International Journal of Engineering Trends and Technology
ISSN: 2231 – 5381 /doi:10.14445/22315381/IJETT-V68I10P217
Volume 68 Issue 10, 100-104, October 2020
© 2020 Seventh Sense Research Group®

Big Data Preprocessing: Needs and Methods

Sandeep Dalal[1], Vandna Dahiya[2]
[1,2]DCSA, Maharshi Dayanand University, Rohtak, Haryana, India
[1]sandeepdalal.80@gmail.com
[2]vandanadahiya2010@gmail.com

Abstract *- Big data is an assemblage of large and complex data that is difficult to process with the traditional DBMS tools. The scale, diversity, and complexity of this huge data demand new analytics techniques to extract useful and hidden value from it. Data must be prepared before starting mining as real data is sometimes not suitable for mining, and poor quality finishes in poor results. This paper presents the needs, various problems, and solutions for the preprocessing of big data.*

Keywords - *Big data, Discretization, MapReduce, Preprocessing*

I. INTRODUCTION

Data is exploding from various fields such as business, astronomy, transportation, genomics, retail, entertainment, etc. This huge data generating from all these fields are high in speed, huge in volume, and dissimilar in variety. This data is called big data, which is a collection of enormous and complicated data sets. This data cannot be processed directly by manual applications or with traditional data processing applications. The current rate of generating data has exceeded the managing capability of our conventional systems. This factor indicates the 'big' challenge for data analytics society.

Many applications have been developed in recent times to tackle the problem of big data analytics. These applications make the distributed know-hows familiar to the user while hiding the complex strategies and technical details of distributed environments. Also, the methods and algorithms required for big data analytics are different from conventional algorithms, as they need to work with the distributed environment. So, they need to be re-designed or build again for these systems, which is a vast challenge for the data mining society.

II. DATA PREPROCESSING

Data preprocessing signifies the set of practices applied to the data before applying any data mining approach. It is one of the vital steps of the KDD procedure. According to Dorian Pyle, "The fundamental purpose of data preparation is to manipulate and transform raw data so that the information content enfolded in the data set can be exposed, or made more easily accessible." The raw data can be imperfect, incomplete, or might have redundancies, which are not suitable for directly mining the process of mining. With an increase in size and dimensions of the data, high-level mechanisms are required to analyze it. Data preprocessing uncover quality data before applying the knowledge extraction algorithms. With the preprocessing of data, it becomes more feasible to be adopted by the particular data-mining algorithm. The following points can summarize the importance of data preprocessing-

- Real data are most likely to be flawed, inconsistent, and redundant. It may be lacking the attributes or may have errors or outliers. It isn't easy to use it directly for the process of mining. The quality of results depends on the quality of the data.
- Real data come from diverse sources. It needs to be in a standard form before feeding it to some data mining method.
- Preprocessing of data helps in making the data adaptable for the necessities of a data-mining algorithm.
- Preprocessing data helps generate a smaller subset of data, which is essential for many applications for their learning phases.
- Preprocessing data helps generate a smaller subset of data, which is essential for many applications for their learning phases.

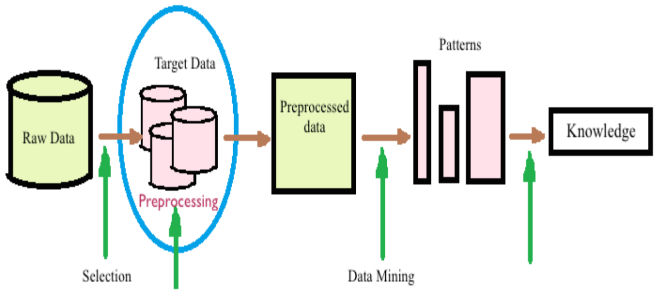

Figure 1 Preprocessing in KDD Process

A. *Major Tasks in Preprocessing of Data*

Before mining, various steps are required for preprocessing the data. Some major steps are discussed in this section.

a) Data Integration: Data is obtained from various sources. These sources might use different coding mechanisms, which may generate diverse representations. So, the data need to be integrated from numerous sources and then symbolized in a homogeneous form. For example, for the same organization, the employees' salary can be denoted in dollars or euros based on the office's geography. It can be paid monthly or yearly. All this information needs to be converted into a standardized form. So, the problems of representation and codification are addressed under the integration of data.

b) Data Cleaning*:* Data mining methods may have mechanisms to pact with data that is noisy and incomplete. Most of the time, the cleaning of data is done before starting the process of mining. For example, sometimes, the mining method's requirement may be the age in place of birth year. So, cleaning refers to repair the inconsistencies, smoothening of noisy data, and documentation of outliers.

c) Data Perfection: Most of the mining approaches consider that the data is complete and noise-free. Some values might be missing or not available in real-world data, and there can be much noise. To make the data suitable for mining, missing values need to be filled, and noisy data need to be removed. If the data is composed of missing values that have not been stored for any reason, then the dealing of such values is necessary before starting the mining process. Otherwise, mining will result in inappropriate results. It can produce biasedness in the whole process. One approach is to discard the complete instance, which again leads to the biasedness and causes information-loss of the rest of the instance to deal with them. The other option is to use statistical and probabilistic models to fill these values. Noise can also affect the input values, and so does the output. To remove the noise, preprocessing is required based on the amount of noise present. If the noise is small, data polishing methods are sufficient. But if the noise is large, noise filters are used, which find the noisy instances and modify the values.

d) Data Transformation: Transformation of data aims at converting the data into a format that is best conceivable for the mining algorithms. It can be done through operations like smoothing, aggregation, normalizing, or generalizing the data. Smoothing refers to the removal of the noise. Aggregation means integrating smaller values. Normalizing refers to the scaling of data into better and specified ranges. Generalization refers to obtain higher degrees of hierarchy.

e) Data Reduction: Data reduction refers to the process of obtaining the data with reduced presentations while producing the same results on analysis. Methods of data reduction are feature selection, instance selection, and discretization. Feature selection indicates selecting a smaller subset of the data/attributes that optimizes mining by decreasing the complexity. For instance, selection, examples are chosen, which are more relevant to the application by using different sampling methods. Continuous values, such as waves, are transformed into discrete values that represent the information more concisely and easier to understand.

Thus, various preprocessing methods help in applying data mining techniques easily and finding better results. But the process of preprocessing is not structured. The arrangement of various methods decides the quality of the results. So, the design of an automatic process for various preprocessing steps is a challenge among the data mining community.

III. BIG DATA PREPROCESSING

With the increase in volume and variety, the practice of data preprocessing might get more complex and time-consuming. In this section, the importance of preprocessing for various types of learning is discussed. Various techniques of preprocessing the big data and frameworks are also presented.

A. Importance of Big Data Preprocessing

a) For Unsupervised Learning: Quality of results depends on the quality of data. For unsupervised learning, such as classification and association rule mining for big data, steps such as data cleaning, transformation, and discretization are crucial.

b) For Semi-Supervised Learning: Data preprocessing is important at the instance level. More and more data will generate, and labeling examples cannot be done for all the data, especially predictive techniques.

c) For Real-time Applications**:** For business, medical and other fields, real-time processing is important. Preprocessing steps such as noise reduction are essential in such cases.

B. Methods for Big Data Preprocessing

a) Feature Selection: Feature selection performs a vibrant role in dealing with big datasets with wider dimensions. There are several means for feature selection, which can be used along with the distributed computing to scale well with multiple nodes' complexity. Some of them are discussed here- Approximate Heuristic- Proposed by Singh et al. in 2009. This method is enhanced for logistic regression in MapReduce. It uses the greedy search approach to hand-picked the features. Meena et al. proposed another method for feature selection in 2012, based on Ant Colony Optimization, to select an ideal subsection of features. It also works on MapReduce architecture. Tanupabrungsun et al. proposed a genetic algorithm-based method in 2013 with a fitness function. It works in parallel with MapReduce for management and fitness evaluation. In 2015, Kumar et al. proposed two methods for feature selection- ANOVA and Friedman Test. These are statistical methods and can be run independently in parallel on MapReduce. Zhao et al. proposed a platform for supervised and unsupervised learning using Akaike information criterion and Bayesian information criterion in 2013.

b) Imbalanced Data*:* Instances in big data sets are sparse. Some of the classes have a high probability of the instances, while some have less probability. This causes the problem of imbalance in supervised learning. Various studies have been proposed based on oversampling and under-sampling methods to handle the obstacle of imbalanced data. In 2014, Hu et al. suggested an improved SMOTE (Synthetic Minority Over-sampling Technique), where classes with smaller instances are replicated. Zhai et al. developed a method of oversampling grounded on the nearest neighbor approach for collaborative knowledge in 2016. In 2015, Triguero et al. developed an under-sampling method based on Map and Reduce phases. Decision trees are generated after every Map stage for under-sampling. These trees are then used to classify the test set. This method is widely used for big data classification. Park et al. developed a parallel version of SMOTE for the revealing of traffic among the instances. The Map phase counts the gaps among the input, and the Reduce phase sorts them.

c) Incomplete Data*:* Data incompleteness is a common problem in real-life data sets. As there may be some missing value due to individual or system failure. It can cause gaps in the results. The systems in big data are more prone to the problem of data incompleteness. Missing values need to be replaced before they generate some undesirable consequences in mining. Missing value imputation is difficult in the big data scenario as there is a need to consider many relationships among the instances to replenish the finest probable value. Noise treatment is another solution. It can be done in two ways- by computing the likenesses among the other data points and blending several ensembles' decisions from the noise identification method. To solve incomplete data, Chen et al. proposed a data cleansing technique in 2014 based on a set-valued choice technique. The method is being implemented on the MapReduce system. Zhang et al. gave another method in 2015, based on calculating the relation matrix of rough set theory.

d) Discretization: Every data mining approach requires its input to be of a certain domain or data type. For example, decision tree-based classification requires the data to be categorical. Discretization converts the data from quantitative to qualitative form. It also simplifies the data, although a certain loss of information is there during conversion. For big data sets, enhanced versions have been developed. In 2014, Zhang et al. developed a Hadoop-based method, which uses a Chi-Squared discretization model. But the method is not very scalable. Another approach was developed by Ramirez et al. in 2015. Feature sorting and boundary point generation operations are being proposed for discretization, and evaluation is done on some big data sets.

e) Instance Reduction: Instance reduction is a decreasing number of samples for an algorithm's learning phase. Techniques, for instance, reduction, reduce the subset size of data. It isn't easy to use this method as a preprocessing step due to big data analytics's size. High level and iterative computations are required. It also sinks the performance. In 2015, Triguero et al. developed a method of instance reduction by applying the IR technique over data partitions. The Reduce phase of the Hadoop provides ways to combine the instance sets.

Most of the above methods are used as a preprocessing step for datasets smaller than five GB. Table 1 shows a summary of these methods conducted by various researchers and the maximum data size handled by them. More scalable preprocessing methods are required for big data sets. The next section of the paper presents various challenges in preprocessing in a big data environment.

Table 1: Various Preprocessing Methods and Data Size Handled by them

Method	**Author**	**Maximum Size (GB)**	**Framework**
Feature Selection	Peralta D et al. [2015]	305	Hadoop MapReduce
Feature Selection	Ordozgoiti B et al. [2015]	1.9	Apache Spark
Feature Selection	Zhao Z et al. [2013]	48	MPI
Feature Selection	Tan M et al. [2014]	4	MATLAB
Imbalanced Data	Lopez V et al. [2014]	150	Hadoop MapReduce
Imbalanced	Triguero I et al. [2016]	75	Apache Spark
Incomplete	Zhang J et al. [2015]	19	Hadoop MapReduce
Discretization	Ramirez-Gallego S et al. [2016]	305	Apache Spark
Discretization	Zhang Y et al. [2014]	4	Hadoop MapReduce
Instance Reduction	Triguero I et al. [2015]	1	Hadoop MapReduce

C. Frameworks for Big Data Preprocessing

There are various frameworks available now a day for the preprocessing of large-scale data with distributing platforms. These frameworks provide abstraction by enveloping all the technical complications required to manage and merge the data from diverse data sources into an integrated system of data that can then be directly used by data scientists. Some of the frameworks are discussed here.

a) MapReduce: MapReduce is the first java-based framework that enables handling large datasets independently on multiple nodes. Datasets are processed in automation in a distributed way. MapReduce consists of two components- Map and Reduce. The Map takes the input data in the shape of a file or directory and converts it into key-value pairs. The reducer then fetches these pairs as the input, which reduces the data and generates the results. The framework is very scalable and works with multiple computing nodes. Once an algorithm is written in MapReduce form, it can be made to run over thousands of machines or nodes. The framework itself manages all the tasks of data processing such as issuing the tasks to nodes, verifying the completion of tasks, managing the nodes, etc., and taking care of technical distinctions such as fault tolerance, data partition, and communications overhead.

b) Apache Hadoop**:** Hadoop is an open-source software based on MapReduce architecture from the house of Apache. It has been designed with all the features mentioned for MapReduce. It doesn't scale well with large problems as it is a disk-based architecture, and the cost of input-output communication is very high.

c) Apache Spark**:** Apache Spark is another implementation based on MapReduce. It performs faster than Hadoop as it is an in-memory based computing. It loads data into memory and then re-uses it again and again. Spark is based on RDD (Resilient Distributed Datasets) that regulators the persistence and manages data partition in addition to other MapReduce features.

***d) Apache Flink**:* Flink is an approach for real-time applications that bridges the gap between stream and batch processing. It runs the algorithms either in parallel or in the pipeline. It also supports the iterative process of algorithms. It does not have its system for data storage but depends on other systems such as HDFS, Cassandra, etc.

***e) MLib**:* MLib is an influential library for machine learning. It empowers the use of Spark for data analytics. MLib consists of two packages- mlib and ml. The mlib has been built on top of RDD, which has popular machine learning methods. The ml consists of new methods such as pipelines and high-level APIs for machine learning with Spark. The MLib contains packages for discretization, normalization, feature selection, etc. Some of them are Binarizer, Tokenizer, StopWordsRemover, Bucketizer, VectorAssambler, VectorSlicer, Chi-Squared selector, StringIndexer, VectorIndexer, etc.

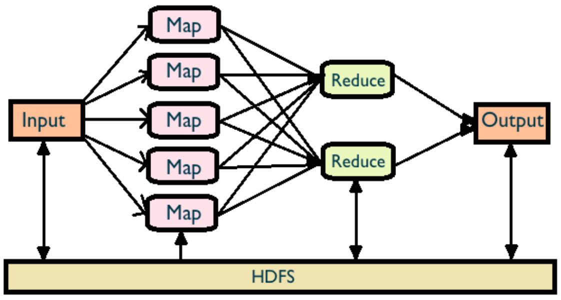

Figure 2 Framework of MapReduce

IV. CHALLENGES IN BIG DATA PREPROCESSING

Preprocessing in the field of big data faces several challenges. To scale up the methods is the major challenge and key point. Data mining in big data is not static. With the increase in new problems of big data, the challenges of preprocessing are also increasing. For instance, there is a need to develop new techniques to obtain a subset of data from large datasets without diminishing mining performance. They need to be scaling well with the demand.

Similarly, it is a great confront to substitute the finest conceivable value by analyzing the relationships among other attributes. Another challenge is how to arrange and combine all the methods of preprocessing for an optimal approach. As studied by Garcia A et al. in 2016, the arrangement affects the overall process of mining. Various other factors also affect the arrangement, such as the dependency of intermediate results, huge data volume, parallel framework, and iterative approaches. Various new technologies have been developed in recent years to mine the big data, and the preprocessing methods take advantage of them. Spark is one of them. Methods have been developed under the MLib of Spark. Flink is another approach that bridges the gap between stream and batch processing for real-time applications. Still, there is the need to develop the preprocessing techniques on Flink for these applications.

V. CONCLUSION

Data is exploding from various sources, and the volume, variety, and velocity are increasing. With the practice of large data skeletons like Spark and Flink, there is a huge change in mining and preprocessing. Data preprocessing contributions have been presented in this study for these new frameworks for various methods like feature selection, data imperfection, instance reduction, etc. Various challenges are also presented concerning big data, such as the scaling of various techniques. For further research work, these challenges need to be addressed by the data mining industry. There should be collaboration among the researchers, data scientists, and academia for big data preprocessing while exploring new domains.

REFERENCES

[1] Data Preparation for Data Mining, Dorian Pyle, 1999

[2] Singh S, Kubica J, Larsen SE, Sorokina D. "*Parallel Large Scale Feature Selection for Logistic Regression. In: SIAM*" International Conference on Data Mining (SDM). Sparks, Nevada: 2009. p., 1172–1183.

[3] Meena MJ, Chandran KR, Karthik A, Samuel AV. "*An Enhanced ACO algorithm to Select Features for Text Categorization and its Parallelization*". Expert Syst Appl. 2012; 39(5):5861–871.

[4] Zhao Z, Zhang R, Cox J, Duling D, Sarle W. "*Massively Parallel Feature Selection: An Approach Based on Variance Preservation*". Mach Learn. 2013; 92(1):195–220.

[5] Hu F, Li H, Lou H, Dai J. "*A Parallel Oversampling Algorithm Based on NRSBoundary-SMOTE*". J Inf Comput Sci. 2014; 11(13):4655–665.

[6] Zhai J, Zhang S, Wang C. "*The Classification of Imbalanced Large Data Sets Based on Mapreduce and Ensemble of Elm Classifiers".* Int J Mach Learn Cybern. 2016. doi:http://dx.doi.org/10.1007/s13042-015-0478-7

[7] Park S-h, Kim S-m, Ha Y-g. "*Highway Traffic Accident Prediction Using Big Data Analysis*". J Supercomput. 2016. doi:http://dx.doi.org/10.1007/s11227-016-1624-z.

[8] Zhang Y, Yu J, Wang J. "*Parallel Implementation of chi2 Algorithm in MapReduce Framework*". In: Human-Centered Computing - First International Conference, HCC. Germany: Springer: 2014. p. 890–9.

[9] Triguero I, Peralta D, Bacardit J, García S, Herrera F. "*MRPR: A Mapreduce Solution for Prototype Reduction in Big Data Classification*". Neurocomputing. 2015; 150 Part A:331–45.

[10] Vandna Dahiya, Sandeep Dalal, "*Big data Mining: Current Status and Future Prospects*", International Journal of Advanced Science and Technology, Volume 29, No 3, pp. 4659- 4670, 2020.

[11] García S, Luengo J, Herrera F. *"Tutorial On Practical Tips of the Most Influential Data Preprocessing Algorithms in Data Mining. Knowl-Based Syst*". 2016. doi:http://dx.doi.org/10.1016/j.knosys.2015.12.006.

[12] Tanupabrungsun S, Achalakul T. "*Feature Reduction for Anomaly Detection in Manufacturing with Mapreduce GA/kNN*". In: 19th IEEE International Conference on Parallel and Distributed Systems (ICPADS). Seoul: 2013. p. 639–44.

[13] Triguero I, Galar M, Vluymans S, Cornelis C, Bustince H, Herrera F, Saeys Y. "*Evolutionary Under Sampling for Imbalanced Big Data Classification*". In: IEEE Congress on Evolutionary Computation, CEC. USA: IEEE: 2015. p. 715–22.

[14] Chen F, Jiang L. "*A Parallel Algorithm for Data Cleansing in Incomplete Information Systems Using Mapreduce"*. In: 10th International Conference on Computational Intelligence and Security (CIS). Kunmina, China: 2014. p. 273-277.

[15] Sandeep Dalal, Vandna Dahiya, "*A Novel Technique - Absolute High Utility Itemset Mining (AHUIM) Algorithm for Big Data"*, International Journal of Advanced Trends in Computer Science and Engineering (IJATCSE), Volume 9, Issue 5, 2020. pp 7451-7460.

[16] Zhang J, Wong JS, Pan Y, Li T. "*A Parallel Matrix-Based Method for Computing Approximations in Incomplete Information Systems*". IEEE Trans Knowl Data Eng. 2015; 27(2):326–39.

[17] D. Jeffrey and S. Ghemawat. "*MapReduce: Simplified Data Processing On Large Clusters*". Communications of the ACM, volume 51, pp. 107-113, Jan. 2008.

[18] Apache Hadoop Project, 2015. https://hadoop.apache.org

[19] Apache Spark: Lightning-fast cluster computing. https://spark.apache.org/

[20] Apache Flink. https://flink.apache.org/

[21] https://en.wikipedia.org/wiki/Apache_Flink

[22] https://pypi.org/project/mLib/