Big Data Mining: Tools & Algorithms

http://dx.doi.org/10.3991/ijes.v4i1.5350

A.S. Hashmi and T. Ahmad
Jamia Millia Islamia, Delhi, India

***Abstract*—We are now in Big Data era, and there is a growing demand for tools which can process and analyze it. Big data analytics deals with extracting valuable information from that complex data which can't be handled by traditional data mining tools. This paper surveys the available tools which can handle large volumes of data as well as evolving data streams. The data mining tools and algorithms which can handle big data have also been summarized, and one of the tools has been used for mining of large datasets using distributed algorithms.**

***Index Terms*—Big Data, Data Mining, Hadoop, Large-scale Machine Learning.**

I. INTRODUCTION

The term "big data" refers to any collection of data which is so large and complex that it becomes difficult to handle using traditional database management systems and data processing tools. The first research paper on 'big data' appeared in 2000 by Diebold [1]. The sources of big data are social networking sites, e-commerce portals, sensors (smart devices/IoT), etc.

Big Data is characterized by 3 V's [2]: Volume, Variety, and Velocity. There is no fixed size to classify a dataset as big data or not, instead 'volume' dimension refers to a dataset which is large enough to be beyond the processing capabilities of traditional data processing tools, and the dataset can grow up to any size which can be in peta-bytes (10^{15} bytes), exa-bytes (10^{18} bytes) or even more. The 'variety' dimension is included because the traditional data warehouses store only structured data, but data generated on Twitter, etc. is highly unstructured, so such data is also beyond the processing capabilities of traditional data processing tools. While traditional data analytics is based on periodic analysis of data, big data is processed and analyzed in real-time or near real-time. Therefore, the third dimension of 'velocity' has also been included.

Apart from these 3 V's, two more V's have been added viz. Veracity and Value. 'Veracity' refers to the lack of quality and accuracy, as we can understand by the example of data generated on Twitter consisting of abbreviations, typos and colloquial speech. The dimension of 'value' was added due to the fact that big data after being processed must provide valuable information to the organization which can use it for making important business decisions, policies and strategies.

The remainder of this paper is structured as follows: In Section II, we discuss various tools available to process big data. Section III summarizes the machine learning models and data mining algorithms. Then in Section IV, we discuss the tools and algorithms available for data mining, with focus on tools supporting distributed processing and stream processing. The experimental setup and the results are discussed in Section V, and then we conclude the paper in Section VI.

II. BIG DATA TOOLS

Traditional data processing tools (e.g. Excel, SPSS, etc.) are not able to scale up with the size of growing datasets. For example, with Microsoft Excel 2007, analysts can't perform analysis on more than 1 million rows. So a tool which can scale-up should be used to deal with the growing datasets. Social networking platforms like Facebook, Twitter, etc. generate huge volumes of valuable social data per day at very high-speeds. This data need to be processed in real-time which can be used to predict outcome of elections, stock market behavior, etc. To do this we require tools which can perform analysis on streaming data. The major tools for handling different characteristics of big data are summarized in Table I.

MPP (Massively Parallel Processing) relational databases like *Greenplum*, *Vertica*, etc. have the capability to store and manage petabytes of data, where the data is partitioned across multiple nodes with each node having processors/memory to process data locally (it is a shared-nothing architecture i.e. no disk-level sharing). But MPP databases have an upper limit on storage capacity as well as it has same data processing limitations as associated with SQL.

Semi-structured data is structured data stored in a form other than tables (e.g. XML, JSON, etc.), and a database management system which provides a mechanism for storage and retrieval of such data is called a **NoSQL** data store (*NoSQL stands for Not Only SQL* indicating that they may also support SQL-like query languages). Examples of NoSQL data stores are *Cassandra*, *HBase*, *MongoDB*, etc. The NoSQL data stores do not have a fixed rigid schema like RDBMS for data to fit into, hence they can handle disparate data coming from different sources. **BigTable** [3] is a distributed storage system designed for managing large volumes of semi-structured data, or BigTable can be said to be a distributed NoSQL data store.

TABLE I.
BIG DATA TOOLS

Characteristic	Tool	Remark
Volume	MPP databases, Hadoop	distributed processing
Variety	NoSQL databases, Hadoop	schema-less data store
Velocity	In-memory databases, Spark	query data inside RAM instead of disk
Stream processing	Storm, S4	Real-time processing instead of batch processing

Google File System (GFS) [4] is a distributed file system designed to provide access to data using large clusters of commodity hardware. **Map-Reduce** [5] is a programming model for distributed and parallel processing. Map-Reduce model consists of two functions: "map" function splits the input data and distributes it across multiple machines to process the data in parallel, and "reduce" function that collects the results from the machines and resolves these results into a final aggregated result. Map-Reduce programming can be done in Java, Python, Ruby, etc. **Apache Hadoop** is an open-source tool for distributed storage and distributed processing. Hadoop uses HDFS (Hadoop Distributed File System) which is based on GFS for distributed storage of semi-structured data, and it uses Map-Reduce for distributed processing. **Amazon Elastic MapReduce (EMR)** is web-service which uses Hadoop to process large volumes of data on Amazon EC2 (Elastic Compute Cloud) cloud.

There are various Hadoop-based tools which make Big Data processing more convenient and efficient. **Apache Sqoop** is a tool for transferring data b/w Hadoop and RDBMS. **Apache Flume** is a tool for collecting, aggregating, and moving data from multiple sources into Hadoop. **Apache HBase** is an open-source distributed NoSQL data store modeled on BigTable, and it runs on top of HDFS. HDFS is concerned with storage of large files, whereas HBase allows storage of data as tables (but the data stored does not need a pre-defined fixed schema like with an RDBMS). HBase helps Hadoop to overcome challenges in random read and write. HBase doesn't provide a SQL-like query language and to process data in HBase tables we have to use Map-Reduce programs. **Apache Hive** is a data warehousing framework built on top of Hadoop, which allows users to store data in tables and write queries in HiveQL (Hive Query Language) to retrieve data instead of complex Map-Reduce program. Hive converts the HiveQL queries into a series of Map-Reduce jobs. But HiveQL has limited capability and can't be used to perform complex operations. *Cloudera*, *MapR*, *Hortonworks*, *IBM Infosphere BigInsights* are some Hadoop-based distributions providing HDFS, Map-Reduce, Pig, Hive, HBase, Sqoop, Flume, Hue, Impala, Oozie, ZooKeeper, Sentry, etc in one single package.

The tools discussed so far deal with 'volume' only and we need to look for tools which can also handle data streams**. In-memory databases** like *SAP HANA*, *Altibase*, etc. are gaining popularity for applications where response time is critical, but almost all of them are RDBMS. **Apache Storm** and **Apache S4 (Simple Scalable Streaming Systems)** are distributed real-time computation framework for processing fast, large streams of data. **Apache Spark** is a cluster computing framework, which provides performance up to 100 times faster than Map-Reduce through data caching. Apache Spark can also handle data streams through Spark Streaming library. **Apache Hadoop YARN** is a sub-project of Hadoop which enables Hadoop to go beyond batch processing and support broader data processing. YARN (Yet Another Resource Negotiator) is a part of Hadoop NextGen, and enables Hadoop to provide resource management capabilities to new engines like Apache Spark, etc. **Apache Samza** is a YARN-based stream processing framework. **Amazon Kinesis** is a cloud-based service for processing of distributed data streams.

The tools discussed in this section deal with only processing of voluminous or streaming data, and we can build applications using these tools for processing of such data. The tools which deal with mining of Big Data will be discussed in Section IV.

III. DATA MINING

Data Mining is the computational process of finding patterns in given data and making predictions for the new data. The main techniques for finding patterns are association-rule/frequent-pattern mining, clustering, classification and regression analysis. The data mining process consists of 3 stages: Getting and Cleaning data (or ETL – Extract, Transform, and Load), Model building (choosing an appropriate learning model/algorithm to get the desired result), and Deployment (application of model to data).

The data mining models can be divided into 2 categories: *unsupervised and supervised* models. Unsupervised models are concerned with finding patterns/clusters in the given data, whereas supervised models deal with training the system with historical data and making predictions or classifying new data. Various statistical models like regression analysis, bayes model, maximum-likelihood estimation, etc. and machine learning models like Neural Network, Decision tree, etc. are used for data mining.

The models for supervised learning are: Regression, Nearest Neighbors, Naïve Bayes, Decision trees, Perceptron, Support Vector Machines, Ensemble models. The popular algorithms for clustering are: k-means, fuzzy c-means (centroid models), BIRCH (hierarchical models), DBScan, OPTICS (density models). Other techniques for clustering are Expectation-Maximization (distribution models), Self Organizing Maps (Neural Network). The algorithms for frequent pattern mining are: Apriori algorithm, FP-Growth algorithm, Eclat, etc.

Dimensionality reduction may be required before applying a learning model as high-dimensional data (data that requires more than 2 or 3 dimensions) can be difficult to analyze. Principal Component Analysis (PCA), Independent Component Analysis (ICA), Factor Analysis, and Non-Negative Matrix Factorization are few techniques for dimensionality reduction.

Ensemble learning is the combination of multiple algorithms to get improved results for supervised learning. The popular ensemble methods are Bagging, Boosting, and Random Forests.

Outlier detection is a data mining technique to find anomalies in data. There are various clustering based algorithms for outlier detection.

Machine Learned Ranking is the use of machine learning for construction of ranking models in information retrieval systems (e.g. PageRank algorithm used by Google search engine). *Topic models* are used to discover "topics" in a collection of documents. A popular algorithm for topic mining is Latent Dirichlet Allocation.

The data mining algorithm studied so far are not able to handle data streams. So, in order to obtain useful information from data streams, we need to find new algorithms or modify existing algorithms. There are various models which have been proposed for learning a data stream: we can use the entire data stream, or we can take a sample of data stream, or we can also use the *sliding-window model* of computation. The sliding-window model considers only the last *n* elements of the stream, based on the assumption

that it is more important to use recent data for learning from data streams.

Widmer and Kubat [6] noticed that in the data stream, the classification or clustering centers continuously change with time. This is known as the problem of *concept-drift* in data streams. The model which is most appropriate to handle concept-drift is the sliding-window model. Gama et. al. [7] discusses the strategies for handling concept drift.

Many algorithms have been proposed in literature which deal with mining of data streams. These include algorithms for data stream clustering, data stream classification, etc. Data stream algorithms have gained more focus recently with the advent of big data, as the data generated is to be processed in real-time to obtain the desired value from it.

The most popular algorithms for clustering of a data stream are BIRCH [8], Scalable k-means [9], Single-pass k-means [10], Stream [11], Stream LSearch [12], CluStream [13], ODAC [14], DenStream [15], D-Stream [16], SWClustering [17], ClusTree [18], DGClust [19], StreamKM++ [20], SOMKE [21].

The algorithms for classification of data stream can be divided into two categories – *incremental learning* and *ensemble learning* algorithms. Ensemble learning combines multiple learning algorithms for prediction and may discard some training data as out-dated whereas incremental learning adjusts old results to what has been learned from the new data. The popular algorithms for incremental learning are Incremental Bayesian algorithm [22], Incremental SVM [23], VFDT (Hoeffding Tree) [24], CVFDT [25]. Some ensemble learning algorithms are Streaming Ensemble Algorithm (SEA) [26], Accuracy-Weighted Ensemble (AWE) [27], Dynamic Weighted Majority (DWM) [28].

The most notable work on regression on data streams is Hoeffding-based Regression trees [29]. The famous algorithms for pattern mining from a data stream are estDec [30], FP Stream [31], Moment [32], IncMine [33], CloStream [34]. The algorithms proposed for outlier detection in data streams are Abstract-C [35], STORM [36], COD and MCOD [37]. BRISMFPredictor [38] has been proposed for online Recommendation Systems.

The streaming data may come from multiple sources, hence the stream mining algorithms must be able to synchronize and aggregate these data streams for efficient data mining. The most notable work on multi-source data stream mining are: the SPIRIT [39] algorithm which discovers patterns from multiple data streams, Siddiqui proposed techniques for clustering and classification of multiple data streams in [40, 41], whereas Ikonomovska worked on multi-stream regression in [42].

IV. BIG DATA MINING TOOLS

For voluminous data, we need tools which can scale-up to data size. The traditional data mining tools like Weka, R, RapidMiner, etc. are not scalable as they do not support distributed processing. However, Wegener et al. [43] integrated Weka with Hadoop, and Das et al. [44] achieved integration of R and Hadoop.

As far as distributed learning algorithms are concerned, Chu et al. [45] adapted Google's Map-Reduce paradigm on a variety of learning algorithms including k-means, Expectation-Maximization, Locally Weighted Linear Regression, Logistic Regression, Naive Bayes, Support Vector Machine, Back-propagation Neural Network, Principal Component Analysis, Independent Component Analysis, and Gaussian Discriminative Analysis.

A classification of data mining tools based on type of data (batch/stream) and type of processing (distributed and non-distributed) is given in Fig. 1 below.

For large scale machine learning, we require tools and algorithms which can scale up to the large volumes of data. One such tool is **Apache Mahout**, which is a machine learning library built on top of Hadoop. It has Map-Reduce implementations of major machine learning models, which can help us to mine large volumes of data through distributed processing on a Hadoop cluster. Another tool is **Apache Spark's MLlib** which is also a machine learning library, has a broader range of machine learning models compared to Mahout, and can also run on a Hadoop cluster. Another important tool is **H2O** which is a "scalable" machine-learning API and supports R as well as Hadoop.

For mining of data streams, we require tools which support stream processing. **VFML (Very Fast Machine Learning)** toolkit supports k-means, EM algorithm for stream clustering, and Bayesian Network, Hoeffding Tree for stream classification. **SPMF** supports estDec, CloStream for pattern mining from data stream. **Massive Online Analysis (MOA)** is the most popular open-source tool which supports mining of data streams. MOA doesn't support distributed processing, and there may be situations where we may need distributed processing of data streams.

Jubatus is a distributed online machine learning framework supporting many data stream mining algorithms. Jubatus is one such tool which supports data mining of data streams through distributed processing. **Scalable Advanced Massive Online Analysis (SAMOA)** is a framework for development of distributed streaming machine learning algorithms while keeping programming abstraction for the underlying streaming processing engines like Apache Storm, S4, and Samza.

Vowpal Wabbit is an out-of-core machine learning system which speeds up machine learning with techniques such as hashing, reductions, etc. VW supports algorithms for multi-class classification like One-Against-All (OAA), Cost-sensitive OAA (CSOAA), Error Correcting Tournament (ect), Weighted All Pairs (wap). VW also supports algorithms for Regression (Ordinary Least Squares), Recommendation Systems (Matrix Factorization, Contextual Bandit), Regularization (Truncated Gradient Descent).

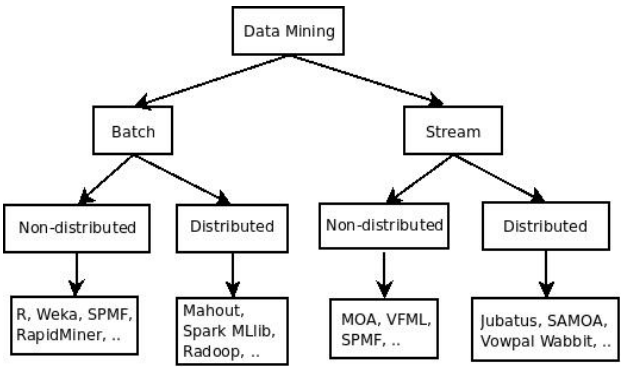

Figure 1. Data Mining Tools

V. EXPERIMENTS & RESULTS

In order to provide more insight into big data mining, the authors conducted experiments on large data sets (larger than the available RAM on the system). The datasets chosen were airline dataset (11.8 GB) for clustering and HIGGS dataset for classification (7.48 GB). The airline dataset [67] consists of 29 features with 123534969 observations. The HIGGS dataset [68] consists of 28 features with 11000000 observations.

The RAM available on the systems was only 2 GB and therefore full datasets could not be loaded in RAM for data mining. To handle the large datasets, we created a private cloud such that the combined RAM of all these systems in the cloud is more than the size of the dataset. The tool used for creation of the cloud as well as mining is H2O. It was observed that the data loaded into the H2O cloud is further compressed as the airline dataset loaded into the H2O cluster was of 3.78 GB and the HIGGS dataset loaded was of 2.3 GB.

The algorithms chosen were distributed k-means for clustering, and distributed random forest, distributed deep learning for classification. The time taken to cluster the dataset or to generate the classification model for each algorithm is given in Table II. The authors also varied the size of the cloud to study the effect of change of size of the cloud on the turn-around time of the results (the size of dataset was beyond the processing capabilities of a single-node, so multi-node was cloud was setup).

TABLE II.
EXPERIMENTAL RESULTS: TIME TAKEN FOR MODEL GENERATION

Dataset	**Algorithm**	**Cloud-size (4 core per system)**		
		1	***3***	***6***
airline.csv	Distributed k-means	-	1.35 hours	6.24 min
HIGGS.csv	Distributed Random Forest (20 trees)	-	32 min	3.68 min
HIGGS.csv	Deep Learning (200X200, 10 epochs)	-	2.66 hrs	47.2 min

VI. CONCLUSION

In the experiments performed, the authors were able to cluster and classify a large dataset on a private cloud, which can be scaled up to handle the growing dataset. The time taken to cluster the dataset and to generate the models was also quite satisfactory. The authors also observed that the turnaround-time of the results improved by increasing the size of the cloud.

The turn-around time to generate the classification model and the accuracy of the algorithm depends upon the parameters of the algorithm. The authors here have not attempted to compare these algorithms in any way, as the focus of this paper is only to survey the tools and algorithms.

REFERENCES

[1] F. Diebold, "Big Data: Dynamic Factor Models for Macroeconomic Measurement and Forecasting," *Discussion read to the Eighth World Congress of the Econometric Society*, 2000.

[2] D. Laney, "3-D Data Management: Controlling Data Volume, Velocity and Variety," *META Group Research Note*, February 6, 2001.

[3] F. Chang, J. Dean, S. Ghemawat, W.C. Hsieh, D.A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R.E. Gruber, "Bigtable: A Distributed Storage System for Structured Data Research," Google, 2006.

[4] S. Ghemawat, H. Gobioff, and S. Leung, "The Google file system," *Proceedings of the nineteenth ACM symposium on Operating systems principles*, Vol. 37, Issue 5, pp. 29-43, 2003. http://dx.doi.org/10.1145/945445.945450

[5] J. Dean, and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Proceedings of the 6th OSDI*, pp. 137-150, 2004.

[6] G. Widmer, and M. Kubat, "Learning in the presence of concept drift and hidden contexts," *Journal of Machine Learning*, Vol. 23, Issue 1, pp. 69-101, 1996. http://dx.doi.org/10.1007/BF00116900

[7] J. Gama, A. Bifet, I. Zliobaite, M. Pechenizkiy, and A. Bouchachia, "A Survey on Concept Drift Adaptation," *ACM Computing Surveys*, Vol. 1, No. 1, 2013.

[8] T. Zhang, R. Ramakrishnan, and M. Livn, "BIRCH: A new data clustering algorithm and its applications," *Data Mining and Knowledge Discovery*, Vol. 1, Issue 2, pp. 141-182, 1997. http://dx.doi.org/10.1023/A:1009783824328

[9] P. S. Bradley, U. M. Fayyad, and C. Reina, "Scaling clustering algorithms to large databases," *Proceedings of Knowledge Discovery and Data Mining, AAAI Press*, pp. 9-15, 1998.

[10] F. Farnstrom, J. Lewis, and C. Elkan, "Scalability for clustering algorithms revisited," *SIGKDD Explorations*, pp. 51-57, 2000. http://dx.doi.org/10.1145/360402.360419

[11] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan, "Clustering data streams," *IEEE Symposium on Foundations of Computer Science*, pp. 359-366, 2000. http://dx.doi.org/10.1109/sfcs.2000.892124

[12] L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani, "Streaming data algorithms for high-quality clustering," *18th International Conference on Data Engineering*, pp. 685-694, 2002. http://dx.doi.org/10.1109/ICDE.2002.994785

[13] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," *Proceedings of the 29th Conference on Very Large Data Bases*, pp. 81-92, 2003. http://dx.doi.org/10.1016/b978-012722442-8/50016-1

[14] P. Rodrigues, J. Gama, and J. Pedroso, "ODAC: Hierarchical clustering of time series data streams," *Proceedings of the Sixth SIAM International Conference on Data Mining*, pp. 499-503, 2006. http://dx.doi.org/10.1137/1.9781611972764.48

[15] F. Cao, M. Ester, W. Qian, and A. Zhou, "Density-based clustering over an evolving data stream with noise," *Proceedings of the Sixth SIAM International Conference on Data Mining*, SIAM, pp. 328-339, 2006. http://dx.doi.org/10.1137/1.9781611972764.29

[16] Y. Chen, and L. Tu, "Density-based clustering for real-time stream data," *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM Press, pp. 133-142, 2007. http://dx.doi.org/10.1145/1281192.1281210

[17] A. Zhou, F. Cao, W. Qian, and C. Jin, "Tracking clusters in evolving data streams over sliding windows," *Knowledge and Information Systems*, Vol. 15, No. 2, pp. 181-214, 2008. http://dx.doi.org/10.1007/s10115-007-0070-x

[18] P. Kranen, I. Assent, C. Baldauf, and T. Seidl, "The clustree: indexing micro-clusters for anytime stream mining," *Knowledge and Information Systems*, Vol. 29, No. 2, pp. 249-272, 2011. http://dx.doi.org/10.1007/s10115-010-0342-8

[19] J. Gama, P. P. Rodrigues, and L. Lopes, "Clustering distributed sensor data streams using local processing and reduced communication," *Intelligent Data Analysis*, Vol. 15, No. 1, pp. 3-28, 2011.

[20] M. R. Ackermann, M. Martens, C. Raupach, K. Swierkot, C. Lammersen, and C. Sohler, "Streamkm++: A clustering algorithm for data streams," *ACM Journal of Experimental Algorithmics*, Vol. 17, No. 1, 2012. http://dx.doi.org/10.1145/2133803.2184450

[21] H. He, and H. Man, "SOMKE: Kernel Density Estimation Over Data Streams by Sequences of Self-Organizing Maps," *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 23, No. 8, 2012.

[22] J. Roure, and R. Sanguesa, "Incremental Methods for Bayesian Network Learning," 1999.

[23] N. A. Syed, H. Liu, and K. K. Sung, "Handling concept drift in incremental learning with support vector machines," *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 317-321, 1999. http://dx.doi.org/10.1145/312129.312267

[24] P. Domingos, and G. Hulten, "Mining high-speed data streams," *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 71-80, 2000. http://dx.doi.org/10.1145/347090.347107

[25] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 97-106, 2001. http://dx.doi.org/10.1145/502512.502529

[26] W. N. Street, and Y. A. Kim, "Streaming Ensemble Algorithm (SEA) for large-scale classification," *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 377-382, 2001. http://dx.doi.org/10.1145/502512.502568

[27] H. Wang, W. Fan, P.S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," *Proceedings of the 9th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 226-235, 2003. http://dx.doi.org/10.1145/956750.956778

[28] J. Z. Kolter, and M. A. Maloof, "Dynamic Weighted Majority: A New Ensemble Method for Drifting Concepts," *ACM Journal of Machine Learning Research*, Vol. 8, pp. 2755-2790, 2007.

[29] E. Ikonomovska, J. Gama, R. Sebastiao, and D. Gjorgjevik, "Regression Trees from Data Streams with Drift Detection," *Proceedings of the 12th International Conference on Data Science*, pp. 121-135, 2009. http://dx.doi.org/10.1007/978-3-642-04747-3_12

[30] J. H. Chang, and W. S. Lee, "Finding Recent Frequent Itemsets Adaptively over Online Data Streams," *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 487-492, 2003. http://dx.doi.org/10.1145/956750.956807

[31] C. Gianella, J. Han, J. Pei, X. Yan, and P. S. Yu, "Mining Frequent Patterns in Data Streams at Multiple Time Granularities," *Data Mining: next generation challenges and future directions*, MIT/AAAI Press, pp. 191-212, 2004.

[32] Y. Chi, H. Wang, P. S. Yu, and R. R. Muntz, "Moment: Maintaining Closed Frequent Itemsets over a Stream Sliding Window," *4th IEEE International Conference on Data Mining*, pp. 59-66, 2004.

[33] J. Cheng, Y. Ke, and W. Ng, "Maintaining frequent closed itemsets over a sliding window," *Journal of Intelligent Information Systems*, Vol. 31, No. 3, pp. 191-215, 2008. http://dx.doi.org/10.1007/s10844-007-0042-3

[34] S. J. Yen, C. W. Wu, Y. S. Lee, V. S. Tseng, and C. H. Hsieh, "A Fast Algorithm for Mining Frequent Closed Itemsets over Stream Sliding Window," *IEEE International Conference on Fuzzy Systems (FUZZ)*, 2011. http://dx.doi.org/10.1109/fuzzy.2011.6007724

[35] D. Yang, E. Rundensteiner, and M. Ward, "Neighbor-based pattern detection for windows over streaming data," *In EDBT*, pp. 529–540, 2009. http://dx.doi.org/10.1145/1516360.1516422

[36] F. Angiulli, and F. Fassetti, "Distance-based outlier queries in data streams: the novel task and algorithms," *Data Mining and Knowledge Discovery*, Vol. 20, Issue 2, pp. 290–324, 2010. http://dx.doi.org/10.1007/s10618-009-0159-9

[37] D. Georgiadis, M. Kontaki, A. Gounaris, A. Papadopoulos, K. Tsichlas, and Y. Manolopoulos, "Continuous Outlier Detection in Data Streams: An Extensible Framework and State-Of-The-Art Algorithms," *SIGMOD*, 2013. http://dx.doi.org/10.1145/2463676.2463691

[38] G. Takacs, I. Pilaszy, B. Nemeth, and D. Tikk, "Scalable Collaborative Filtering Approaches for Large Recommender Systems," *Journal of Machine Learning Research*, pp. 623-656, 2009.

[39] S. Papadimitriou, J. Sun, and C. Faloutsos, "Streaming Pattern Discovery in Multiple Time-Series," *Proceedings of 31st VLDB Conference*, 2005.

[40] Z. F. Siddiqui, and M. Spiliopoulou, "Combining Multiple Interrelated Streams for Incremental Clustering," *SSDBM*, 2009. http://dx.doi.org/10.1007/978-3-642-02279-1_38

[41] Z. F. Siddiqui, and M. Spiliopoulou, "Tree Induction over Perennial Objects," *SSDBM*, 2010. http://dx.doi.org/10.1007/978-3-642-13818-8_43

[42] E. Ikonomovska, "Regression on evolving multi-relational data streams," *LEMIR*, 2011. http://dx.doi.org/10.1145/1966874.1966875

[43] D. Wegener, M. Mock, D. Adranale, and S. Wrobel, "Toolkit-based High-Performance Data Mining of Large Data on MapReduce Clusters," *Proc. Int'l Conf. Data Mining Workshops (ICDMW'09)*, pp. 296-301, 2009. http://dx.doi.org/10.1109/icdmw.2009.34

[44] S. Das, Y. Sismanis, K. S. Beyer, R. Gemulla, P. J. Haas, and J. McPherson, "Ricardo: Integrating R and Hadoop," *Proceedings ACM SIGMOD Int'l Conf. Management Data (SIGMOD'10)*, pp. 987-998, 2010. http://dx.doi.org/10.1145/1807167.1807275

[45] C. T. Chu, S. K. Kim, Y. A. Lin, Y. Yu, G. R. Bradski, A. Y. Ng, and K. Olukotun, "Map-Reduce for Machine Learning on Multicore," *Proc. 20th Ann. Conf. Neural Information Processing Systems (NIPS'06)*, pp. 281-288, 2006.

AUTHORS

A.S. Hashmi and **T. Ahmad** are with Jamia Millia Islamia, Delhi, India.

Submitted 10 December 2015. Published as resubmitted by the authors 28 February 2016.