MONITORING AND CLUSTERING EVENTS IN KNOWLEDGE ENGINEERING

D.Priya,
UG Scholar,
Computer Science and Engineering,
Dhanalakshmi College of Engineering,
Chennai,India.

C.Kayalvizhi,
Assistant Professor,
Computer Science and Engineering,
Dhanalakshmi College of Engineering,
Chennai,India.

Abstract - Data mining, the extraction of hidden predictive information from large databases, is a powerful new technology with great potential to help companies focus on the most important information in their data warehouses. In this paper, we propose an idea for monitoring and grouping the events that occur in tweet streams. This may help us to scour databases for hidden patterns, finding predictive information that experts may miss because it lies outside their expectations. We capture the events using four operations (create, absorb, split and merge). The posted tweet is grouped by a keyword in that post. When a tweet is posted it is compared by another post and forms a group using the words in the tweet. In addition, we also find the nearest neighbour who lies in the similar line based on the tweets posted by them. Moreover, we propose a novel event indexing structure, called Multi-layer Inverted List (MIL), to manage dynamic event databases for the acceleration of large-scale event search and update. Extensive experiments have been conducted on a large-scale real-life tweet dataset. The results demonstrate the promising performance of our event indexing and monitoring methods on both efficiency and effectiveness.

Keywords: event monitoring, multi-layer inverted list, nearest neighbour.

I. INTRODUCTION

Data mining, the extraction of hidden predictive information from large databases, is a powerful new technology with great potential to help companies focus on the most important information in their data warehouses. Data mining tools predict future trends and behaviours, allowing businesses to make proactive, knowledge-driven decisions. The automated, prospective analyses offered by data mining move beyond the analyses of past events provided by retrospective tools typical of decision support systems. Data mining tools can answer business questions that have traditionally been too time consuming to resolve. They scour databases for hidden patterns, finding predictive information that experts may miss because it lies outside their expectations.

Most companies already collect and refine massive quantities of data. Data mining techniques can be implemented rapidly on existing software and hardware platforms to enhance the value of existing information resources, and can be integrated with new products and systems as they are brought on-line. When implemented on high performance client/server or parallel processing, computers, data mining tools can analyse massive databases to deliver answers to questions such as, "Which clients are most likely to respond to my next promotional mailing, and why?"

Examples of profitable applications illustrate its relevance to today's business environment as well as a basic description of how data warehouse architectures can evolve to deliver the value of data mining to end users. Growing Data Volume The main reason for necessity of automated computer systems for intelligent data analysis is the enormous volume of existing and newly appearing data that require processing. The amount of data accumulated each day by various businesses, scientific, and governmental organizations around the world are daunting. According to information from the GTE research center, only scientific organizations store each day about 1 TB (terabyte!) of new information.

Two other problems that surface when human analyst process data are the inadequacy of the human brain when searching for complex multifactor dependencies in data, and the lack of objectivity in such an analysis. A human expert is always a hostage of the previous experience of investigating other systems. Sometimes this helps, sometimes this hurts, but it is almost impossible to get rid of this fact.

One additional benefit of using automated data mining systems is that this process has a much lower cost. While data mining does not eliminate human participation in solving the task completely, it significantly simplifies the job and allows an analyst who is not a professional in statistics and programming to manage the process of extracting knowledge from data.

Data mining process for monitoring and grouping events. Such initiation helps to handle with posting the tweets in the wall. Tweets includes both text and images with jpg, png and gif image formats. The short format of the tweet is a defining characteristic of the service, allowing informal collaboration and quick information sharing that provides relief. Its based on social networking which is used for both personal and business purpose. Its used to share our feelings in the wall via post. For business its used as a blog to communicate. The tweets posted are automatically grouped using their respective title. The post which is not related to the title is removed. Remaining are grouped. Data mining process for monitoring events. Such initiation helps to handle sharing the image without losing the original quality.

Tweet streams provide a variety of real-life and real-time information on social events that dynamically change over time. In this paper, Although social event detection has been actively studied, how to efficiently monitor evolving events from continuous tweet streams remains open and challenging. However, this approach does not track the evolution of events, nor does it address the issue of efficient monitoring in the presence of a large number of events.

In this project, we capture the dynamics of events using four event operations (create, absorb, split and merge) of tweets, which can be effectively used to monitor evolving events. First the post is created. The post includes images. Then, the post is absorbed and monitored. The post is then splitted to groups and members. Merged when search is done. The posted tweet is grouped by a keyword in that post. Tweet is compared by text summerization and is grouped using the keyword. When a tweet is posted it is compared by another post and forms a group using the words in the tweet and another post comes in it is also compared and grouped. The post doesn't match the title is belong to a member.

II. RELATED WORK

Twitter received much attention in recent years. Twitter data is openly available, motivating research in social interactions on the Web, micro-blogging and data mining. At the same time, Twitter differs from other blogging software due to its shorter messages, facilitating up-to-date publishing. Researchers have been motivated to analyze Twitter as a source of sensory information provided by Twitter users, reacting on real-life events such as social events or natural occurrences as earthquakes . The most prominent works investigating event detection on Twitter are based on statistical and machine learning techniques. Past works involved classification and particle filtering methods to event detection from Twitter messages, reporting a significant accuracy in detecting earthquakes. Another proposed an approach to group event-related tweets in real-time applying Hidden Markov Models. These approaches can be used for well-structured events, but requires prior knowledge on events, participating athletes and defined event-related hashtags. [1] applied an online clustering approach, grouping tweets with similar content together. After manually labeling clusters as event-related and not event-related, they trained a Naive Bayes text classifier for identifying event-related tweets. This approach, however, requires calculating pairwise similarities before actually identifying tweets as related to events. [2] used named entity recognition and decision trees, calculating the quantity of found named entities in time. Other works identify event content using other information sources besides Twitter.

Event Detection. Recently, considerable efforts have been devoted to summarizing online textual streams such as tweets in the form of events. There are various ways to give a taxonomy of all these related works. As shown in [11], according to the detection task, the techniques can be classified into retrospective event detection (RED) [12] and new event detection (NED) [13]. Based on the event type, they are categorized into specified [14] and unspecified [15] event detection. The former relies on the prior available information on the event of interest, while the latter detects events from the bursts or trends in Twitter streams. Depending on the detection method, there are supervised [16] and unsupervised [17] algorithms. In this paper, we focus on NED, unspecified and unsupervised event detection. We under-take a brief review of several representative studies in the same category. To capture emerging events, [18] identifies the locally dense sub-graphs from a graph under a streaming model. In [17], daily signals for each word are constructed by applying wavelet analysis. The words are clustered into events using a modularity-based graph partitioning algorithm. This kind of daily detected events lose track of their developments over multiple days. In [19], the single-pass incremental clustering algorithm is adopted for event detection and the threshold is dynamically generated by the statistics of existing clusters. However, the influence of cluster center shift is ignored, thus it fails to capture the dynamics of events. Event Evolution. Most of the work on event evolution tracks the details of one specified event in real time. For example, in [1], the authors track the representative tweets of an event and mine geographical diffusion trajectory on the map. Instead of tracking one individual event, we focus on event evolution, aiming to monitor the event evolution among relevant events. To the best of our knowledge, [5] is the only effort to monitor event evolution. In their work, a subgraph-by-subgraph incremental tracking framework is proposed for event monitoring. A skeletal graph is designed to summarize the information within a fading time window in a dynamic network. However, the evolution graph is constructed by treating each event snapshot as a node and the trajectory between snapshots as paths. So the evolution of an event is only checked when the time window moves. This time window strategy faces the problem that long time window length may lead to losing track of highly dynamic events' evolutions and short time window length will lead to storing redundant snapshots for steady events.

Indexing for Twitter. Unlike traditional text indexing, the indexing structure for Twitter should be capable of ingesting the data rapidly and support efficient search and quick update on large-scale data. In [24], a retrieval engine which supports Twitter's real-time search is introduced. The authors focus on how to organize the inverted list indexing to support low-latency, high-throughput retrieval and how to implement concurrency management. In [25], a query-based classification approach is designed to distinguish between tweets that may appear as a search result with high probability and the noisy tweets. The former will be indexed by the inverted list in real-time, and the latter will be indexed in a background batch scheme. As we can see, all these indexing struc-tures are constructed to index tweets and the search algorithms are designed to support fast tweet search on tweets data not event search on the detected events (clusters of tweets). The difference is that only a limited number of words exist in one tweet (even with weights) and the tweets search uses a boolean query language, i.e., search tweets containing (or not containing) specified search term(s); while for events, there might be hundreds of words in one event and only a few of them are dominant words (i.e., words with much higher weight than others). Moreover, event search is nearest neighbour search which is more complex than boolean queries. To the best of our knowledge, only one event indexing structure named variable dimensional extendible hash (VDEH) has been proposed before. Given a query tweet, they use the fraction of matched tweets in an event to calculate the similarity between a query tweet and an event. VDEH stores highly similar tweets together in one bucket of the hash to accelerate the comparison between the query tweet and each event. Besides indexing, another approach for improving efficiency is the distributed processing topology. However, for the highly dynamic Twitter data stream, detecting events immediately after the tweets are posted is of great importance. Unlike the distributed system, our proposed in-memory indexing structure can process data in real time by avoiding disk IO.

Our work in this paper can be distinguished from previous work in several ways. First, in addition to event detection, we aim to track evolving events over time. Unlike previous work which checks the evolution patterns only when the time window moves, we record the evolution patterns and identify event relationships whenever events evolve. Second, we design an indexing structure for event databases to support event search and update. Although VDEH has been proposed to index events, the actual data indexed by VDEH contains all the tweets in each event, which leads to large storage cost. In contrast, ours is an in-memory indexing structure which stores only the summary representation of the event. By avoiding the IO cost, we are able to process the highly dynamic tweets data in real time. Third, we design new and tight upper bounds on the Cosine similarity between tweets and events to quickly reduce the search space for nearest neighbour search. Fourth, existing work on indexing tweets mostly focuses on organizing the quickly incoming tweets while our work aims to improve the performance of searching the detected events given tweets.

III. EXISTING SYSTEM

Social event detection has been actively studied, how to efficiently monitor evolving events from continuous tweet streams remains open and challenging. However, this approach does not track the evolution of events, nor does it address the issue of efficient monitoring in the presence of a large number of events. Using Biometric function to detecting the face. Existing system event detection which is less efficient. Existing system fail to monitor event evolutions in real time. Existing system performance is low. Existing system fail to track all the events which is posted. We can see major two issues in the existing system. First, once a tweet is posted it is not splitted into members and groups. Second, it lacks in grouping the tweets. To group a tweets it compares all the tweets. Which takes more time. Also there is no option for viewing the nearest neighbour who matches with our tweets.

IV. PROPOSED SYSTEM

In this system we are using text summerization to compare the text and group the tweet. We use four operations to capture the dynamic events evolution pattern, including creation of tweets, absorption of tweets, spliting of tweets and merge tweets. Implementation of fast search. Event detection and Tracking in twitter is more efficient using our system.

We use four event operations posting of tweets, absorb the posted tweets, splitting of posted tweets and merging of posted tweets. We propose a novel event indexing structure, referred to as the Multilayer Inverted List, to facilitate event search and event update for large-scale, dynamic event database. We present efficient algorithms for nearest neighbor search with upper bound pruning to avoid a large proportion of expensive event similarity computations. We conduct an extensive performance study on dynamic event databases generated from over 10 million tweets and the results demonstrate the superiority of our methods over existing methods.

Here we use a algorithm named New Tweet algorithm. The efficiency of our approach is achieved via this algorithm. It is based on the fact that it groups the tweets based on the keyword. The grouped tweets are put under one single group and that group is named relevant to the keyword. The search that it makes is the finest approach. It works on the fact that no words with two or three characters can be a keyword or an important word that would represent the tweet. Hence, those words are ignored by this algorithm. It processes the tweets in three steps namely process a new tweet, merge event, nearest neighbor search.

Processing a new tweet is the first step in this process. The user can post their tweets in their wall. The tweets can be of any type either it can be a text or an image. In both cases, the first step to be processed while handling the tweets is absorbing the tweets. The tweets that are posted are absorbed by the system. Each user's activity will be monitored by the administrator. In case of a new tweet the tweet should be absorbed first then it should be read character by character where the words with two letters and three letters are ignored. The words greater than three letters are compared with the keywords that are already stored in the database. Based on the keywords the class of the tweet is found and a new group is created and the following tweet is posted in that group. The user will receive notifications regarding the same.

Merge event follows the above. This is used in the incremental event evolution technique. The basic idea behind this is that once a new tweet is posted across the network it is processed as the same way that a new tweet is processed. Except that, here the tweets are just simply merged to the existing groups.

For a new arriving tweet or newly split event, we are first interested in identifying its most similar event from the existing events indexed by MIL, corresponding to nearest neighbour search. To reduce the computational cost, it is critical to design an effective similarity upper bound for quick candidate pruning to avoid full similarity computations. Here the traversing the indexing structure MIL, the maximum number of layers reached by the query tweet is processed. The time complexity of the algorithm is linear to the number of events that share at least one common word with the query. However, the proposed upper bound pruning strategy starting from lower to upper layers can greatly reduce the search cost by saving a large number of the Cosine similarity computations. As the query tweet's length increases, such reduction becomes more significant since the Cosine similarity computation gets more expensive. This will be further verified by our experiment results.

V. CONCLUSIONS

In this paper, we have presented a novel event monitoring method along with a multi-layer inverted indexing structure to efficiently and effectively index evolving events from tweet streams. Four operations are designed to capture the dynamics of events over time. In addition a novel approach have been specified to find the nearest neighbour who exist with the similar type of tweets as that of ours. With the consideration of time, the event detection performance will be further improved in both accuracy and scalability.

REFERENCES

[1] Becker, H., Naaman, M., Gravano, L. "Beyond trending topics: Real-world event identification on twitter. In" Proceedings of the 5th International AAAI Conference on Weblogs and Social Media (ICWSM), North America, pp. 438-441, July 2011.

[2] Popescu, A., Pennacchiotti, M., Paranjpe, D." Extracting events and event descriptions from Twitter. In" Proceedings of the 20th International Conference on World Wide Web (WWW), pp. 105-106. ACM (2011).

[3] J. Allan, R. Papka, and V. Lavrenko, "On-line new event detection and tracking," in SIGIR, pp. 37–45, 1998.

[4] Y. Jie, L. Andrew, C. Mark, R. Bella, and P. Robert, "Using social media to enhance emergency situation awareness," IEEE Intelligent Systems, vol. 27, no. 6, pp. 52–59, 2012.

[5] P. Lee, L. V. S. Lakshmanan, and E. E. Milios, "Incremental cluster evolution tracking from highly dynamic network data," in ICDE, pp. 3–14, 2014.

[6] A. Gruenheid, X. L. Dong, and D. Srivastava, "Incremental record linkage," PVLDB, vol. 7, no. 9, pp. 697–708, 2014.

[7] H. Abdelhaq, C. Sengstock, and M. Gertz, "Eventweet:

Online localized event detection from twitter," PVLDB, vol. 6, no. 12, pp. 1326–1329, 2013.

[8] K. Massoudi, M. Tsagkias, M. de Rijke, and W. Weerkamp, "Incorporat-ing query expansion and quality indicators in searching microblog posts," in ECIR, pp. 362–367, 2011.

[9] H. Becker, M. Naaman, and L. Gravano, "Beyond trending topics: Real-world event identification on twitter," in ICWSM, 2011.

[10] J. Weng and B.-S. Lee, "Event detection in twitter," in ICWSM, pp. 401–408, 2011.

[11] J. Zobel and A. Moffat, "Inverted files for text search engines," ACM Comput. Surv., vol. 38, no. 2, 2006.

[12] M. Chang and C. K. Poon, "Efficient phrase querying with common phrase index," in ECIR, pp. 61–71, 2006.

[13] M. Pitts, S. Savvana, S. B. Roy, and V. Mandava, "ALIAS: author disambiguation in microsoft academic search engine dataset," in EDBT, pp. 648–651, 2014,.

[14] M. Busch, K. Gade, B. Larson, P. Lok, S. Luckenbill, and J. Lin, "Earlybird: Real-time search at twitter," in ICDE, pp. 1360–1369, 2012,.

[15] C. Chen, F. Li, B. C. Ooi, and S. Wu, "Ti: an efficient indexing mechanism for real-time search on tweets," in SIGMOD, pp. 649– 660, 2011.

[16] X. Zhou and L. Chen, "Event detection over twitter social media streams," VLDB J., vol. 23, no. 3, pp. 381–400, Jun. 2014.

[17] R. McCreadie, C. Macdonald, I. Ounis, M. Osborne, and S. Petrovic, "Scalable distributed event detection for twitter," in Int. Conf. on Big Data, pp. 543–549, 2013.

[18] C. D. Manning, P. Raghavan, and H. Schutze,¨ Introduction to Information Retrieval. Cambridge University Press, 2008.

[19] S. Unankard, X. Li, and M. Sharaf, "Emerging event detection in social networks with location sensitivity," World Wide Web, pp. 1–25, 2014.

[20] M. Steinbach, G. Karypis, and V. Kumar, "A comparison of document clustering techniques," in In KDD Workshop on Text Mining, 2000.

[21] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," in SIGMOD, pp. 1–12, 2000.

[22] R. Fagin, A. Lotem, and M. Naor, "Optimal aggregation algorithms for middleware," in PODS, pp. 102–113, 2001.

[23] A. C. Awekar and N. F. Samatova, "Fast matching for all pairs similarity search." in Web Intelligence, pp. 295–300, 2009.

[24] E. Cohen, "Decay models," in Encyclopedia of Database Systems, pp. 757–761, 2009.