International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2015): 78.96 | Impact Factor (2015): 6.391

Live Data Stream Classification for Reducing Query Processing Time: Design and Analysis

Spraha Kamriya[1], Vandana Kate[2]

[1]Research Scholar, Department of Computer Science and Engineering, Acropolis institute of technology and Research, Indore, India

[2]Professor, Department of Computer Science and Engineering, Acropolis institute of technology and Research, Indore, India

Abstract: ***The problem of data analysis and making decisions are increases with the volume of data. In other words processing of large data requires large resources to process and providing the final response. The big data is environment which is used for the large data processing and their analytics. But when the traffic is high and block size of data is larger than the query response is generated with the significant amount of delay. In order to optimize the delayed response need to make some effort for improving the performance of the big data systems. In this paper we proposed a new approach for solving this delayed data response based on streamed data mining. The proposed approach contributes for demonstration of the live twitter stream gathering, pre-processing of data and transformation of the unstructured data into the structured data features, classification of data streams using the group learning concept for streamed text data. This approach improves the query processing time and produces response in less time even when a single pattern is appeared for the query processing***.

Keywords: Data Streaming; Twitter; Big Data; Hadoop; C4.5 Decision Tree; OVA; MapReduce

1. Introduction

Data streams are ordered and probably unbounded sequences of data points generated by a typically non-stationary data generating process. Common data mining tasks associated with data streams include clustering, classification and frequent pattern mining. New algorithms for these types of data are proposed repeatedly and it is important to figure out them efficiently under standardized conditions.

A. Streaming Data

Streaming Data is data that is generated continuously by thousands of data sources, which typically send in the data records simultaneously, and in small sizes (order of Kilobytes). Streaming data incorporate a wide variety of data such as log files generated by customers using your mobile or web applications, ecommerce purchases, in-game player activity, information from social networks, financial trading floors, or geospatial services, and telemetry from connected devices or instrumentation in data centers [1].This data needs to be processed sequentially and incrementally on a record-by-
record basis or over sliding time windows, and used for a wide variety of analytics including correlations, aggregations, filtering, and sampling. Information derived from such analysis gives companies visibility into many aspects of their business and customer activity such as –service usage (for metering/billing), server activity, website clicks, and geo-location of devices, people, and physical goods –and enables them to respond promptly to emerging situations. For example, businesses can track changes in public sentiment on their brands and products by continuously analyzing social media streams, and respond in a timely fashion as the necessity arises [4].

B. The Data Stream Model

In the data stream model, data are arrive one or more Continuous data stream it can not access from disk or memory.Data streams differ from the conventional stored relation model in several ways [2]:

- The data elements in the stream arrive online.
- The system has de control over the procedure in which data elements comes to be processed, either within a data stream or across data streams.
- Data streams are potentially unbounded in size.
- Once an element from a data stream has been processed it is discarded or archived — it cannot be retrieved easilyunless it is explicitly stored in memory, which typically is small relative to the size of the data streams.

Operating in the data stream model does not preclude the presence of some data in conventional stored relations. Often, data stream queries may perform joins between data streams and stored relational data. For the purposes of this paper, we will assume that if stored relations are used, their contents live static. Thus, we preclude any potential transaction-processing issues that might arise from the presence of updates to stored relations that occur concurrently with data stream processing.

The paper is composed as takes after. Section II quickly depicts the related work of Data Streaming Classification. In Section III, we give a depiction of live stream Data Classification Technique using binary classifier. Result summary are introduced in Section IV. At last, Sections V talk about our conclusion and future works.

2. Literature Survey

This section provides the listing of different research efforts that are made in order to enhance query processing time for streaming data, whereas different methodologies have been adopted for significant literature.

In this paper, Haojun Liao et al. [5] present an approach to construct a built-in block-based hierarchical index structures, like Rtree, to organize data sets in one, two, or higher dimensional space and improve the query performance towards the common query types (e.g., point query, range query) on Hadoop distributed file system (HDFS). The query response time for data sets that are stored in HDFS can be significantly reduced by avoiding exhaustive search on the corresponding data sets in the presence of index structures. The basic idea is to adopt the conventional hierarchical structure to HDFS and several issues, including index organization, index node size, buffer management, and data transfer protocol, are considered to reduce the query response time and data transfer overhead through network. Experimental evaluation demonstrates that the built-in index structure can efficiently improve query performance, and serve as cornerstones for structured or semi-structured data management.

In data mining process and spatial and multimedia databases, a effective tool is the kNN join, which is use to provide the k nearest neighbors (NN), from a dataset S, of every object in a dataset R. Since it influence both the join and the NN search, performing kNN joins well is a demanding task.Data which is store in map reduce cluster kNN joins run fluently on it, this is an interesting problem. In this work, ***Chi Zhang et al. [6]*** proposes novel (exact and approximate) algorithms in MapReduce to perform efficient parallel kNN joins on large data. Authors demonstrate our ideas using Hadoop. Experiments shows onLarge and Real Data Set with millions of records in both data Sets up to 30 dimensions, have find the efficiency, effectiveness, and scalability of proposed methods.

k nearest neighbor join (kNN join), designed to find k nearest neighbors from a dataset S for every object in another dataset R, is a primitive operation widely adopted by many data mining applications. As a combination of the k nearest neighbor query and the joint operation, kNN join is an expensive operation. Given the increasing volume of data, it is difficult to perform a kNN join on a centralized machine efficiently. In this paper, ***Wei Lu et al. [7]***kNN join withMapReduce is a well-accepted framework for data-intensive applications over the clusters of systems. In brief, the mappers cluster objects into groups; the reducers perform the kNN join on each group of objects separately. Authors design an efficient mapping mechanism that adventure pruning rules for distance filtering, and overcome both the shuffling and computational costs. To reduce the shuffling cost, they propose two approximate algorithms to minimize the number of replicas. Large-scale experiments on our in-house cluster expose that our proposed methods are efficient, robust and scalable.

This paper describes the basic processing model and architecture of Aurora, a new system to manage data streams for monitoring applications. Monitoring applications differ substantially from conventional business data processing. The fact that a software system must process and revert to continual inputs from many sources (e.g., sensors) rather than from human operators requires one to rethink the structural architecture of a DBMS for this application. In this paper, ***D. J. Abadi et al. [8]*** present Aurora, a new DBMS currently under construction at Brandeis University, Brown University, and M.I.T. They first provide an overview of the basic Aurora model and architecture and then describe in detail a stream-oriented set of operators.

This paper presents the design of a read-optimized relational DBMS that contrasts sharply with most current systems, which are write-optimized. Among the many discrepancy in its design are: storage of data by column rather than by row, careful coding and packing of objects into storage including main memory during query processing, storing an overlapping collection of column oriented projections, rather than the current fare of tables and indexes, a non-traditional implementation of transactions which includes high availability and snapshot segregation for read-only transactions, and the extensive use of bitmap indexes to complement B-tree structures. Mike ***Stonebraker et al [9]*** present preliminary performance data on a subset of TPC-H and show that the system we are building, C-Store, is substantially faster than popular commercial products. Hence, the architecture looks very encouraging.

3. Proposed Work

The proposed methodologies high level conceptual model is demonstrated using the fig 1. The entire data model and their processes are defined using four stage process:

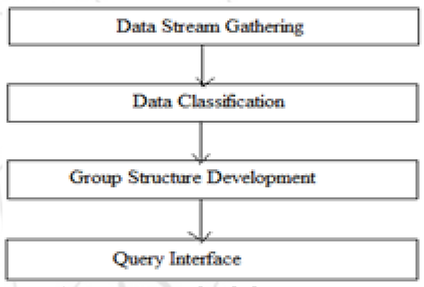

Figure 1: Methodology Layers

A. Data Stream Gethering

The data mining is always deals with the different kinds ofdata and their formats, in addition of that there are a number of different kinds of data formats are present. According to the current context we differentiate the data formats in two major categories, static data and dynamic data. The static data are those which are available in a specific volume and that are not increases in timed manner. On the other hand the dynamic data are those which are continuously increases with the time. According to literature such kind of data is also named as time series data or streamed data. In order to gather the data from the live streams some additional process is followed. Additionally the twitter is assumed as the live stream source of data and to gather the data from this data source the techniqueis presented using fig2.

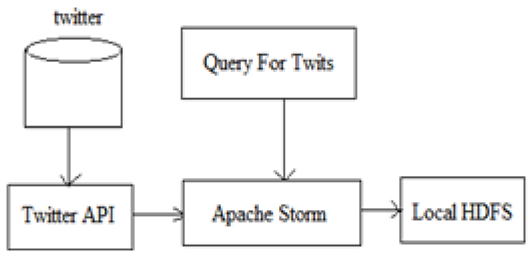

Figure 2: Data Gathering System

According to the described figure twitter is assumed as the stream data source. In order to get the data directly from the twitter the twitter provides the query API, which first authenticate the user for their account access and then the apache strom infrastructure is configured with the local Hadoop storage. That provides the ability to make query over the twitter account for gathering the twits from the twitter server. The query response generated from the server is temporarily stored on a file of HDFS directory. That is further used with the classification purpose.

B. Data Classification

The data classification is a technique by which the pattern of the data is identified to club the similar pattern data. Therefore the collected data need to be process and obtain some features by which the actual group of data can identifiable. But any classifier can directly works on the static data for making more accurate analysis. Therefore here need to change the process of classification according to data source interface. Before discussion about the changes made on the classification strategy need to discuss about the classifier model. In this experiment the C4.5 decision tree algorithm is utilized for classification. The classifier steps are discussed using table 1.

Table 1: C4.5 Algorithm

```
Input: A data set (D)
Output: A decision tree say T constructed
Process:
  1. A node (X) is created;
  2. If instance in same class.
  3. Make node (X) as the leaf node and assign a label CLASS C;
  4. Check IF the attribute list is empty, THEN
  5. Make node(X) a leaf node and assign a label of most customary CLASS;
  6. Now choose an attribute which has highest information gain from the provided attribute List, and then marked as the test_attribute;
  7. Confirming X in the role of the test_attribute;
  8. In order to have a recognized value for every test_attribute for dividing the samples;
  9. Generating a fresh twig of tree that is suitable for test_attribute = att_i from node X;
  10. Take an assumption that Bi is a group of test_attribute= att_i in the samples;
  11. If Bi is NULL, THEN
  12. Next, add a new leaf node, with label of the most general class;
  13. ELSE a leaf node is going to be added and returned by the Generate_decision_tree;
```

Now this data set is works on the pure structured format thus the unstructured data need to be modifying for adopting the algorithm. The processing of the data in unstructured to structured format conversion is defined using fig 3.

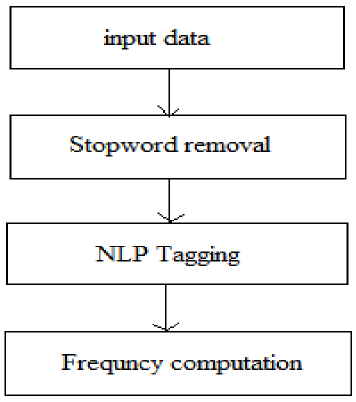

Figure 4: Data Parsing

The collected data is produced as input to the system. That data is in unstructured format and may be noisy for processing with some algorithm. Therefore in first the stop words from the entire text are removed. The stop words are frequently occurred in the sentences and not much impacting to discovery of any subject or domain i.e. is, am, are, this, the, so on. In order to remove them the following step process is used.

Table 2: Stop Word Removal

```
Input: collected data D, Stop word list $S_w$
Output: refined data R
Process:
  1. $for(i = 0; i \leq S_w.length(\ \ ); i++)$
  2. $R = FindAndReplace(D, S_w(i),)$
  3. $endfor$
  4. Return R
```

After removing the stop words from the data now need to tag the data using the NLP parser. The NLP parser extracts the part of speech information from the data. This part of speech data is used with the next phase for extracting the features form the data. In this context the frequency of the each part of speech information is computed and listed with a data table. The example of the extracted feature is given using table 3.

Table 3: Example Features

Noun	Pronoun	Verb	Adverb
2	2	1	1
2	3	1	4

The computed features can be used with the decision tree make tree data structure for finding the suitable group of data. So we take reference from the previous made statement worked with the binary classification algorithm, thus the between two different classes a new instance of the classifier is introduced. The process of stream data classification is given using table4.

Table 4: Classification of Samples

```
Input: classifier C, Pattern data D
Output: grouped data G
Process:
  1. While D != null
  2. Fetch pattern $D_i$
  3. If new class appeared
```

```
        a. If count = = 0
            i. Initialize the classifier instance $C_i$
            ii. Classify $D_i$ using $C_i$
            iii. Update count = count +1
        b. Elseif count = = 1
            i. Classify $D_i$ using $C_i$
            ii. Update count = count +1
        c. End if
    4. G = $C_i$
    5. End if
    6. i = i + 1
    7. End while
    8. Return G
```

After completing the classification of data the numbers of different groups is prepared these groups are used in further processes.

C. Group Structure Development

After the classification of the data, the different groups of data are preserved in a different big data hive structure. Therefore the similar group of data is stored in different structures of the hive. This hive data is used for making query for finding the required data.

D. Query Interface

The query interface is used to provide input the query keywords to the system and system works to find the required records from the data base using big data query. Based on the response generated the performance of the system is computed and stored in data base for further results analysis.

4. Result Analysis

a) Accuracy

In a classification technique the accuracy is measurement of accurately classified patterns over the total input patterns produced for classification result. Therefore that can be a measurement of successful training of the classification method. The accuracy of the classifier can be evaluated using the following formula:

$$\text{Accuracy} = \frac{\text{Total correctly classified patterns}}{\text{Total input patterns to Classify}} X100$$

The accuracy of the implemented proposed algorithm of Live Data Stream Classification is characterized using figure 4. The given fig 4 shows the accuracy of the implemented method. X- Axis of the figure show the number of runs of the algorithm during classification and tagging and Y axis shows the obtained performance in terms of accuracy percentage.To demonstrate the performance of both the proposed data classification technique and traditional decision tree C4.5, the blue line is used for proposed model and red line shows the performance of traditional C 4.5. According to the obtained results the performance of the proposed model provides more accurate results. Additionally the accuracy of the Data classification model is showing the consistent result while we varying number of runs.

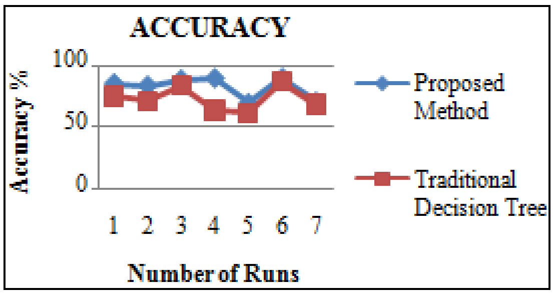

Figure 4: Accuracy

b) Error Rate

The amount of data misclassified samples during classification of algorithms is known as error rate of the system. That can also be computed using the following formula.

$$\text{Error Rate } \% = \frac{\text{Total Misclassified Patterns}}{\text{Total Input Patterns}} X100$$

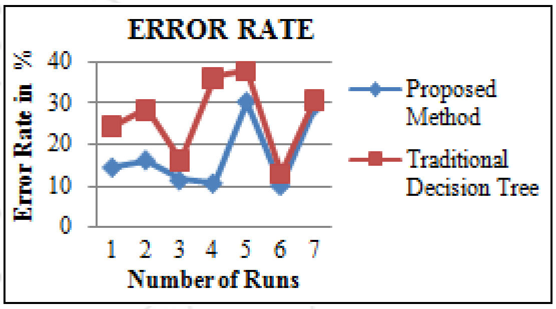

Figure 5: Error Rate

The fig 5 shows the comparative error rate of implemented classification algorithm. In order to show the performance of the system the X axis contains the number of experiments for tagging and the Y axis shows the performance in terms of error rate percentage. The error rate of the traditional C 4.5 is given using the red line and the performance of the proposed classification method is given using the blue line. The performance of the proposed classification is effective and efficient during different execution and reducing with the amount of data increases. Thus the given classifier is more efficient and accurate than the traditional C4.5 of data classification.

c) Memory Consumption

Memory consumption of the system also termed as the space complexity in terms of algorithm performance. That can be calculated using the following formula:

$$\text{Memory Consumption} = \text{Total Memory} - \text{Free Memory}$$

The amount of memory consumption depends on the amount of data reside in the main memory, therefore that affect the computational cost of an algorithm execution. The performance of the implemented classifier for Data Streaming is given using fig 6 to show the diagrammatically result the performance the X axis demonstrate the Number of experiments of the algorithm and the Y axis shows the respective memory consumption during execution in terms of kilobytes (KB). According to given demonstration, memory of the both implemented result memory showing the complexity of the system and traditional C4.5 consuming less space as compared to proposed algorithm of data stream classification.

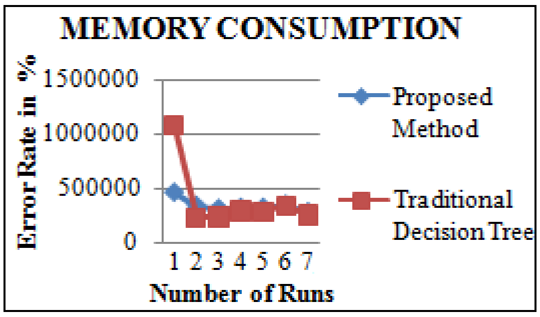

Figure 6: Memory Consumption

d) Time Computation

The amount of time required to classify the entire live stream data is known as the time consumption. That can be computed using the following formula:

$$\boxed{\textit{Time Consumed} = \textit{End Time} - \textit{Start Time}}$$

The time consumption of the proposed algorithm is given using fig 7. In this diagram the X axis depict the number of runs and the Y axis contains time require to process the algorithm in terms of milliseconds. According to the comparative results analysis the performance of the proposed technique minimize the time consumption. Therefore, this proposed is better and efficient of stream data classification of twitter data using enhancement of the decision tree i.e. C4.5.

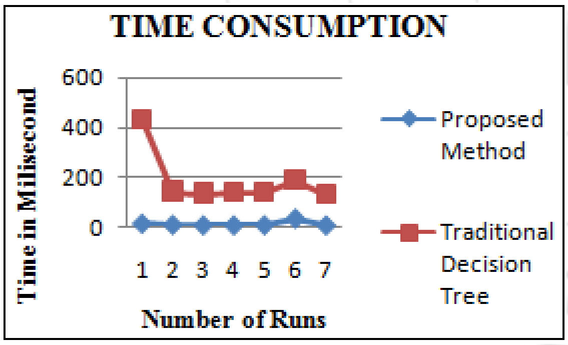

Figure 7: Time Consumption

5. Conclusion and Future Work

The huge quantity of data is not effectively and efficiently processed using the normal computing techniques. Therefore the requirement of new techniques is appeared for efficient computing and enhancing the system response time. In order to process data more efficiently the big data infrastructure is used. The key issue of this technology is, it start working only when the complete block of the data is completed. If the data is less in quantity the model delayed the processes of data response. Therefore a new technique for improving the user query response time is need to find which works on the streams of data and produces frequent response for the end user query. In order to achieve the solution for the addressed issue the "less data may response faster" hypothesis is used. The proposed technique is modeled on the streamed text data therefore the twitter live stream data is used. That data collection is performed with the help of storm infrastructure and twitter API. First the data collected and then the feature extraction is performed using the NLP tagging and the TF-IDF concept. Finally the ensemble based classification technique is applied to classify the data into the groups. These grouped or classified data is preserved in the HIVE structured data base directly. The preserved data is can be used with the query interface or any quick response application for providing the end user response.

6. Future Works

The proposed concept is implemented successfully and observed it is effective improving the performance. By motivating this performance enhancement the following research directions are suggested for future improvements.

- Involving the technique for processing of the increasing amount of data with time
- Increases the speed of the data fetching
- Utilizing the concept for data pre-fetching and caching

References

[1] Ikonomovska, Elena, Suzana Loskovska, and Dejan Gjorgjevik, "A survey of stream data mining", Proceedings of 8th National Conference with International Participation, ETAI. 2007.

[2] "What is Streaming Data?" available online at: https://aws.amazon.com/streaming-data/

[3] Hahsler, Michael, Matthew Bolanos, and John Forrest, "Introduction to stream: An Extensible Framework for Data Stream Clustering Research with R", Journal of Statistical Software (2015).

[4] P. Buddha Reddy and CH Sravan Kumar, "A Simplified Data Processing in MapReduce", (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 7 (3), PP. 1400-1402, 2016.

[5] Liao, Haojun, Jizhong Han, and Jinyun Fang, "Multi-dimensional index on hadoop distributed file system", Fifth International Conference on Networking, Architecture and Storage (NAS), 2010 IEEE, 2010.

[6] Zhang, Chi, Feifei Li, and Jeffrey Jestes, "Efficient parallel knn joins for large data in mapreduce", Proceedings of the 15th International Conference on Extending Database Technology, ACM, 2012.

[7] Lu, Wei, et al. "Efficient processing of k nearest neighbor joins using MapReduce." Proceedings of the VLDB Endowment 5.10, PP. 1016-1027, 2012

[8] Abadi, Daniel J., et al. "Aurora: a new model and architecture for data stream management" The VLDB Journal—the International Journal on Very Large Data Bases 12.2 (2003): 120-139.

[9] Stonebraker, Mike, et al. "C-store: a column-oriented DBMS", Proceedings of the 31st international conference on Very large data bases, VLDB Endowment, 2005.

[10] Floratou, Avrilia, et al. "Column-oriented storage techniques for MapReduce" Proceedings of the VLDB Endowment 4.7, PP. 419-429, 2011.

[11] Lam, Wang, et al. "Muppet: MapReduce-style processing of fast data." Proceedings of the VLDB Endowment 5.12 (2012): 1814-1825.