Scientific Journal of Impact Factor (SJIF): 5.71

e-ISSN (O): 2348-4470
p-ISSN (P): 2348-6406

International Journal of Advance Engineering and Research Development

Volume 5, Issue 02, February -2018

High Speed Classification of Massive Data Streaming Using Spark

M.JALASRI [1], AARTHIKA.K [2], VISHNU PRIYA.A [3]

[1]*Asst. Professor, Information Technology, JeppiaarMaamallan Engineering College,*
[2]*Student, Information Technology, JeppiaarMaamallan Engineering College,*
[3]*Student, Information Technology, JeppiaarMaamallan Engineering College*

Abstract—*Big data analytics deals with the mining of massive and high speed data streams with contemporary challenges—In this paper we perform an efficient nearest neighbor solution to classify high-speed and massive data streams using Apache Spark. A distributed metric tree has been designed to organize the case-base and consequently to speed up the neighbor searches. DS-RNGE algorithm is an instance selection method to find out the object in the nearest neighbor searches .Resilient distributed data set is a base to check the record in searches .Smart partitioning of the incoming data streams to parallelize the proposed algorithm using Apache Kafka which is a Spark tool to process the huge amount of data.. Spark is able to load data into memory and query it repeatedly, making it suitable for iterative processes (e.g., machine learning algorithms). Pseudo Random mode is used to partition the data in effective manner compared to references. We use the hashing algorithm to detect the duplicate records. Our work is used sequentially for real time entities of analyzing the live streaming records in nearest neighbor searches***.***

Keyword—*Big data, Apache Spark, Machine learning, Resilient distributed dataset, data streams, Pseudo Random mode, Hash algorithm.*

I. INTRODUCTION

Big Data are now rapidly expanding in all science and engineering domains, including physical, biological and biomedical sciences.Big Data concern large-volume, complex data sets with multiple autonomous sources [1]. Analyzing data that has not been carefully screened for such problems can produce misleading result. Therefore the quality of data and representation is must for analysis running.

The knowledge discovery during the training phase is more difficult when there is redundant and irrelevant information exist or noisy data. Data preparation and filtering steps can take considerable amount of processing time [2]. There is a popular open-source platform called Apache Spark which is used for large-scale data processing for iterative machine learning tasks. Modern datasets are rapidly growing in size and complexity, and there is a pressing need to develop solutions to harness this wealth of data using statistical methods [3]. The data are collected in the form of transient data streams when the dynamic environments applications implemented in machine learning. Compared to static data mining, processing streams imposes new computational requirements for algorithms to incrementally process incoming examples while using limited memory and time [4].

The approximate nearest neighbor search algorithm of scalable version is explained and which is used to find near duplicates among over a billion images. We address the nearest neighbor problem as the first step towards scalable image processing [5][6]. The further development of imbalanced learning is discussed with the addressing of challenges. The imbalanced data learning of full spectrum coverage is identified in seven research areas are: classification, regression, clustering, data streams, big data analytics and applications, e.g., in social media and computer vision [7].

The mining task was completed using map reduce jobs of three types. The reducers perform combination operations by constructing small ultra metric trees, and the actual mining of these trees separately in the decomposition of the third map reduce job. The data distribution and dimensions in the FiDoop cluster is sensitive, because item sets with different lengths have different decomposition and construction costs [8].

The transformation of the vast amount of stream raw data into information and knowledge representation is the challenge in continuous data flow, and accumulate experience over time to support future decision-making process. ADAIN is capable of learning from continuous raw data which is a incremental learning framework, accumulating experience over time, and using such knowledge to improve future learning and prediction performance [9].

In many cases we do not deal with static data collections, but rather with dynamic ones. They arrive in a form of continuous batches of data, known as data streams. To add a further difficulty, many modern data sources generate their outputs with very short intervals, thus creating the issue of high-speed data streams [10].

II. RELATED WORK

Potential applications for massive data analysis technique could be found in each human activity domain. Enterprises would like to discover interesting client behavior characteristics, example: on the basis of sensor or internet data, social media etc., are discussed with similar analysis of our work.

2.1. Big Data Analysis:

Google designed MapReduce in 2003, which is considered as one of the first distributed frameworks for large-scale data processing. MapReduce allows for automatically processing data in an easy and transparent way through a cluster of computers. The user only needs to implement two operators: 1) Map and 2) Reduce. In the Map phase, the system processes key-value pairs read directly from a distributed file system and transform them into another set of pairs (intermediate results). Each node is in charge of reading and transforming a set of pairs from one or more data partitions. In the Reduce phase, the key coincident pairs are sent to the same node and merged to yield the final result through an user-defined function. For further information about MapReduce and others distributed frameworks, please check. Apache Hadoop is an open-source implementation of MapReduce for reliable, scalable, and distributed computing. Despite its popularity and a number of implemented data mining algorithms, Hadoop is not suitable for many scenarios, with emphasis on those where there is a need for explicit data reusage. For instance, online, interactive, and/or iterative computing is affected by this problem.

Apache Spark is a distributed computing platform that became one of the most powerful engines developed for the big data scenario. According to its creators, this platform was designed to overcome the limitations of Hadoop. In fact, the Spark engine has shown to perform faster than Hadoop in many cases (up to 100× in memory). Thanks to its in-memory primitives, Spark is able to load data into memory and query it repeatedly, making it suitable for iterative processes (e.g., machine learning algorithms). In Spark, the driver (the main program) controls multiple workers (slaves) and collects results from them, whereas worker nodes read data blocks (partitions) from a distributed file system, perform some computations and save the result to disk. Resilient distributed dataset (RDD) is the base structure in Spark, on which the distributed operations are performed. A wide variety of operations are offered by RDDs, such as: filtering, mapping, and joining large data. These operations are designed to transform datasets by locally executing tasks within the data partitions, thus maintaining the data locality. Furthermore, RDDs are a versatile tool that allows programmers to preserve intermediate results (in memory and/or disk) in several formats for reusability purposes, as well as customize the partitioning for data placement optimization. Spark also allows us to use the RDD's API in streaming environments through the transformation of data streams into small batches. Spark Streaming's design enables the same batch code (formed by RDD transformations) to be used in streaming analytics, without a requirement for significant modifications.

2.2. Fast NearestNeighbor Searches:

Many techniques have been proposed to alleviate the *k*-NN search complexity. They range from metric trees (M-trees), which index data through a metric-space ordering; to locally sensitive hashing, which map (with high probability) those elements near in the space to the same bins. M-tree exploit properties such as the triangle inequality to make searches much more efficient in average, skipping a great amount of comparisons.

M-tree can be considered as one of the most important and simplest data structure in the space indexing domain. When searching in an M-tree, the algorithm descends through the structure by choosing the nearest node in each level, discarding every node that is not within the searching distance. M-tree may "backtrack" in case that some branches of the tree have remained un pruned. It is done to assure the correctness of the query. The overlap is allowed between the nodes when the backtracking process is skipped. *Spill trees* allow overlapping by introducing a new variable called *overlap buffer*.

This buffer represents the common area between two overlapping nodes and allows two nodes to have repeated examples. No backtrack is needed in these trees, however, at the cost of introducing potential redundancy. The overlap buffer is estimated by computing the distance (averaged over every instance in the training set) between every example and their nearest neighbors. single machine and their adaptation to distributed platforms poses a major difficulty. In distributed version of a metric tree is presented. The authors propose to maintain one top-tree in the master node that route the elements in the first levels. Once the elements have been mapped to the leaves a set of distributed subtrees performs searches in parallel. The idea behind is that the top-tree and the subtrees act like a complete metric tree, but in a fully distributed way.

The neighbors of an instance are determined by a special proximity graph called relative neighborhood graph by using RNGE, which is considered as most accurate method. Two points are considered as neighbors in the graph if there exist a connecting edge between them. The rule that determines this association is defined as follows: there exist an edge between two given points if there does not exist a third point that is closer to any of them than they are to each other. After building a graph the algorithm removes those instances misclassified by their neighbors (majority voting).

2.3. Data Partitioning:

In our system consists of building a distributed metric tree formed by a top-tree and a set of local trees. This distributed tree will be queried and updated during next iterations with incoming batches of data. The sampled data should be small enough to fit in a single machine and should maximize the separability between examples to avoid overlapping in the future subtrees. For each element, the algorithm searches the nearest leaf node in the top-tree. According to the correspondence between leaf nodes and subtrees we can determine to which subtree each element will be sent. This process is performed in a Map phase. The elements are shuffled to the subtrees according to their keys. Each subtree gets a list of elements to be inserted. For each subtree all received elements are inserted to the tree in a local way. This process is performed in a Reduce phase.

2.4. Streaming Process:

When a new batch of data arrives, we need to start the updating process with edition. This is aimed at inserting new instances, as well as removing those that became redundant over time.the algorithm computes which subtree each element falls into, following the same process described in the previous section. Once all instances are shuffled to subtrees, a local nearest neighbor search for each element is started in corresponding subtrees. the algorithm computes which subtree each element falls into, following the same process described in the previous section. Once all instances are shuffled to subtrees, a local nearest neighbor search for each element is started in corresponding subtrees. New examples can be inserted or not, whereas old examples (neighbors) can be removed or maintained.

2.5. Data Mining:

Contemporary machine learning problems are often characterized not only by a significant volume of data, but also by its velocity. Instances may arrive continuously in a form of a potentially unbounded data stream. This poses new challenges for learning algorithms, as they must offer adaptation mechanisms for ever-growing dataset, being able to update their structure in accordance with the current state of a stream. Data streams can be processed in two different operation modes. One is Chunk (batch), where data arrive in a form of instance blocks or we collect enough instances to form one. Second is Online, where instances arrive one by one and we must process them as soon as they become available.

2.6. Data Prediction:

For each element the algorithm searches for the nearest leaf node in the master node and shuffles the elements to the slave machines. Next, the standard M-tree search process is used to retrieve the *kp*-neighbors of each new element. For each group, formed by a new element and its neighbors, the algorithm predicts the element's class by applying the majority voting scheme to its neighbors. Notice that the query and the prediction are both performed in the same MapReduce phase as in the edition process.

III. PROBLEM DEFINITION

In many cases we do not deal with static data collections, but rather with dynamic ones. They arrive in a form of continuous batches of data, known as data streams.In such scenarios, we need not only to manage the volume but also the velocity of data, thus constantly updating and adapting our learning.

To add a further difficulty, many modern data sources generate their outputs with very short intervals, thus creating the issue of high-speed data streams.However, there exist a considerable gap between contemporary processing and storage capacities, which demonstrate that our ability to capture and store data has far outpaced our ability to process and utilize it.

IV. APPROACHED WORK

We perform an efficient nearest neighbor solution to classify high-speed and massive data streams using Spark. This algorithm consists of a distributed case base and an instance selection method that enhances its performance and effectiveness. Best of our knowledge, this is the first lazy learning solution in dealing with large-scale, high-speed, and streaming problems. In Spark, the driver (the main program) controls multiple workers (slaves) and collects results from them, whereas worker nodes read data blocks (partitions) from a distributed file system, perform some computations and save the result to disk. Resilient distributed dataset (RDD) is the base.

Smart partitioning of the incoming data streams to parallelize the proposed algorithm using Spark environment. Resilient distributed dataset (RDD) is the base structure in Spark, on which the distributed operations are performed. A wide variety of operations are offered by RDDs, such as: filtering, mapping, and joining large data. Nearest neighbor algorithms are highly popular in traditional machine learning, as they offer an easy implementation and a high efficiency.

4.1.ArchitectureDiagram:

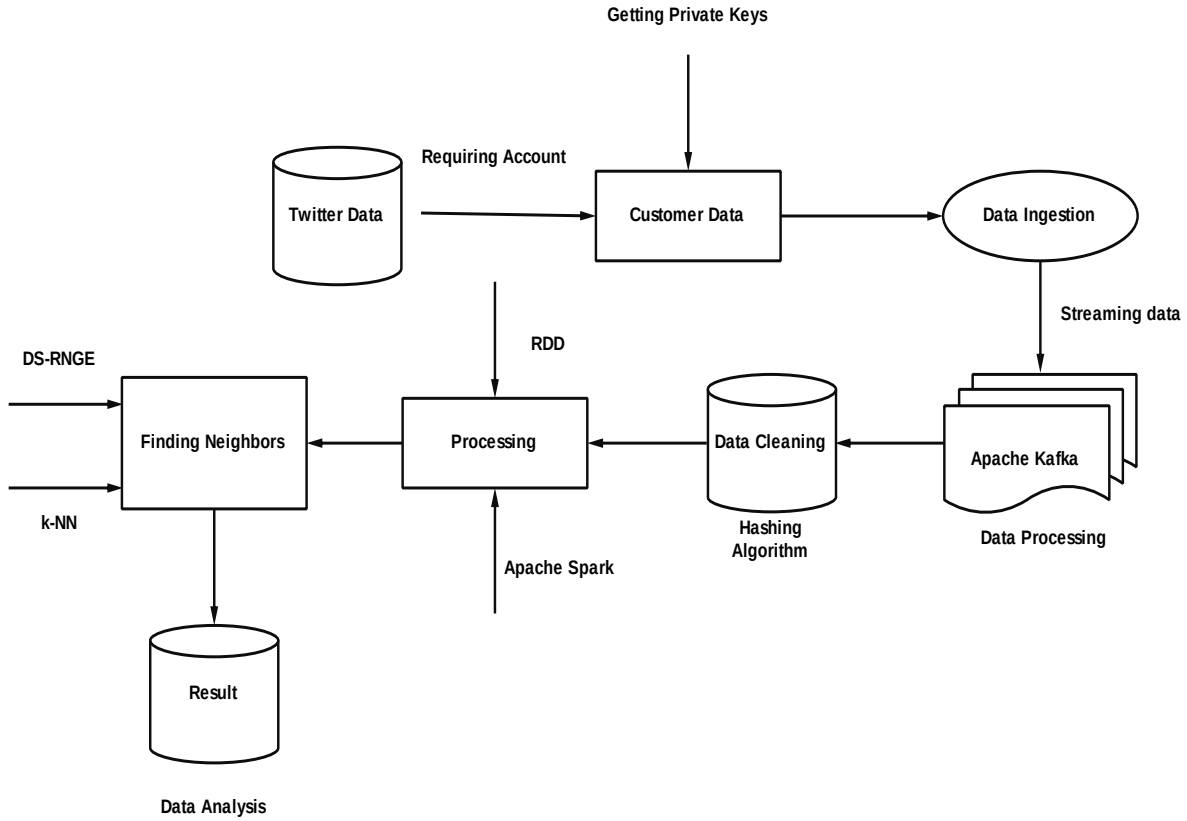

Figure 1:Architecture Diagram for High Speed Classification of Massive Data Streaming

4.2. Description:

The huge amount of incoming data is collected using the twitter media by accessing account information of getting the private keys from the customer. After collecting the details the data is ingested into the streaming records using Apache Kafka. The hashing algorithm is used to eliminate the duplicate datas. When the datas are cleaned thedatas are processed by the RDD and then the nearest neighbours are find out by using DS-RNGE and k-NN. Thus the data is analysed in Figure 1.

V. IMPLEMENTATION WORK

5.1. M-tree:

M-tree search process is used to retrieve the neighbors of each new element. For each group, formed by a new element and its neighbors, the algorithm predicts the element's class by applying the majority voting scheme to its neighbors.

5.2. DS-RNGE:

It consists of a distributed case-base and an instance selection method that enhances its performance and effectiveness. Our case-base is structured using a distributed metric tree, which is entirely maintained in memory to expedite further neighbor queries. DS-RNGE proceeds in two phases for each newly arrived batch of data. 1) An edition/update phase aimed at maintaining and enhancing the case-base. 2) A prediction phase that classifies new unlabeled data.

5.3. Hashing Algorithm:

A hash function is any function that can be used to map data of arbitrary size to data of fixed size. The hash table is used for rapid data lookup. The duplicated records are detected in accelerating table and database lookup.

VI. CONCLUSION

In DS-RNGE, a nearest neighbor classification solution for processing massive and high-speed data streams using Apache Spark. Up to our knowledge, DS-RNGE is the first lazy learning solution designed for large-scale, high-speed, and streaming problems. We recommended the instances by using a distributed metric tree, consisting of a top-level tree that routes the queries to the leaf nodes and a set of distributed subtrees that performs the searches in parallel. DS-RNGE includes an instance selection technique that constantly improves the performance and effectiveness of the learner by

only allowing the insertion of correct examples and removing outdated ones. As we used apache kafka for mining data, so our system is able to quickly respond to the continuous stream of data. That makes system more effective and efficient in processing in the streaming large data's.

VII. FUTURE WORK

Future work will concentrate on adding a condensation technique in order to control the ever-growing size of the case-base over time. By removing redundancy, the time cost derived from edition will be alleviated, while at the same time maintaining the original effectiveness. Additionally, we plan extend our approach to drifting data streams and propose time and memory efficient solutions for rebuilding the model as soon as the change occurs. We plan to tackle this challenge by extending our model with drift detection module, as well as by using instance weighting with forgetting to allow for smooth adaptation to changes. Additionally, we envision modifications of our algorithm that will make it suitable for mining massive and imbalanced data streams.We using apache kafka is very recent technology to mining streaming data's. Further we enhance our system efficient and effective using proposed technologies.

REFERENCES

[1] X. Wu, X. Zhu, G-Q Wu, and W. Ding, "Data mining with big data",IEEE Trans. KnowData Eng.,Jan. 2014.
[2] Cham, S. García, J. Luengo, and F. Herrera, "Data Preprocessing in Data Mining", S. García, J.Luengo, and F. Herrera, Switzerland: Springer, 2014.
[3] X. Menget al., J. Mach. Learn. Res., "Mllib: Machine learning in apache spark," vol. 17, no. 1, pp. 1235–1241, 2016.
[4] B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski, and M. Wo´zniak, "Ensemble learning for data stream analysis: A survey," Inf. Fusion, vol. 37, pp. 132–156, Sep. 2017.
[5] S. Ramírez-Gallego, B. Krawczyk, S. García, M. Wo´zniak, and F. Herrera, "A survey on data preprocessing for data stream mining: Current status and future directions,"Neurocomputing, vol. 239, pp. 39–57, May 2017.
[6] T. Liu, C. Rosenberg and H. A. Rowley in Proc,"Clustering billions of images with large scale nearest neighborsearch",IEEE Workshop Appl. Comput. Vis. (WACV), Austin, TX, USA, 2007, p. 28.
[7] B. Krawczyk, Progr. Artif.Intell.,"Learning from imbalanced data: Open challenges and future directions,"vol. 5, no. 4, pp. 221–232, 2016.
[8] Y. Xun, J. Zhang, and X. Qin, "FiDoop: Parallel mining of frequent itemsets using MapReduce," IEEE Trans. Syst., Man, Cybern., Syst., vol. 46, no. 3, pp. 313–325, Mar. 2016.
[9] H. He, S. Chen, K. Li, and X. Xu, "Incremental learning from stream data" IEEE Trans. Neural Netw.,vol. 22, no. 12, pp. 1901–1914, Dec. 2011.
[10] J. Maillo, S. Ramírez, I. Triguero, and F. Herrera, "kNN-IS: An iterative spark-based design of the k-nearest neighbors classifier for big data,"Knowl. Based Syst., vol. 117, pp. 3–15, Feb. 2017.

