IJCSE International Journal of Computer Sciences and Engineering Open Access
Survey Paper Vol.-7, Special Issue, 6, March 2019 E-ISSN: 2347-2693

A Survey on Information Flow Monitoring System Using Skyline Algorithm

K.M. Jyothsna Priya[1*], A. Srinivasulu[2]

Data Analytics Research Center, Sree Vidyanikethan Engineering College, India

Corresponding Author: dsaichitravathi@gmail.com

DOI: https://doi.org/10.26438/ijcse/v7si6.18 | Available online at: www.ijcseonline.org

***Abstract*:** Social media are websites and computer programs that enable users to create and share information on the internet using a computer or a mobile phone. Large quantities of data are generated by social networks in seconds. The information which is generated in a social network is transformed into a flow by the subjects who produce, transmit, and consume it. This flow can be represented as a very complicated directional graph. In this graph each subject is represented as a node, and the flow of information is represented as a directed edge. In this paper, we introduce a method of dividing this complex directional graph by user and quantifying the flow of information between and among users based on information flow vectors. We propose a system that can monitor the flow of information in social networks using information flow vectors extracted from social media data. We also introduce an improved skyline algorithm that can respond quickly to a user's various queries.

Keywords – Information flow, Social media data, Skyline, Lambda architecture, MapReduce.

I. INTRODUCTION

Social media generates large amounts of data every day. For example, 456,000 tweets are generated daily on twitter. Such massive amounts of data from social media are measured in petabytes. Information propagation in online social networks like Twitter is unique in that word-of-mouth propagation and traditional media sources coexist. A large amount of data from twitter to compare the relative roles different types of users play in information flow. Twitter as a means to conduct research on longstanding social science research questions in a computational framework. [1]

In the field of big data, analyzing which users of social media generate what information and how broadly the data are disseminated is especially challenging because the data are generated at a high and data structures are diverse.Identifying the relationships among information bearers, producers, and consumers offers a sociological approach to analyzing the interaction patterns among social actors to elucidate social structures via instruments such as graphs.A frequent subgraph mining algorithm called FSM-H which uses an iterative MapReduce based framework. FSM-H is complete as it returns all the frequent subgraphs for a given user-defined support, and it is efficient as it applies all the optimizations that the latest FSM algorithms adopt. Experiments with real life and large synthetic datasets validate the effectiveness of FSM-H for mining frequent subgraphs from large graph datasets. [2]

The systems for tracking and monitoring the flow of information in social networks, measuring the flow of information contained in social data, and informing users of information flows. A new class of problems called network information flow which is inspired by computer network applications. Consider a point-to-point communication network on which a number of information sources are to be mulitcast to certain sets of destinations. In existing computer networks, each node functions as a switch in the sense that it either relays information from an input link to an output link, or it replicates information received from an input link and sends it to a certain set of output links. From the information-theoretic point of view, there is no reason to restrict the function of a node to that of a switch. Rather, a node can function as an encoder in the sense that it receives information from all the input links, encodes, and sends information to all the output links. [3]

Data from Twitter is used to extract and measure information flows.The first system is based on a Lambda architecture that can collect and analyze social media data in real time, including a means to quantify social media data in terms of information flow and an algorithm to extract the information flow path. Many algorithms have been proposed to solve the task.frequent itemset discovery algorithms have been used to find interesting patterns in various application areas. However, as data mining techniques are being increasingly applied to non-traditional domains, existing frequent pattern discovery approaches cannot be used. The transaction framework that is assumed by these algorithms

cannot be used to effectively model the data sets in these domains. An alternate way of modeling the objects in these data sets is to represent those using graphs. Within that model, one way of formulating the frequent pattern discovery problem is that of discovering subgraphs that occur frequently over the entire set of graphs. An efficient algorithm called FSG, for finding all frequent subgraphs in large graph data sets and experimentally evaluates the performance of FSG using a variety of real and synthetic data sets. [4]

The Lambda architecture enables the reconstruction of a bigdata system as a series of layers: the speed layer, the serving layer, and the deployment layer. Each layer is characterized by a subset of properties and is based on the functionality provided by the layers below it. The second system is a skyline algorithm used to respond quickly to various queries from system users who want to monitor information. Skyline SFS algorithm based on the pre-sorting, that is general for use with many skyline query, efficient and well behaved in a relational sorting. SFS is a realistic algorithm for implementation of skyline in realistic engines. There are numerous improvements that can be made to SFS, and pursuing better skyline algorithm based upon SFS.

II. LITERARTURE SURVEY

[1].M. Cha, F. Benevenuto, H. Haddadi, and K. Gummadi, ''the world of connections and information flow in Twitter,'' IEEE Trans. Syst., Man, Cybern. A, Syst., Humans, vol. 42, no. 4, pp. 991–998, Jul. 2012.

Information propagation in online social networks like Twitter is unique in that word-of-mouth propagation and traditional media sources coexist. Collect a large amount of data from twitter to compare the relative roles different types of users play in information flow Using empirical data on the spread of news about major international headlines as well as minor topics and investigate the relative roles of three types of information spreaders: 1) mass media sources like BBC; 2) grassroots, consisting of ordinary users; and 3) evangelists, consisting of opinion leaders, politicians, celebrities, and local businesses. Mass media sources play a vital role in reaching the majority of the audience in any major topics. Evangelists, however, introduce both major and minor topics to audiences who are further away from the core of the network and would otherwise be unreachable. Grassroots users are relatively passive in helping spread the news, although they account for the 98% of the network. Results bring insights into what contributes to rapid information propagation at different levels of topic popularity, which believe are useful to the designers of social search and recommendation engines.

The impressive growth of social networking services has made personal contacts and relationships more visible and quantifiable than ever before. These services have also become important vehicles for news and channels of influence. Twitter has emerged as a popular medium for discussing noteworthy events that are happening around the world.Twitter as a means to conduct research on longstanding social science research questions in a computational framework. Twitter as a means to conduct research on longstanding social science research questions in a computational framework. The focus on the relative roles different users play on information flow in order to understand why certain trends or news are adopted more widely than others. For the study, crawled the Twitter network and gathered all public tweets and follow links. In total, 2 billion follow relationships among 54 million users who produced a total of 1.7 billion tweets. To the best of our knowledge, is the largest data gathered and analysed from the Twitter network.

By analyzing the structure of the connection network and the distribution of links, a broad division that yields three distinct user groups based on in-degree: the extremely well-connected users with more than 100 000 followers, the least connected masses with no more than 200 followers, and the remaining well-connected small group of users. Our division of users is based on the definition of different user roles from the theory on information flow : mass media, who can reach a large audience, but do not follow others actively; grassroots, who are not followed by a large number of users, but have a huge presence in the network; and evangelists, who are socially connected and actively take part in information flow like opinion leaders.

Twitter administrators allow us to gather data from their site at scale. They graciously white-listed the IP address range containing 58 of our servers, which allow dust to gather large amounts of data. The Twitter API is to gather two pieces of information for each Twitter user: 1) profile data including information about the user's social links, i.e., other Twitter users she is following; and 2) all tweets ever posted by the user including the time when tweets were posted.

The first extensive analysis of a near complete data set obtained from the micro blogging service Twitter. Acquisition of such a rich data set enabled us to identify the relationship among distinct groups of users—mass media, evangelists, and grassroots and the roles that they play in viral spreading of political and social news messages. The connectivity trends between users differentiate Twitter, away from conventional social networks, toward a collaborative gossip and news publishing tool and makes Twitter an ideal medium for studying the relative roles these distinct user groups play. The Twitter network exhibits topological features that distinguish it from other social networks; it stands out as a broadcasting system encompassing users of vastly different abilities to propagate and receive information.

[2].M.A. Bhuiyanan M. AlHasan, "An iterative Map Reduce based frequent subgraph mining algorithm," IEEE Trans. Knowl. Data Eng., vol. 27, no. 3, pp. 608–620, Mar. 2015.

Over the years, many algorithms have been proposed to solve task. FSM- algorithms assume that the data structure of the mining task is small enough to fit in the main memory of a computer. However, as the real-world graph data grows, both in size and quantity, such an assumption does not hold any longer. To overcome the graph database-centric methods have been proposed in recent years for solving FSM; however, a distributed solution using MapReduce paradigm has not been explored extensively. Since MapReduce is becoming the de-facto paradigm for computation on massive data, an efficient FSM algorithm is a paradigm of huge demand. A frequent subgraph mining algorithm called FSM-H which uses an iterative MapReduce based framework. FSM-H is complete as it returns all the frequent subgraphs for a given user-defined support, and it is efficient as it applies all the optimizations that the latest FSM algorithms adopt. Experiments with real life and large synthetic datasets validate the effectiveness of FSM-H for mining frequent subgraphs from large graph datasets.

Solving the task of frequent subgraph mining on a distributed platform like MapReduce is challenging for various reasons. First, an FSM method computes the support of a candidate subgraph pattern over the entire set of input graphs in a graph dataset. In a distributed platform, if the input graphs are partitioned over various worker nodes, the local support of a subgraph in the respective partition at a worker node is not much useful for deciding whether the given subgraph is frequent or not. Also, local support of a subgraph in various nodes cannot be aggregated in a global data structure, because, MapReduce programming model does not provide any built-in mechanism for communicating with a global state. Also, the support computation cannot be delayed arbitrarily, as following Apriori principle future candidate frequent patterns1 can be generated only from a frequent pattern.

FSM-H is designed as an iterative MapReduce process. At the beginning of iteration i, FSM-H has at its disposal all the frequent patterns of size i-1 (Fi-1), and at the end of iteration i, it returns all the frequent patterns of size i,(Fi). The size of a graph is equal to the number ofedges it contains. For a mining task if F is the set of frequent patterns, FSM-H runs for a total of l iterations, where l is equal to the size of the largest graph in F.

The iterative Map Reduce based frequent sub graph mining algorithm, called FSMH. It shows the performance of FSM-H over real life and large synthetic datasets for various system and input configurations.

[3]. R. Ahlswede, N. CAI, S.-Y. R. Li, and R. W. Yeung, ''Network information flow,'' IEEE Trans. Inf. Theory, vol. 46, no. 4, pp. 1204–1216, Jul. 2000.

A new class of problems called network information flow which is inspired by computer network applications. Consider a point-to-point communication network on which a number of information sources are to be mulitcast to certain sets of destinations. Assume that the information sources are mutually independent. The problem is to characterize the admissible coding rate region. Result can be regarded as the Max-flow Min-cut Theorem for network information flow.

In existing computer networks, each node functions as a switch in the sense that it either relays information from an input link to an output link, or it replicates information received from an input link and sends it to a certain set of output links. From the information-theoretic point of view, there is no reason to restrict the function of a node to that of a switch. Rather, a node can function as an encoder in the sense that it receives information from all the input links, encodes, and sends information to all the output links. From the point of view, a switch is a special case of an encoder.

In the classical information theory for point-to-point communication, if two information sources are independent, optimality can be achieved (asymptotically) by coding the sources separately. The coding method is referred to as coding by superposition . If the coding method is always optimal for multisource network information flow problems, then in order to solve the problem, only need to solve the sub problems for the individual information sources separately, where each of these sub problems is a single-source problem. However, the multisource problem is not a trivial extension of the single-source problem, and it is extremely difficult in general.

A theorem which characterizes the admissible coding rate region for the single-source problem.. Result can be regarded as the Max-flow Min-cut Theorem for network information flow and discussion is based on a class of block codes called -codes. Therefore, it is possible, though not likely, that the result can be enhanced by considering more general coding schemes. Nevertheless, prove in the Appendix that probabilistic coding does not improve performance.

The problem with one information source, and have obtained a simple characterization of the admissible coding rate region. Our result can be regarded as the Max-flow Min-cut Theorem for network information flow.

[4].M. Kuramochi and G. Karypis, ''an efficient algorithm for discovering frequent subgraphs,'' IEEE Trans. Knowl. Data Eng., vol. 16, no. 9, pp. 1038–1051, Sep. 2004.

Over the years, frequent itemset discovery algorithms have been used to find interesting patterns in various application areas. However, as data mining techniques are being increasingly applied to non-traditional domains, existing frequent pattern discovery approaches cannot be used. The transaction framework that is assumed by these algorithms cannot be used to effectively model the data sets in these domains. An alternate way of modeling the objects in these data sets is to represent those using graphs. Within that model, one way of formulating the frequent pattern discovery problem is that of discovering subgraphs that occur frequently over the entire set of graphs. An efficient algorithm called FSG, for finding all frequent subgraphs in large graph data sets and experimentally evaluates the performance of FSG using a variety of real and synthetic data sets. Results show that despite the underlying complexity associated with frequent subgraph discovery, FSG is effective in finding all frequently occurring subgraphs in data sets containing more than 200,000 graph transactions and scales linearly with respect to the size of the data set.

Developing algorithms that discover all frequently occurring subgraphs in a large graph data set is particularly challenging and computationally intensive, as graph and subgraph isomorphism's play a key role throughout the computations. A new algorithm, called FSG, for finding all connected subgraphs that appear frequently in a large graph data set and finds frequent sub graphs using the level-by-level expansion strategy adopted by Apriori.

There are two key aspects in the above problem statement. Motivated by the fact that the resulting frequent subgraphs will be encapsulating relations (or edges) between some of the entities (or vertices) of various objects. Within the context, connectivity is a natural property of frequent patterns. An additional benefit of the restriction is that it reduces the complexity of the problem, as need not to consider disconnected combinations of frequent connected subgraphs. Second, allows the graphs to be labelled, and discovered frequent patterns can contain multiple vertices and edges carrying the same label. It greatly increases modeling ability, as it allows us to find a pattern involving multiple occurrences of the same entities and relations, but at the same time makes the problem of finding such frequently occurring subgraphs nontrivial. Due to such cases, any frequent subgraph discovery algorithm needs to correctly identify how a particular subgraph maps to the vertices and edges of each graph transaction, that can only be done by solving many instances of the subgraph isomorphism problem, which has been shown to be in NP-complete .

The FSG algorithm for finding frequently occurring subgraphs in large graph data sets that can be used to discover recurrent patterns in scientific, spatial, and relational data sets. Such patterns can play an important role for understanding the nature of these data sets and can be used as input to other data-mining tasks detailed experimental evaluation shows that FSG can scale reasonably well to very large graph data sets provided that the graphs contain a sufficiently many different labels of edges and vertices. Key elements to FSG's computational scalability are the highly efficient canonical labelling algorithm and candidate generation scheme and its use of a TID list-based approach for frequency counting. These three features combined allow FSG to uniquely identify the various generated subgraphs, generate candidate patterns with limited degree of redundancy, and to quickly prune most of the infrequent subgraphs without having to resort to computationally expensive graph and subgraph isomorphism computations. Furthermore, presented and evaluated a database partitioning-based approach that substantially reduces FSG's memory requirement for storing TID lists with only a moderate increase in runtime.

[5].X. Yan and J. Han, ''span: Graph-based substructure pattern mining,'' in Proc. IEEE Int. Conf. Data Mining, Dec. 2002, pp. 721–724.

The new approach for frequent graph-based pattern mining in graph datasets and propose a novel a algorithm called gSpan (graph-based, substructure pattern mining), which discovers frequent substructures without candidate generation & a builds a new lexicographic order among graphs, and maps each graph to a unique mini- mum DFS code as its canonical label. Based on lexicographic order & adopts the depth-first search strategy to mine efficiently. Performance study shows that gSpan substantially outperforms previous algorithm, sometimes by an order of magnitude.

Frequent substructure pattern mining has been an emerging data mining problem with many scientific and com- metrical applications.As a general data structure, labelled graph can be used to model much complicated sub- structure patterns among data. Two techniques, DFS lexicographic order and minimum DFS code, are introduced here, which form a novel canonical labelling system to support DFS search. GSpan discovers all the frequent subgraphs without candidate generation and false positives pruning. It combines the growing and checking of frequent subgraphs into one procedure, thus accelerates the mining process.

The gSpan uses a sparse adjacency list representation to store graphs. Subgraph mining stops searching either when the support of a graph is less than minSup, or its code is not a minimum code, which means the graph and all its descendants have been generated and discovered before.

A new lexicographic ordering system and developed a depth-first search-based mining algorithm gSpan for efficient mining of frequent subgraphs in large graph database. Performance study shows that gSpan outperforms

FSG by an order of magnitude and is capable to mine large frequent subgraphs in a bigger graph set with lower minimum supports than previous studies.

The problem with one information source, and have obtained a simple characterization of the admissible coding rate region.Result can be regarded as the Max-flow Min-cut Theorem for network information flow.

2.1 OPEN ISSUES OF LITERATURE SURVEY

1. Frequent sub graph mining algorithm provides a distributed solution using MapReduce paradigm has not been explored extensively.Since MapReduce is becoming the de-facto paradigm for computation on massive data.
2. Frequent item set discovery algorithms providespartitioning-based approach that substantially reduces FSG's memory requirement for storing lists with only a moderate increase in runtime but not high increase in runtime.
3. The connectivity trends between users differentiate Twitter, away from conventional social networks; toward a collaborative gossip.This makes twitter an ideal medium for studying the relative roles these distinct user groups play.
4. Frequent substructure pattern mining has been an emerging data mining problem with many scientific and com- metrical applications and does not model complicated sub- structure patterns among data

III. METHODOLOGY

Analyzing real-time data is an important aspect of analyzing big data. Because social media data is generated in real time and the volume of generated data is large, an alternative to the general method of analyzing big data is needed .The Lambda architecture is a structured methodology for merging results that includes newly generated real-time data.The Lambda architecture consists of three layers: batch, serving, and speed. The batch layer combines the collected data and analyzes big data with Map Reduce. The speed layer analyzes the real-time data generated during the analysis time consumed by the batch layer. The serving layer generates a view that was merged from the results of the analyses in the batch layer and the speed layer and provides the analysis service to a user's query.The proposed method is implemented on the Map Reduce platform, which is suitable for processing big data. This platform executes of two consecutive functions: first, it identifies meaningful IFVs between two users from their Twitter messages, and second, it iteratively merges the IFVs generated in the first function.

The speed layer executes a spark streaming function to analyze data generated in real time. The spark streaming function is identical to the Map Reduce algorithm described above. In order to merge the results from the batch layer and the results from the speed layer in the serving layer, the analysis from the batch layer must be checked to determine if it was applied to data generated in real time, and any results that do not overlap with the analysis result from the speed layer must be merged. To perform this check, generate a hash key for each item of data so that the results of the analyses from the batch layer and the speed layer can be merged without duplication. Just as Hadoop's practitioner uses the md5sum hash function to distinguish keys, use the hash function to create a unique hash key that takes the user and date values of the tweet data as input.

The results of the speed and batch layers must be summarized for the servicing layer and prepared for the user's view. Therefore, it is important to be able to efficiently navigate through vast amount of high-dimensional data and present results in aa comprehensible way. This can be achieved using various algorithms, and because architecture relies on saved data, and propose using the skyline algorithm to generate the user's view based on the information the user considers important .Objective, need to get top k tuples from a data stored from the batch and serving layers which have been pre-processed and saved into a database from which will receive data.

skyline points here represent data points that have values among all dimensions that are equal to or greater than the corresponding values of other points in the dataset and therefore are not dominated. Therefore, in the context of social data, the higher a user's values in multiple dimensions, the more important is in the social network. A naïve method of skyline computation is using nested loops, whereby each dimension is scanned and each data point is compared with all remaining data points. This approach is inefficient, and using a skyline window to compute skyline points has higher efficiency.

The sort-filter-skyline is another algorithm that applies a monotonic function to sort tuples in relation to each other and then minimizing comparison operations by windowing dominant skyline points together early in the process There are many other more sophisticated methods for skyline computations that have been proposed recently; however, use these algorithms in the experiments to enable comparability with previously published work. In addition, the experiments described in the subsequent section showed that the sort-filter-skyline is the fastest algorithm. The sort-filter-skyline approach requires data to be sorted by a monotonic function, and that pre-ordered dataset is then used in the algorithm. Data points are taken one by one from the pre-ordered dataset. The first data point is added to the window because there is no data in the skyline window.

For each subsequent data point in the pre-ordered dataset, a comparison operation is performed to reveal if such data point is dominated, in which case it is not added to the window, whereas if it is not dominated, it is added as a

skyline point. The size of the skyline window can be varied: if the window size is relatively bigger than the number of skyline points in the dataset, then one pass of the whole dataset will not fill the skyline window and the algorithm will terminate by outputting the contents of the skyline. In the opposite case, if the skyline window is filled before the rest of the files are loaded to the skyline, and then the algorithm will output the contents of the skyline window. The algorithm terminates when all data points have been visited. The major advantage to this approach is that a maximum number of data points are discarded and fewer data points need to be compared with each other.

IV. IMPLEMENTATION

4.1 LAMBDA ARCHITECTURE:

Analyzing real-time data is an important aspect of analyzing big data. Because social media data is generated in real time and the volume of generated data is large, an alternative to the general method of analyzing big data is needed .The Lambda architecture is a structured methodology for merging results that includes newly generated real-time data.The Lambda architecture consists of three layers: batch, serving, and speed. The batch layer combines the collected data and analyzes big data with Map Reduce. The speed layer analyzes the real-time data generated during the analysis time consumed by the batch layer. The serving layer generates a view that was merged from the results of the analyses in the batch layer and the speed layer and provides the analysis service to a user's query.

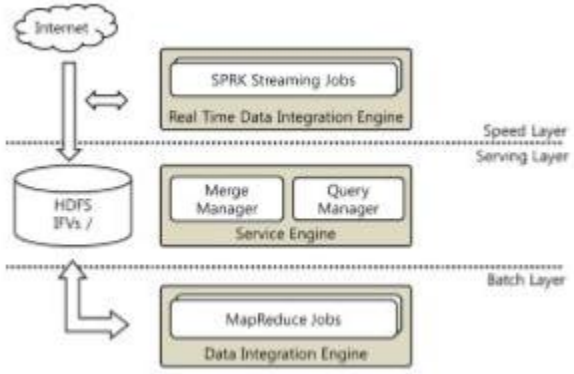

Fig: Proposed Architecture

The proposed method is implemented on the MapReduce platform, which is suitable for processing bigdata. This platform executes of two consecutive functions: first, it identifies meaningful IFVs between two users from their Twitter messages, and second, it iteratively merges the IFVs generated in the first function.Algorithm1 is implemented in MapReduce to extract1-lengthIFVs.The mapper class of algorithm 1 extracts all the users included in the given tweet data, and the reduce class outputs the IFV that satisfies the threshold among the 1-length IFV extracted from the mapper class as the final result. Algorithm 2 is the main algorithm that finds and merges matching IFVs among the (n−1)-length IFVs extracted from the previous stage. The mapper class of algorithm 2 distinguishes (n-1)-length IFV by key and value. For example, if the 2-length IFV consisting of a, b, and c is an input to the mapper class, then it passes the intermediate result, with ab and bc as keys, to the reduce class. The reduce class outputs the 3-length IFV by concatenating the extracts from different positions using delimiters to see where the intermediate results are extracted.

The speed layer executes a spark streaming function to analyze data generated in real-time. In order to merge the results from the batch layer and the results from the speed layer in the serving layer, the analysis from the batch layer must be checked to determine if it was applied to data generated in real time, and any results that do not overlap with the analysis result from the speed layer must be merged. To perform this check, generate a hash key for each item of data so that the results of the analyses from the batch layer and the speed layer can be merged without duplication. Just as Hadoop's partitioner uses the hash function to distinguish keys, use the hash function to create a unique hash key that the user and date values of the tweet data as input.

ALGORITHM 1:

1: class MAPPER
2: method MAP(docid a, DB D)
3: d ←D.Tweet
4: for all term t ∈Tweet d do
5: [u1,u2,...]=Extract user(t)
6: if (Retweet)then
7: EMIT(pair(u2, u1), count 1)
8: else
9: EMIT(pair(u1, u2), count 1)
1: class REDUCER
2: method REDUCE(pair(u1, u2), counts [c1,c2,...])
3: sum←0
4: for all count c∈counts [c1,c2,...] do
5: sum←sum+c
6: if sum > support.THRESHOLD then
7: EMIT(pair(u1, u2), count sum)

ALGORITHM 2:

1: class MAPPER
2: methodMAP(Intermediate DB D)
3: [u1,u2]←D.directIFV
4: EMIT(u2, pair('A'+k,u1, u2))
5: EMIT(u1, pair('B'+k,u1, u2))
1: class REDUCER
2: method REDUCE(uk, pair[(k1, u1, u2), (k2, u1, u2),...])
3: LA ← new LISTARRAY, LB ← new LISTARRAY
4: for all pair(k,u1,u2) ∈ pairs [(k1,u1,u2),(k2,u1,u2),...] do
5: if (STARTWITH(k)='A')then
6: LA ←pair (u1,u2)
7: else if(STARTWITH(k)='B')then

8: LB ←pair (u1,u2)
9: if (LA 6=EMPTY) AND (LB 6=EMPTY) then
10: for all pair(u1.A,u2.A) ∈LA do
11: for all pair(u1.B,u2.B) ∈LB do
12: EMIT(pair(u1.A,u2.A,u2.B), pair(k.B,t.FW))

4.2 SKYLINE ALGORITHM:

The results of the speed and batch layers must be summarized for the servicing layer and prepared for the user's view. Therefore, it is important to be able to efficiently navigate through vast amount of high-dimensional data and present results in aa comprehensible way. This can be achieved using various algorithms, and because architecture relies on saved data, propose using the skyline algorithm to generate the user's view based on the information the user considers important .Objective, will need to get top k tuples from a data stored from the batch and serving layers which have been pre-processed and saved into a database from which will receive data. Skyline points here represent data points that have values among all dimensions that are equal to or greater than the corresponding values of other points in the dataset and therefore are not dominated. Therefore, in the context of social data, the higher a user's values in multiple dimensions, the more important in the social network. A naïve method of skyline computation is using nested loops, whereby each dimension is scanned and each data point is compared with all remaining data points. This approach is inefficient, and using a skyline window to compute skyline points has higher efficiency.

The sort-filter-skyline is another algorithm that applies a monotonic function to sort tuples in relation to each other and then minimizing comparison operations by windowing dominant skyline points together early in the process There are many other more sophisticated methods for skyline computations that have been proposed recently; however, use these algorithms in experiments to enable comparability with previously published work. In addition, the experiments described in the subsequent section showed that the sort-filter-skyline is the fastest algorithm. The sort-filter-skyline approach requires data to be sorted by a monotonic function, and that pre-ordered dataset is then used in the algorithm. Data points are taken one by one from the pre-ordered dataset. The first data point is added to the window because there is no data in the skyline window.

For each subsequent data point in the pre-ordered dataset, a comparison operation is performed to reveal if such data point is dominated, in which case it is not added to the window, whereas if it is not dominated, it is added as a skyline point. The size of the skyline window can be varied: if the window size is relatively bigger than the number of skyline points in the dataset, then one pass of the whole dataset will not fill the skyline window and the algorithm will terminate by outputting the contents of the skyline. In the opposite case, if the skyline window is filled before the rest of the files are loaded to the skyline, then the algorithm will output the contents of the skyline window. The algorithm terminates when all data points have been visited. The major advantage to this approach is that a maximum number of data points are discarded and fewer data points need to be compared with each other.

ALGORITHM:

1. Input: Result of speed and batch layers in the form of sorted dataset D.
2. Output: A set of skyline points of dataset D.
3. not completed = true
4. I = getiterator (heap)
5. not completed = false
6. while (has_data(I,d)) do
7. if ("I is not dominated") then
8. if("skyline window is full") then
9. not completed=true
10. break
11. else
"add d to skyline"
12. if(not completed) then
13. F=open_new_file_(next_iteration)
14. write (F,d)
15.while(has_data(F,d)) do
16. if("d not dominated") then
17. write(F,d)
18. Iterator=next_iteration
19." send skyline_window data to output"

V. ANALYSIS OF METHODOLOGY

The proposed method factored out the quantifiable attributes from the dataset of the batch and speed processes, which resulted in five attributes or dimensions that were taken as the basis of calculations, with approximately 150K tuples. Experiments were done on algorithms including naïve, top k, block-nested-loops (BNL), and sortfilter-skyline (SFS), and they were implemented in Java.Based on the experimental results from the batch and speed layers of Lambda-based architecture system, the SFS method performed best and required the least time to compute the skyline. The BNL algorithm was the next most efficient for higher dimensions; the BNL algorithm also outperformed the SFS in some dimensions because the former algorithm does not require data to be ordered by a monotonic function as does SFS. Therefore, SFS is slightly limited in lower dimensions due to the extra computational time required for monotone scoring function and sorting algorithm. However, with more dimensions and tuples added into the computation, SFS performs faster than other algorithms, as was expected. The naïve method was the slowest, which was also expected because the naïve method compares each tuple with each other tuple without maintaining the candidate window in main memory like BNL or SFS.

Overall, skyline computations implemented here in the batch and layer system, are useful tools in the analysis of social data. Skyline algorithms are relatively fast and can efficiently service system users by providing them necessary information in the form of nodes defined to be the most active and influential in social web graph. Therefore, in conjunction with a skyline algorithm, particularly SFS algorithm, the proposed system provides an important tool for data analysis of social media, which is an emerging need for many practitioners in the field.

VI. CONCLUSION AND FUTURE WORK

The paper proposes a system that can collect real-time social media data and monitor the flow of information distributed in the social network. The proposed system is based on a Lambda architecture that includes measures to quantify the flow of information circulated in social networks and enhance the real-time analysis capability of big data. Lambda architecture can collect and analyze social media data in real time, including a means to quantify social media data in terms of information flow and an algorithm to extract the information flow path. Data processing architecture designed to handle big data using both batch and stream processing methods. The sort-filter-skyline approach requires data to be sorted by a monotonic function, and that pre-ordered dataset is then used in the algorithm. Data points are taken one by one from the pre-ordered dataset. The first data point is added to the window because there is no data in the skyline window. For each subsequent data point in the pre-ordered dataset, a comparison operation is performed to reveal if such data point is dominated, in which case it is not added to the window, whereas if it is not dominated, it is added as a skyline point. The size of the skyline window can be varied: if the window size is relatively bigger than the number of skyline points in the dataset, then one pass of the whole dataset will not fill the skyline window and the algorithm will terminate by outputting the contents of the skyline. In the opposite case, if the skyline window is filled before the rest of the files are loaded to the skyline, then the algorithm will output the contents of the skyline window. The algorithm terminates when all data points have been visited. The major advantage is that a maximum number of data points are discarded and fewer data points need to be compared with each other. Skyline algorithm based on the pre-sorting, that is general for use with many skyline queries, efficient and well behaved in a relational sorting. SFS is a realistic algorithm for implementation of skyline in realistic engines. In addition, an improved skyline algorithm to enable quick responses to user queries, which an important function of the system. The performance of the proposed system is validated experimentally. The above method works by monitoring the flow of information in social networks using information flow vectors extracted from social media data .In future the solution should be implemented with more accuracy and the time complexity of skyline algorithm in existing method is high.

REFERENCES

[1] T. Hale. How Much Data Does the World Generate Every Minute? Accessed: Dec. 22, 2017.

[2] D. Boyd and K. Crawford, ‘‘Critical questions for big data: Provocations for a cultural, technological, and scholarly phenomenon,’’ Inf.,Commun . Soc., vol. 15, no. 5, pp. 662–679, 2012.

[3] L. Palen and S. Vieweg, ‘‘the emergence of online widescale interaction in unexpected Events: Assistance, alliance & retreat,’’ in Proc. ACM Conf. Comput. Supported Cooperat.Work. 2008, pp. 117–126.

[4] M. Taddicken, ‘‘the people’s choice: How the voter makes up his mind in a presidential campaign,’’ in Schlüsselwerke der Medienwirkungsforschung. Wiesbaden, Germany: Springer, 2016, pp. 25–36.

[5] M. Cha, F. Benevenuto, H. Haddadi, and K. Gummadi,‘‘The world of connection and Information flow in twitter’’ IEEE Trans. Syst., Man, Cybern.A, Syst., Humans, vol. 42, no.4pp.991-1998.

[6] N. Marz and J. Warren, Big Data: Principles and Best Practices of Scalable realtime data systems. Shelter Island, NY, USA: Manning Publications, 2015.

[7] J. Scott, Social Network Analysis. Thousand Oaks, CA, USA: Sage, 2017.

[8] M. Kuramochi and G. Karypis, ‘‘An efficient algorithm for discovering frequent subgraphs,’’ IEEE Trans. Knowl. Data Eng., vol. 16, no. 9, pp. 1038–1051, Sep. 2004.

[9] X. Yan and J. Han, ‘‘gSpan: Graph-based substructure pattern mining,’’ in Proc. IEEE Int. Conf. Data Mining, Dec. 2002, pp. 721–724.

[10] L. B. Holder, D. J. Cook, and S. Djoko, ‘‘Substructure discovery in the SUBDUE system,’’ in Proc. KDD Workshop, 1994, pp. 169–180.

[11] F. Ramsey and D. Schafer, The Statistical Sleuth: A Course in Methods of Data Analysis. Boston, MA, USA: Cengage Learning, 2012.

[12] T. Lappas, K. Liu, and E. Terzi, ‘‘Finding a team of experts in social networks,’’ in Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2009, pp. 467–476.

[13] J. Xu and H. Chen, ‘‘Criminal network analysis and visualization,’’ Commun.ACM, vol. 48, no. 6, pp. 100–107, 2005.

[14] M. A. Bhuiyan and M. Al Hasan, ‘‘An iterative MapReduce based frequent subgraph mining algorithm,’’ IEEE Trans. Knowl. Data Eng., vol. 27, no. 3, pp. 608–620, Mar. 2015.

[15] A. Cuzzocrea, F. Jiang, and C. K. Leung, ‘‘Frequent subgraph mining from streams of linked graph structured data,’’ in Proc. EDBT/ICDT Workshops, 2015, pp. 237–244.

[16] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, ‘‘Network information flow,’’ IEEE Trans. Inf. Theory, vol. 46, no. 4, pp. 1204–1216, Jul. 2000.

[17] S. Borzsony, D. Kossmann, and K. Stocker, ‘‘The Skyline operator,’’ in Proc. 17th Int. Conf. Data Eng., 2001, pp. 421–430.

[18] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, ‘‘Skyline with presorting,’’ in Proc. 19th Int. Conf. Data Eng., 2003, pp. 717–719.

[19] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, ‘‘Skyline with presorting: Theory and optimizations,’’ in Proc. Intell. Inf. Process. Web Mining, 2005, pp. 595–604.

[20] I. Bartolini, P. Ciaccia, and M. Patella, ‘‘SaLSa: Computing the skyline without scanning the whole sky,’’ in Proc. 15th ACM Int. Conf. Inf. Knowl. Manage., 2006, pp. 405–414.

[21] A. Vlachou, C. Doulkeridis, and Y. Kotidis, ‘‘Angle-based space partitioning for efficient parallel skyline computation,’’ in Proc. Int. Conf. Manage. Data, 2008, pp. 227–238. 23826 VOLUME 6, 2018.

