Advances in Artificial Intelligence, Big Data and Algorithms
G. Grigoras and P. Lorenz (Eds.)

doi:10.3233/FAIA230923

Research on Key Technologies of Big Data Analysis

Jianjun CAI[1], Nver REN[2], Erxin SUN[3], Yang TIAN[4], Jingming ZHANG[5]
China Automotive Technology Research Center Co., Ltd.

Abstract. With the advent of emerging computing services such as cloud computing and the decline in hardware prices, the unit price per GB of mechanical hard drives has plummeted by 87%, the amount of information on the Internet has shown explosive growth. Furthermore, the scale of data in large enterprises has reached EB level at present. These massive data form subject-oriented, integrated, relatively stable data warehouses which can reflect historical changes. How to process and analyze these data has gradually become the key technology of Big data. This paper will discuss the key technologies of Big data processing and analysis, such as data preprocessing and Big data analysis framework.

Keywords. EB level, big data, data pre-processing, big data analysis framework

1. Introduction

Big data refers to a collection of data that cannot be captured, managed and processed with conventional software tools within a certain time frame. It is a massive, high growth rate and diversified information asset that requires a new processing model to have stronger decision-making power, insight and discovery power, and process optimization capabilities. Broadly speaking, Big Data has the 4V characteristics: Volume, Velocity, Variability, and Value. Processing techniques for Big Data include preprocessing, Big Data processing framework, etc. Wherein pre-processing involves operations such as extraction, cleaning and integration of collected data. The data processing framework, as the carrier of data mining, completes the mapping and reduction of the data. This article introduces the technology of Big Data preprocessing and its detailed process, as well as the mainstream framework for Big Data analysis.

2. Big Data Preprocessing Technology

2.1. Background

The mainstream big data processing frameworks currently include the early-born

[1]Corresponding Author: Jianjun CAI, China Automotive Technology Research Center Co., Ltd.; e-mail: caijianjun@catarc.ac.cn
[2]Nver REN, China Automotive Technology Research Center Co., Ltd.; e-mail: rennver@catarc.ac.cn
[3]Erxin SUN, China Automotive Technology Research Center Co., Ltd.; e-mail: sunerxin@catarc.ac.cn
[4]Yang TIAN, China Automotive Technology Research Center Co., Ltd.; e-mail: tianyang@catarc.ac.cn
[5]Jingming ZHANG, China Automotive Technology Research Center Co., Ltd.; e-mail: zhangjingming@catarc.ac.cn

MapReduce[1], batch-based Spark, stream-based Storm, Spark Streaming, and Flink[2]. However, in these frameworks, the quality of information obtained through data mining depends not only on the design and performance of the method, but also on the quality and applicability of the data. Negative factors such as noise, missing values, inconsistency and redundancy, as well as the large size of samples and features, significantly affect the information extraction from data[3].

Therefore, data preprocessing[4] is an important and essential step.The main goal of obtaining the final dataset that can be used for further data mining algorithms, as shown in the Knowledge Discovery from Data (KDD) process in Figure 1 .

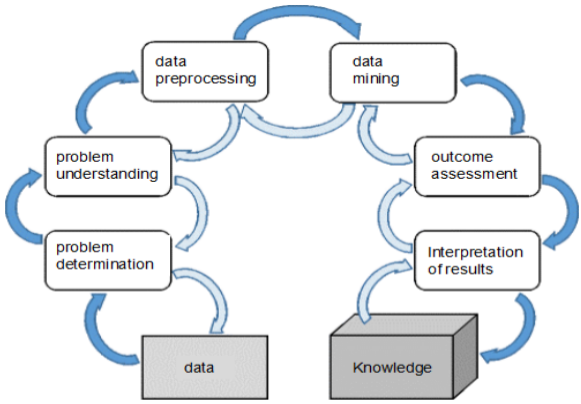

Figure 1. The process of KDD

2.2. *Big Data Preprocessing Technology*

Although data preprocessing improves the quality of information extraction, it can be time-consuming. It requires multiple operations, such as data preparation and data reduction, as shown in Figure 2. The former involves transforming, integrating, cleansing, and normalizing data. The latter aims to reduce data complexity through feature selection and discretization. After the preprocessing phase, the obtained datasets can be considered as reliable and suitable sources for the subsequent data mining algorithms. In this chapter, we will talk about the existing data preprocessing methods.

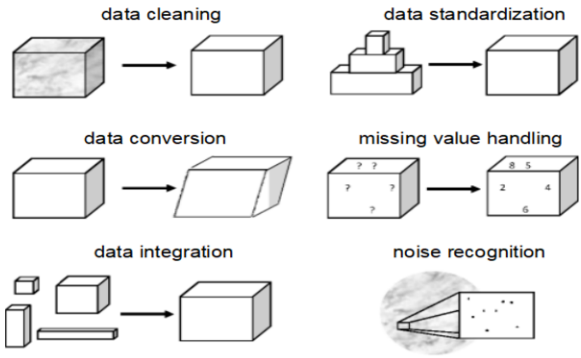

Figure 2. The content of data preprocessing

2.2.1. Missing Value Processing and Noise Processing

Most data mining methods are based on a purportedly complete or noise-free data set. But real data is very unclean or complete. So when preprocessing data, it is usually necessary to remove noisy data or recover missing data.

Data that is not stored or collected due to sampling errors, cost limitations, or certain limitations during the collection process is called missing values.The occurrence of missing values is unavoidable in data analysis and the processing of missing values is complex. Improper handling of missing values can easily affect the extraction of feature knowledge and lead to incorrect conclusions.

The pioneering work on imputation of data comes from statistics. Statisticians model probability functions of data and analyze the causes of missing data. By using a maximum likelihood approach, they sampled the approximate probability model to impute missing values. Because the real statistical model of a specific dataset is often unknown, so machine learning and deep learning methods that do not require prior knowledge have recently become popular research directions.

In order to solve the noise problem in data mining, two methods are usually used to preprocess the data. The first is data smoothing to correct outliers, especially when the noise affects the labeling of examples, but this is usually limited by the amount of noise, otherwise the workload will become unimaginable. The second option is to use noise filtering methods to identify and remove noise instances from the training data, without modifying the data mining.In addition, for data noise in clustering, Outlier can be monitored, and then the Outlier that are farther away from the cluster center than any other data point can be deleted after comparing and analyzing the results according to business logic.

2.2.2. Dimensionality Reduction Processing

Data mining methods encounter dimensionality problems as the number of predictors and dataset instances increases[5]. This is a serious problem since it will slow down data mining techniques as computing costs climb. This section will introduce feature selection (FS) and spatial transformation, the two most often used dimensionality reduction techniques.

The procedure of locating and eliminating unnecessary and duplicate information is known as feature selection (FS)[6]. As demonstrated in Figure 3, the goal of feature selection is to extract a subset of features from the original dataset that accurately characterize it. In general, this subset serves as the model's training set. Feature selection eliminates redundant and pointless information, which could result in unexpected model weights in the learning algorithm and reduce the model's accuracy. In this way, the risk of model overfitting is decreased by the use of feature selection. Furthermore, the feature selection limit the search space dictated by features in learning process, which will be quicker and less memory-intensive.

Contrary to feature selection, spatial transformation techniques combine existing features to create a completely new set of features rather than just choosing the most evident ones. There are many methods for combining original features, such as PCA and factor analysis, etc.

After dimension reduction, it can effectively improve the calculation efficiency, reduce the storage space, improve the Data and information visualization effect and the accuracy of the model.

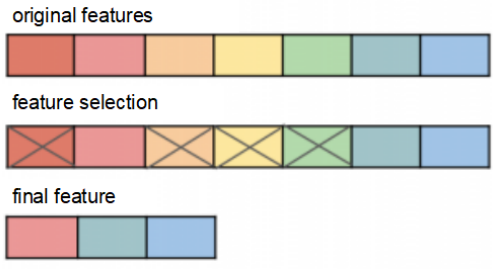

Figure 3. Feature selection

2.2.3. Discretization

Discretization is receiving more and more attention, and it is often used to preprocess data[7]. It converts quantitative data into qualitative data by dividing numerical characteristics into a limited number of non overlapping intervals. Using the generated boundary, each value is mapped to each interval and becomes discrete. Any data mining algorithm that needs fixed class data benefits from Discretization, because many practical applications usually produce real value output. For example, three of the top ten methods in data mining need external or embedded data Discretization: C4.5, Apriori and Naive Bayes.

There are other advantages to discretization, not only the benefits of data simplification and reduction, thus accelerating and improving the accuracy of learning, but also improving the readability of data. Because discrete attributes are usually easier to understand, use, and interpret. However, these benefits come at a cost: any Discretization process will lead to information loss. In order to minimize information loss, Discretization usually tries to select the best segmentation point or interval division method. However, because the optimal Discretization is an NP complete problem, some literatures provide a wide range of alternatives, such as equal frequency, equal distance, clustering and other methods[8]. Although these alternative solutions cannot guarantee finding the global optimal solution, they have been proven to be effective and feasible in practical applications.

3. Big Data Processing Technology

3.1. Background of the Big Data Processing Framework

As Big Data has evolved, many data processing systems have emerged, starting from the first batch processing system MapReduce in the "Google Troika" to Spark, Storm and Flink, which are based on stream processing. In this section, we mainly introduce MapReduce, Spark, and Flink.

3.2. Big Data Processing Framework

3.2.1. Mapreduce

Figure 4 illustrates the various parts of MapReduce. First is the client, which submits jobs to the cluster. The job tracker oversees the execution plan, coordinates the jobs, and

schedules the task tracker. The task tracker decomposes the job into Map and Reduce. Each task records the process of executing Map and Reduce and the output results. Then, the input data is sliced and segmented according to the input format. Input splitting is equivalent to a Map task running in parallel. The input format determines how to parse the file into MapReduce. The Map stage splits the input into temporal key-value pairs according to user-defined code. The Shuffle and Sort stages output and move the intermediate key-value pairs to the Reduce stage and sort them by key. The Reduce stage reduces all key-value pairs associated with the same key, and then generates output according to user-defined code.

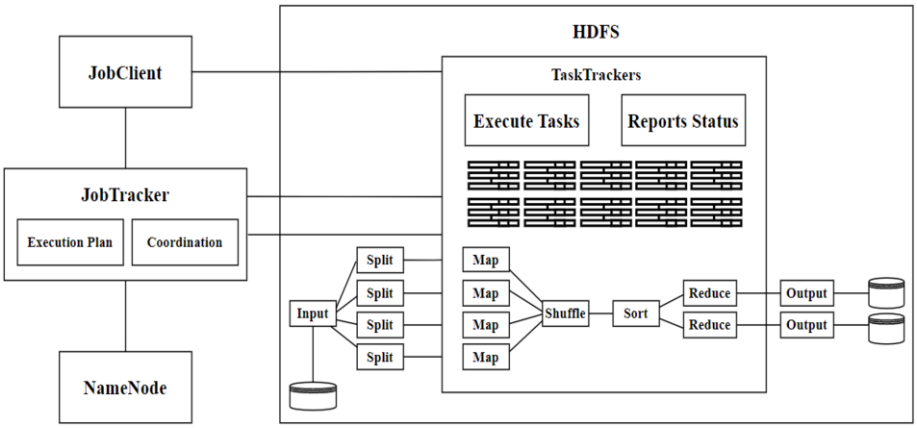

Figure 4. Diagram of MapReduce

3.2.2. SPARK

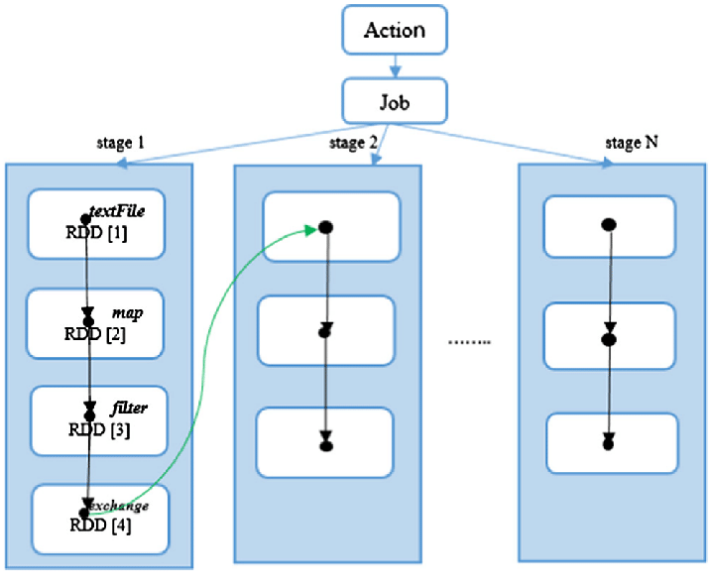

Figure 5. Schematic diagram of Spark reuse data set process

MapReduce and its derivative frameworks have achieved great success in dealing with large-scale data intensive applications, but these systems usually use the acyclic data flow model, so they are not suitable for applications that need to reuse working data sets, such as iterative machine learning algorithms and interactive data analysis tools. To address this issue, a new framework called Spark has been introduced. Spark retains the scalability and fault tolerance of MapReduce and introduces an abstract class called Elastic Distributed Dataset (RDD), which is a collection of read-only objects across multiple machine partitions. If a partition is lost, these objects can be rebuilt, as shown in Figure 5. By using RDD, Spark can efficiently support applications that reuse working datasets and achieve performance tens of times better than MapReduce in iterative machine learning assignments.

Spark comes with a machine learning library Spark MLlib. Spark MLlib consists of common machine learning algorithms and statistical tools. Its main functions include: classification, regression, clustering, collaborative filtering, optimization, and dimensionality reduction. This library is specifically designed to simplify the flow of machine learning data in large-scale environments. In the latest version of Spark, the MLlib library is split into two packages MLlib and ML. MLlib is built on RDDs, while the ML package is mainly used on DataFrames for building data streams.

3.2.3. Flink

A recent open-source framework for distributed batch and stream data processing is called Flink. It focuses on processing large amounts of data with extremely low data latency and high fault tolerance on distributed systems. Real-time data processing of streams is a key function of Flink.

Despite the fact that Spark and Flink both support data reuse and iteration, Spark plans its execution as an acyclic graph plan, which requires it to schedule and execute the same set of instructions in each iteration. On the other hand, Flink's engine uses iterative processing that is based on cyclic data flow (scheduled once per iteration). Furthermore, it offers incremental iteration to benefit from operations that alter only a portion of the data. The state of data flow applications can also be restored using a highly fault-tolerant mechanism provided by Flink. A consistent snapshot of the distributed data flow and operator state is being created by this mechanism. The system can resort to these snapshots if something goes wrong.

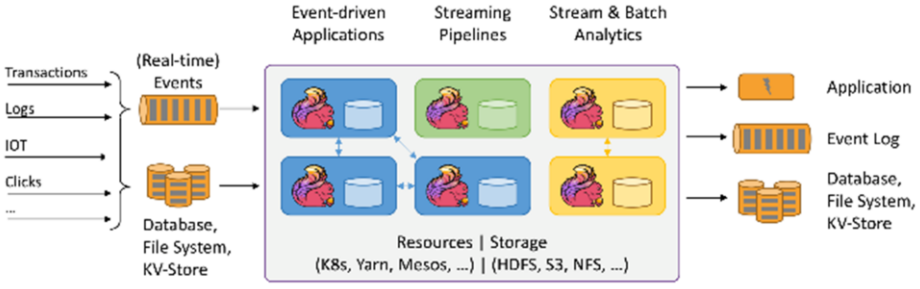

Figure 6. Execution flow of Flink

A schematic of Flink's execution flow is shown in Figure 6. Data sources for Flink include file systems, databases, real-time events, KVStore, and more. Flink programs are represented by a dataflow graph (i.e. directed acyclic graph - DAG) executed on Flink's

core, which is a distributed streaming dataflow engine. Stateful operators and intermediate dataflow partitions make up a dataflow graph. Each operator is executed by a number of parallel instances, the number of which depends on the parallelism level. Each instance of a parallel operator runs in its own independent task slot on a computer cluster machine.

4. Conclusion

By introducing the big data preprocessing process and the conventional analysis framework and highlighting their similarities and differences, this article demonstrates some key technologies in the big data processing process. Big data is a growing sector with enormous potential for service and economic value. Big data practitioners should constantly raise their own standards, utilize big data technology to its fullest potential, increase productivity, and support economic growth.

References

[1] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters[J]. Communications of the ACM, 2008, 51(1): 107-113.
[2] Chintapalli S, Dagit D, Evans B, et al. Benchmarking streaming computation engines: Storm, flink and spark streaming[C]//2016 IEEE international parallel and distributed processing symposium workshops (IPDPSW). IEEE, 2016: 1789-1792.
[3] Pyle D. Data preparation for data mining[M]. morgan kaufmann, 1999.
[4] García S, Luengo J, Herrera F. Data preprocessing in data mining[M]. Cham, Switzerland: Springer International Publishing, 2015.
[5] Bellman R, Kalaba R. On adaptive control processes[J]. IRE Transactions on Automatic Control, 1959, 4(2): 1-9.
[6] Hall M A. Correlation-based feature selection for machine learning[D]. The University of Waikato, 1999.
[7] Liu H, Hussain F, Tan C L, et al. Discretization: An enabling technique[J]. Data mining and knowledge discovery, 2002, 6(4): 393-423.
[8] Carbone P, Katsifodimos A, Ewen S, et al. Apache flink: Stream and batch processing in a single engine[J]. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 2015, 36(4).