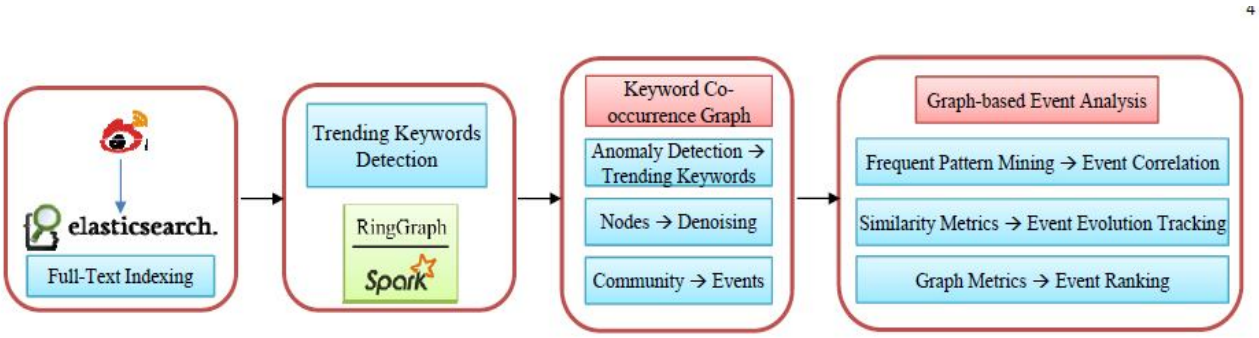


Fig. 1 RAMS Architecture

Graph mining techniques grants unique opportunities to incorporate our anomaly monitoring exploration and system development for optimized performance.

1) Trending Keyword Detection: We consider trending keyword detection task as an outlier detection for busty usage. We define the activity level change for a node. Typically, such change is most informative when the time period is 1/λ, i.e. the half-life of edge weight decay. At each detection period, we perform noise filtering over both kinds of keywords. We first try to remove unrelated co-occurred keywords that have not appeared with a trending keyword recently.

2) Keyword Co-occurrence Graph: A continuously updating graph of keywords and their co-occurrence relationship is built from the text stream. To improve the efficiency of graph update and computation process, an optimized system is developed based on Spark and GraphX and implemented our event monitoring system on it.

3) Evolution Tracking: Event tracking aims to trace temporal evolution of events along the timeline. Under the assumption that related events share at least one noun in their keyword sets, a candidate set CS of potential precedent events is retrieved using nouns in the keyword set.

V. ALGORITHM

The MapReduce framework operates exclusively on <key, value> pairs, that is, the framework views the input to the job as a set of <key, value> pairs and produces a set of <key, value> pairs as the output of the job, conceivably of different types.

1) *Map stage:* The map or mapper's job is to process the input data. Generally the input data is in the form of file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.

2) *Reduce stage:* This stage is the combination of the Shuffle stage and the Reduce stage. The Reducer's job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.

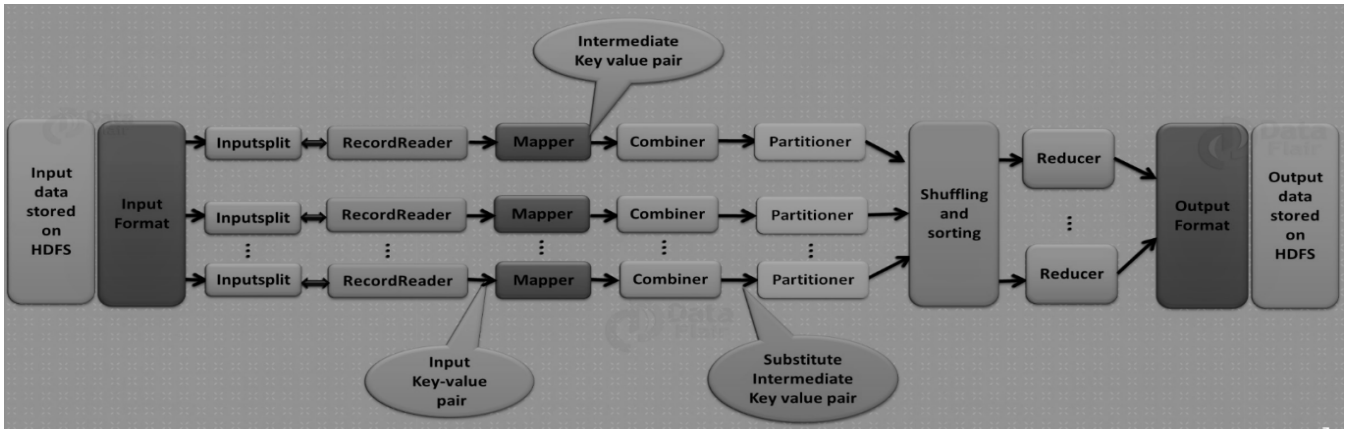


Fig. 2 MapReduce flowchart

Initially, the given input data is stored in the Hadoop Distribution File System (HDFS) in some particular format. Then this input data is divided into many inputsplits. Each InputSplit can be of size 64 MB at maximum.

Each InputSplit is assigned to a single Mapper. The RecordReader acts as an interface between the InputSplit and the Mapper. RecordReader reads <key, value> pairs from an InputSplit. RecordReader, typically, converts the byte-oriented view of the input, provided by the InputSplit, and presents a record-oriented view for the Mapper & Reducer tasks for processing.

Mapper maps input key/value pairs to a set of intermediate key/value pairs.

Maps are the individual tasks that transform input records into intermediate records. Overall, Mapper implementations are passed the JobConf for the job method and override it to initialize themselves. The framework then calls map for each key/value pair in the InputSplit for that task.

A given input pair may map to zero or many output pairs. Output pairs are collected. Applications can use the Reporter to report progress, set application-level status messages and update Counters, or just indicate that they are alive. All intermediate values associated with a given output key are subsequently grouped by the framework, and passed to the Reducer(s) to determine the final output.

The Combiner class is used in between the Map class and the Reduce class to reduce the volume of data transfer between Map and Reduce. Usually, the output of the map task is large and the data transferred to the reduce task is high.

Shuffle phase in Hadoop transfers the map output from Mapper to a Reducer in MapReduce. Sort phase in MapReduce covers the merging and sorting of map outputs. Every reducer obtains all values associated with the same key. Shuffle and sort phase in Hadoop occur simultaneously and are done by the MapReduce framework.

In this phase the reduce method is called for each <key, (collection of values)> in the sorted inputs. The output of the reduce task is typically written to a RecordWriter.

Reducer takes a set of an intermediate key-value pair produced by the mapper as the input and runs a Reducer function on each of them.

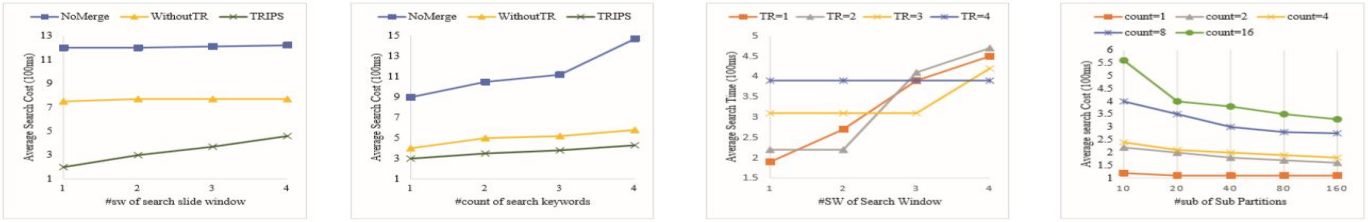
One can aggregate, filter, and combine this data (key, value) in a number of ways for a wide range of processing. Reducer first processes the intermediate values for particular key generated by the map function and then generates the output.

VI. MODULES

The modules that are considered to have much significance in this are as follows.

1) *Data Collection Module:* A distributed crawler is built to collect data from Twitter. The crawler continuously collects the latest microblogs published by users preferably with a large number of followers. It has a master/slave architecture. The master node utilizes key/value store to perform task scheduling. Slave nodes get the assignments from the master and crawl data. A task would monitor the reposts and comments of an original tweet and retrieve the repost and comment list, from which we can construct the forward graph of each tweet. The tasks are scheduled according to posts' priority, which is weighed by the number of reposts and comments.
2) *Indexing Module:* A distributed online index system for temporal microblog data. The whole index is divided into fine-grained time range partitions to provide locality for data access according to temporal proximity. For better concurrent access, each time range partition is divided into sub-partitions by hash functions. Each term's inverted tweet list is only mapped to corresponding sub-partitions. With these structures, given a query with a specific time range the time range information can be used to navigate to corresponding time range partitions and then utilize query to quickly navigate to corresponding sub-partitions.
3) *Processing Data Module:* To efficiently update graph structure, we introduced a hash based graph partitioning method to support find-grained and rapid update. To support incremental computation on evolving graphs, we designed an application scheduler to coordinate applications with different computing request frequency and control the use of compute resources.
4) *Early prediction module:* The anomaly detection method consists of two steps: trending keyword detection and community detection over keywords. Our intuition is that the essential keywords about an event would have similar trends and show a burst in usage compared to their own history. An event is represented as a "bag" of keywords that co-occur and correlate with each other, with its detection time and representative tweet extracted for better comprehension. Then the trending keywords and their co-occurrence relationship is a strong combination to distill emerging anomalous events.

VII. RESULT ANALYSIS



(a) cost on different *DW* **(b)** cost on different *count* **(c)** cost on different *SW* and *TR* **(d)** cost on different *count* and *sub*

Fig 3. Efficiency of RAMS Index Searching

We generate 10,000 and 100,000 messages per batch to simulate real graph stream update scenarios. The update messages consist of three types of updates, i.e., update of vertex attribute, edge attribute and insertion of new nodes and edges, at a ratio of 25%, 25% and 50%. RAMSGRAPH had a speed up of 3.7X, 7.1X and 14.3X respectively compared to GraphX due to finer granularity of updates.

VIII. CONCLUSION

We have demonstrated RAMS, a real-time emerging event monitoring system over microblog platforms, which integrates our efforts on both emerging event monitoring research and system research. RAMS is able to monitor emerging event as to detect emerging events, build event correlations and trace event evolutions. Further, its infrastructure is equipped with customized optimization on its full-text search engine and distributed graph processing engine to perform event monitoring more efficiently. It also supports event and text queries and many other functionalities to assist the analysis of emerging events, as demonstrated in the user interface. The system clearly presents its advantages over existing systems and methods from both technical and system perspectives for the emerging event monitoring task.

IX. FUTURE ENHANCEMENTS

In the future, we would also develop this software not only for microblogging websites but also for other blogging websites which does not have any character limits.

REFERENCES

[1] T. Sakaki, M. Okazaki, and Y. Matsuo, “Earthquake shakes twitter users: real-time event detection by social sensors,” in WWW, 2010.
[2] R. McCreadie, C. Macdonald, I. Ounis, M. Osborne, and S. Petrovic, “Scalable distributed event detection for twitter,” in IEEE BigData, 2013.
[3] Y. Chen, H. Amiri, Z. Li, and T.-S. Chua, “Emerging topic detection for organizations from microblogs,” in SIGIR, 2013.
[4] W. Xie, F. Zhu, J. Jiang, E.-P. Lim, and K. Wang, “Topicsketch: Realtime bursty topic detection from twitter,” in ICDM, 2013.
[5] E. Schubert, M. Weiler, and H.-P. Kriegel, “Signitrend: scalable detection of emerging topics in textual streams by hashed signifi- cance thresholds,” in KDD, 2014.
[6] F. Wei, S. Liu, Y. Song, S. Pan, M. X. Zhou, W. Qian, L. Shi, L. Tan, and Q. Zhang, “Tiara: a visual exploratory text analytic system,” in KDD, 2010.
[7] P. Lee, L. V. Lakshmanan, and E. E. Milios, “Incremental cluster evolution tracking from highly dynamic network data,” in IEEE International Conference on Data Engineering (ICDE), 2014, pp. 3–14.
[8] C. Li, A. Sun, and A. Datta, “Twevent: segment-based event detection from tweets,” in CIKM, 2012.
[9] D. Metzler, C. Cai, and E. Hovy, “Structured event retrieval over microblog archives,” in HLT-NAACL, 2012.
[10] C. Budak, T. Georgiou, and D. A. A. El Abbadi, “Geoscope: Online detection of geo-correlated information trends in social networks,” PVLDB, vol. 7, no. 4, 2013.
[11] J. Allan, R. Papka, and V. Lavrenko, “On-line new event detection and tracking,” in SIGIR, 1998.
[12] T. Hofmann, “Probabilistic latent semantic analysis,” in UAI, 1999.
[13] D.M.Blei,A.Y.Ng,and M.I.Jordan,“Latent dirichlet allocation,” the Journal of machine Learning research, vol. 3, pp. 993–1022, 2003.
[14] D. M. Blei and J. D. Lafferty, “Dynamic topic models,” in ICML, 2006.
[15] X. Yan, J. Guo, Y. Lan, and X. Cheng, “A biterm topic model for short texts,” in WWW, 2013.
[16] M.Mathioudakisand N.Koudas, “Twittermonitor: trend detection over the twitter stream,” in SIGMOG, 2010.
[17] R.Xie,F.Zhu,H.Ma,W.Xie,and C.Lin,“Clear:Areal-timeonline observatory for bursty and viral events,” PVLDB Demo, 2014.
[18] H.Cai,Z.Huang,D.Srivastava,andQ.Zhang,“Indexingevolving events from tweet streams,” Knowledge and Data Engineering, IEEE Transactions on, vol. 27, no. 11, pp. 3001–3015, 2015.
[19] C. Chen, F. Li, B. C. Ooi, and S. Wu, “Ti: an efficient indexing mechanism for real-time search on tweets,” in Proceedings of the 2011ACMSIGMOD International Conference on Management of data. ACM, 2011, pp. 649–660.
[20] W. Yu, C. C. Aggarwal, S. Ma, and H. Wang, “On anomalous hotspot discovery in graph streams,” in ICDM, 2013.