POLITECNICO DI TORINO

DEPARTMENT OF Software Engineering

College of Computer Engineering, Cinema and Mechatronics

Master's Degree in
COMPUTER ENGINEERING (SOFTWARE ENGINEERING)

Master’s Degree Thesis

BIG DATA AND CLUSTER QUALITY INDEX COMPUTATION

Supervisor

Prof. Garza Paolo

Prof. Cerquitelli Tania

Candidate

Aynadis Temesgen Gebru

DECEMBER 2019

Abstract

Clustering analysis is an unsupervised machine learning technique that partitions a dataset into multiple groups or clusters so that instances in the same cluster have high similarity but not with instances of other clusters. The quality of the results generated by a clustering method is measured in cluster evaluation phase using Cluster quality indices. Some clustering methods demand the number of clusters into which data is going to be partitioned. Cluster quality indices can also be used to determine the number of clusters to be provided as an input to the clustering methods. The traditional clustering metrics are not applicable for Big Data due to a size limitation and run time cost. This work introduces a method for evaluating huge size dataset.

This document presents a technique that assists the evaluation process of large amount of data. It proposes a sampling approach using Big Data analysis to reduce the size of the dataset so that the traditional clustering validity indices can process it. Silhouette validity index is selected and adopted to test the sampling result. The sampling technique positions instances in the space which is divided into cells of same size. It iterates through each cell to verify the cell meets a criterion. The reduction is carried out only on those cells containing instances from the same cluster and it provides a weight associated to each newly generated instances of the cells.

The evaluation of the implementation is made on both manually and automatically clustered datasets. In the manually clustered dataset, three datasets containing 18000, 6500 and 3000 instances with 5, 8 and 31 number of clusters respectively were used. Single dataset is used in the auto-clustered dataset with 8000 instances. To observe the performance, a comparison between the original datasets and the average of three different percentage of random sample are done. Generally, silhouette index on smart sampled and original dataset, on all the datasets, are very similar, with slightly higher index of the original dataset. This indicates the smart sampling can be considered as a solution for the cluster evaluation of huge size dataset. The performance of the average random sample shows an equal or slightly higher index than both the smart the sampled and original dataset. However, it is important to consider that finding the average silhouette index on random sample requires multiple executions for a single dataset and results vary on each run while the smart sampling is executed once per dataset which is a good point for the smart sampling.

Keywords: Big data, Cluster Evaluation, Cluster Quality Index, Sampling based cluster evaluation, Clustering Big data, Silhouette index

Acknowledgments

I would like to thank my advisor, Professor Paolo Garza and my co-advisor professor Tania Cerquitelli, for their support, assistance, understanding and motivation throughout this thesis work. My deepest gratitude goes also to my family, specially my bother Befkadu Temesgen and my husband Ivan Akono, for their help with all they can from the very beginning of my journey to this master. The last but not the least is my daughter Makeba and my mom Abeba for the pure unconditional love which gives me strength to continue when I feel like giving up.

Table of Figures

Index of Tables

Contents

Chapter One

Introduction

1.1 Background

Nowadays, the data is growing at a rapid pace because of an increasing number of people, organizations and machines producing data and sharing information in different ways such as online web applications of organizations, entertainment images and video, security devices, files created on personal computers, log files and metadata, internet of things. This data is generated in high velocity, large volume, and a wide variety and it becomes an issue for data storage and analysis [1]. This is mainly because the data is so huge to be stored in a single machine and complex to be processed in the traditional data-processing application software [2]. Cluster analysis and cluster evaluation are examples for challenge in Big Data analysis.

The objective of cluster analysis is to divide data into meaningful and/or useful groups (clusters) in such a way that instances within the cluster are similar to one another with respect to considered variables whereas instances of two different clusters are dissimilar. Therefore, Clusters should exhibit high internal homogeneity and high external heterogeneity. It can be a standalone tool to get insight to the data distribution or used as a preprocessing to other data analysis algorithms such as data summarization. Business, medicine, climate, bioinformatics, information retrieval and education are some of the fields that cluster analysis applied in [3, 4]. It is an unsupervised learning which has dataset as an input, no predefined classes and the composition of the group is determined in the process. The task of cluster analysis can be achieved using various algorithms including partitioning methods, hierarchical clustering, density-based clustering and model-based clustering. Evaluating clustering result is one of the challenges mentioned related to clustering analysis [5]. Visual inspection can be taken as an evaluation method, but it is impractical as real dataset are high dimensional and bigger in size. Cluster Quality Index (CQI), which is considered as one of the success of clustering applications, is a common approach used for evaluation [5, 6].

Cluster Quality indices is crucial to determine the number of clusters in a dataset and measure the clustering quality. In addition to that, CQI can be used to compare how well different clustering algorithms perform on a set of data [7]. Cluster quality indices can be categorized into

three classes; external, internal and relative clustering validation. External clustering validation requires external information or ground-truth information of data which is not a part of the data. Whereas, the internal CQIs performs the evaluation without any external information [8].

Nowadays, there are an increasing number of distributed systems and Big Data frameworks on the market that can be used to store and process Big Data in real-time or nearly real-time. Apache Hadoop is among these frameworks [9]. It is a framework that allows storing Big Data in a distributed environment in order to process it in parallel. There are basically two components in Hadoop: Hadoop Distributed File System (HDFS) and Hadoop MapReduce. HDFS is used as a primary data storage and MapReduce is used to perform all the necessary computations and data processing across the Hadoop cluster [10]. In addition, Apache Spark is one of the most widely used open-source cluster computing frameworks [9, 11]. Apache Spark is a framework for real time data analytics in a distributed computing environment. It has built-in modules for streaming, machine learning, graph processing and SQL support. It provides near real-time, in-memory Big Data processing. Resilient Distributed Dataset (RDD) is a fundamental data structure which is an immutable distributed collection of objects designed for parallel computing and in memory processing of large amounts of data [9, 12]. Spark is 100 times faster than MapReduce [13]. This is mainly because it stores intermediate data in faster logical RAM memory and reduces the number of read/write cycles to disk by executes computations in-memory [13]. Due to it is processing speed, Fraud detection, log processing, and trading data becomes easier with Apache Spark.

The purpose of this paper is to fill the gaps of traditional cluster evaluation indices in the process of evaluating a large size dataset by presenting a smart sampling approach which works with an adopted traditional index to support the sampled dataset. The proposed approach is implemented using Apache Spark framework and an adopted silhouette cluster validity index to support the sample generated using spark. The k-means method was selected for testing the performance. A comparison with the randomly generated samples also performed.

1.2 Motivation

Big data is a term that is used to describe data that is big in size: *high volume*, created in a very fast rate: *high velocity* and different types of data: *high Variety*. This is known as three V's of big data. Big Data is not defined with the three V's, However, it also described with additional two V's: *Veracity*: trustworthiness of the data and *Value*: the worth of the data. Terabytes of

data was considered big years back, which is not the case anymore and what is considered big now may not be so big in the near future. This massive volume of data address problems that were difficult to tackle before in analytics, business intelligence, data mining, machine learning, and pattern recognition. Big Data can be analyzed to extract information for insights that lead to better decisions and important business moves. Speaking of big data, Data storage is not the only challenge but also designing tools to perform operations (like analytical, process and retrieval operations) in order to interpret and get value from such a massive amount data [1]. To get the best out of big data, there should be a technology and infrastructure to store manage and analysis it.

Clustering analysis, main task of exploratory data mining and a common technique for statistical data analysis, can be used as a standalone tool to get insight into the data distribution or as a pre-processing step for other algorithms. There are two types of Big Data clustering techniques, single machine and multiple machine clustering [14, 15]. Single machine clustering is performed on one node using data mining algorithms or dimension reduction. Techniques from data mining are well-known knowledge discovery tools to analyze and reveal valuable knowledge that is hidden within the data. Data mining clustering algorithms are considered essential for Big Data analysis [14, 15, 16]. On the other hand, multiple machine clustering can be performed by partitions in a distributed environment which speeds up the calculation and increases scalability. Parallel clustering and MapReduce-based clustering is mentioned as multiple machine clustering in [14, 15]. When these Clustering methods are applied on a dataset, the generated clustering result needs to be evaluated for several reasons [3]. One of reasons is measuring clustering quality, accessing the quality of the cluster generated by the clustering methods. Determining the number of clusters in a dataset can also be mentioned as other reason why cluster evaluation is performed on a clustering result [3].

Given tremendous amount of clustered data, efficient and effective evaluation tools need to be present. Data provided by the Big Data is beyond the capacity of the traditional cluster quality indices. The question that arises is how to deal with this problem and evaluate results produced from clustering techniques. Taking this issue in to consideration, it is necessary to propose a solution which provide methods for evaluating Big Data clusters.

1.3 Statement of the problem

When a Big Data is clustered, the instances are dispersed over different nodes in the network. The clustering result cannot be gathered in a single machine for evaluation due to its size. In addition to that, the traditional evaluation algorithms, which are centralized and has capacity limitation, are unable to be applied directly on Big Datasets [14, 15]. Therefore, it is important to design solution to evaluate the clustering result applied on Big Data cluster.

This work tried to answer the following questions;

- Can sampling be an option for Big Data cluster analysis?
- Will increasing or decreasing of percentage of the sample have a relation with performance?
- Can the proposed sampling method be used as Big Data cluster evaluation?
- Which one of this method performs better and in which conditions?
- Which one of the methods be recommended for Big Data cluster evaluation?
- How is the performance of both sampling methods on manually and automatically clustered dataset?

1.4 Objective

1.4.1 General objective

The general objective of this thesis is to design and develop a cluster evaluation system for a Big Data by reducing the original large size dataset into a smaller one in order to perform the evaluation on traditional cluster quality indices.

1.4.2 Specific objective

The specific objectives of this project are to:

- review literatures on Big Data analysis, clustering analysis, cluster evaluation and other related researches conducted on Big Data clustering analysis and clustering quality index;

- develop an algorithm for proposed approach used to perform Big Data cluster evaluation;
- implement the algorithm; and
- evaluate the system using collected test sets

1.5 Scope and limitation of the study

This study only covers automatic Big Data cluster evaluation. For comparison purpose, automatically clustered datasets are also used during the experiment in addition to the manually clustered datasets. This work tried to include dataset cleaning and preprocessing before use. The scope of this study is limited to evaluating an already clustered big dataset.

1.6 Methodology

1.6.1 Literature review

Related works will be reviewed to get a deeper understanding about clustering analysis, cluster evaluation, Big Data analysis and fundamental concepts related to this work. A review on different approaches of cluster evaluation will also be made to identify and understand the concept related and their advantages and disadvantages.

1.6.2 Data collection

The test dataset will be selected to evaluate the proposed solution. The collected dataset contains both clustered and un-clustered dataset. The selected un-clustered dataset will be automatically clustered during the experiment.

1.6.3 Prototype development

To implement the proposed solution, clustering analysis tool or clustered dataset is required to facilitate the experiments. The implementation of Big Data clustering evaluation will be done using Apache spark (version 2.2.0) for the data analysis and Apache Hadoop (version 2.7) for HDFS data storage. Spark jobs will be written in Java (version 1.8). The silhouette implementation will also be written in Java.

1.6.4 Evaluation

After implementing the proposed algorithm, the prototype developed will be tested using the test set prepared for this purpose. The performance of the proposed solution is determined by comparing the silhouette index of the proposed sampling method with the random sample dataset and the original dataset.

1.7 Application of result

As this work mainly concerned with evaluating results for cluster analysis for large size dataset, it can be applied wherever clustering analysis is applied in order to enhance the performance of clustering.

1.8 Organization of the thesis

This section describes the organization of the rest of the thesis. Chapter two discuses fundamental concepts, different approaches and techniques related to clustering analysis, cluster evaluation, Big Data and its technologies. It also presents related works to this study. Chapter three describes the approach used in this research in detail. The experiment and results are discussed in chapter four. The last chapter, chapter five, presents the conclusion and recommendation based on the experiment and results.

Chapter Two

Literature Review

2.1 Introduction to Big Data

Data has been slowly growing over the last few centuries, however in the course of the past decade, Big Data has quickly evolved to become as massive as it is today. Big Data is not only about gathering and storing massive amounts of information but, more importantly, applying that information to resolve issues in business or society [1, 17]. Big Data appears to develop at the same time with advancement in technology. Therefore, as the technology advance, Big Data keep on growing in volume and as a field.

Big Data is often described as tremendously large datasets that cannot be stored in single machine and are beyond the ability to be managed and analyzed with traditional data processing tools. The challenges of big data management result from the enlargement of all three properties, rather than just the volume alone. Generally, the term “Big Data” refers to three V’s: volume, velocity and Varity. Figure 2.1 shows the classic characterization of big data which includes another two important V’s: Veracity and Value.

Volume

The amount of data is one of defining properties of big data. A massive amount of data is created each second by number of means such as searches on search engines, videos, photo and texts on social media, structured records of companies' databases and the like. Other than human generated data, machine logs and sensor data are another source of data. 90 percent of the data today is generated over the past couple of years. The volume of Big Data depends on a particular business requirement. For one company or system, 100TB may be considered as Big Data and it may be 10PB for another.

Variety

Variety refers to types of data available. Traditional data types (number, text, time, date...) were structured that fit a rational database whereas now it doesn't easily fit into fields on database

applications. The data types that should be handled are not only structured but also semi structured or/and unstructured (social media feeds, audio, video, images, web pages...) which need additional effort to give meaning to it.

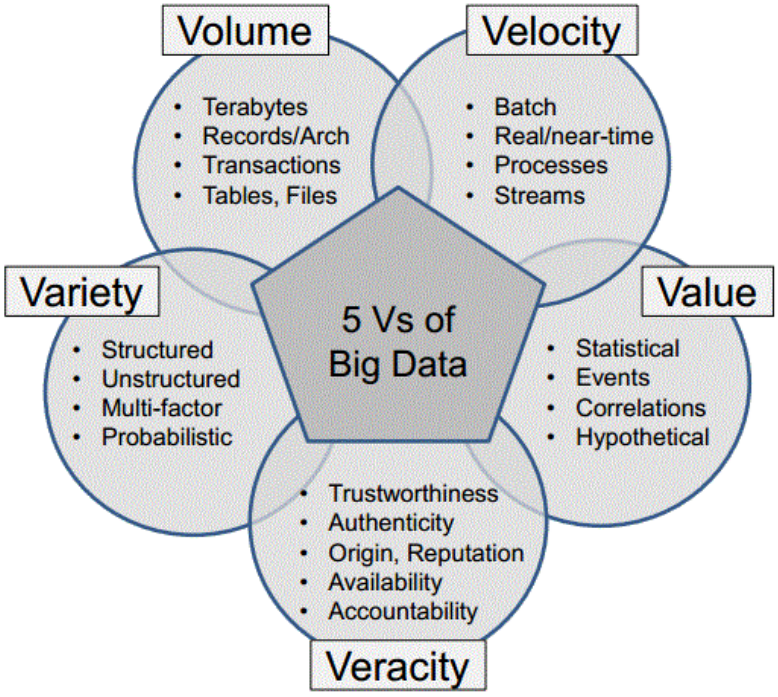

Figure 2.1 The five V's of big data

In general, Big Data can be found in three forms; unstructured, structured and semi structured. Structured data refers to data stored in an ordered manner and designed according to a predefined data model. This type of data is relatively straightforward to enter, store, query, and analyze. Structured data corresponds to a tabular format with relationship between the different rows and columns. Some of the examples of the structured data includes relational databases and excel files. All of these have structured rows and columns that can be grouped. On the other hand, unstructured data is information that either does not conform neatly into a predefined data model or is not arranged in a pre-defined manner. It is not a good fit for relational databases. It basically contains text-heavy which creates irregularities and ambiguities that make it challenging to understand, analyze and drive value out of it compared to data stored in structured databases. Common examples of unstructured data include Word, PDF, Text, Media logs, audio, video files or No-SQL databases. Semi-structured data is a form of structured data that does not designed with the formal structure of data models related to data tables, but some organizational properties that make it easier to analyze. It contains tags or other markers to

separate semantic elements and implement hierarchies of records and fields within the data. Common examples of semi-structured data include JSON and XML are forms of semi-structured data.

Velocity

Velocity refers to the speed of creation, collection and processing of data. Considering Facebook as an example, massive amount of data images and videos are being uploaded, processed, stored and retrieved in seconds. Some activities are time sensitive and requires real time data processing. For instance, to protect an organization from fraud detection, the flow must be processed as it's streamed to increase the data protection.

Value

Value refers to the worth of the data being extracted. It is nothing to have huge amount of data if it cannot be turned into value which is the ability to transform a highly growing data into business. While there is a clear relationship between data and observations, this does not always mean Big Data has a value. The most important issue is to recognize the costs and benefits of gathering and processing the data to guarantee that the collected data can be monetized in the end.

Veracity

This refers to the trustworthiness and accuracy of the data. The higher the volume of the data and data source, the more it becomes uncertain. This leads to serious of data quality which causes inaccurate data analysis and wrong decisions. It can be difficult to trust the accuracy of rapid analysis and change of the information with data of high volumes, from various sources and such high speeds. Traditional Data management techniques provide a consistent and usually accurate solution by means of a structured databases and data warehouses. Nowadays, as the data can be real time, it can be difficult to find a clear, verified and formatted data. Big Data involves working with all degrees of quality data, reliable source and robust algorithms.

2.2 Advantage of Big Data

Organizations have a long practice of capturing and storing transactional data. In addition to that, organizations currently are capturing other data at an increasingly fast speed from operational environment such as web data(page views, searches, purchasing, etc.), text data (email,

news, social media feeds, etc.), time and location data (GPS, mobile phone and Wi-Fi connection) and sensor data(cars, oil pipes, windmill turbines).

Ability to capture and process Big Data brings in number of advantages. Using web data, Organizations can increase performance in areas such as next best offer, targeted advertisement, churn modeling and customer segmentation. Customer service can be improved using Big Data and natural language processing technologies by utilizing to read and evaluate consumer responses from customer feedback systems which are designed with Big Data technologies now a days. Businesses can utilize external intelligence during decision making by getting access to web and social data. Many organizations are realizing the power of knowing where customers are at a particular period, but this is privacy-sensitive types of Big Data and should be treated with great attention. Moreover, Key facts can be extracted from the text data and then used as inputs to other analytic process (for example, fraud detection in insurance claims) [1].

Sensor data can be used to diagnose problems on engines and machinery more easily for faster development of mitigation procedures. Big Data technologies can be used for early identification of risk to the product by creating a landing zone for new data before identifying what data should be moved to the data warehouse. Furthermore, organizations can offload infrequently accessed data by integrating of Big Data technologies and data warehouse.

2.3 Big data analysis technologies

2.3.1 Apache Hadoop

Apache Hadoop is a framework used for distributed storage and distributed processing of very large datasets distributed across clusters of commodity computers. It is an open-source software written in Java. Hadoop follows a master slave architecture design. The main components of Hadoop are Hadoop Distributed File System (HDFS) and Hadoop MapReduce for distributed data storage and distributed data processing respectively [11, 13] .

HDFS gives access to files and directories to the user application. The files and directories are stored over different machines on the network. In HDFS, the actual data and metadata are kept separately on dedicated servers. HDFS has two important components: NameNode and DataNode. NameNode is a single master server in the cluster and it stores the metadata which is the directory tree of all files in the file system to track the files across the cluster. All metadata

operations on the file system such as creating, opening, closing or renaming files and directories are served by the NameNode. A file is split into same size data blocks and data blocks are stored DataNodes and a list of blocks and their location are stored in NameNode. Therefore, DataNode, which is also known as slave, stores the actual data. HDFS cluster contains one or more DataNodes and it replicates the file content on these DataNodes for fault tolerance. All decisions regarding replication are left for the NameNode. It periodically receives information from each of the DataNodes in the cluster. This insures the proper function of the DataNode [11].

The core architectural goal of HDFS is detection of hardware failure, which assumed as a common failure, and fast recovery from it without any interaction by the user. It's easily portability property across heterogeneous hardware and software platforms assists its adoption as a platform of choice for a large set of applications [11].

MapReduce is a program model within the Hadoop framework for accessing and processing a huge data stored in the Hadoop File System (HDFS). MapReduce programs can be written in various languages such as Java, Ruby, Python, and C++. MapReduce performs analysis by splitting huge size of data into smaller chunks on a huge dataset and process the chunks in-parallel using multiple machine in the cluster in a reliable, fault-tolerant manner. The MapReduce algorithm contains two important tasks or functions, namely Map and Reduce. Map takes a set of data as a key/value pair, performs processing and produces another intermediate set of data as a key/value pair. Initially, the input data is divided into fixed-size smaller blocks and each block is assigned to a mapper for processing. When all the mappers finish processing, the reducers receive shuffled and sorted results. Shuffling consolidates the relevant records from mapping output. Reducer combines a set of intermediate values, which have the same a key to a smaller set of aggregated values. All the map output values that have the same key are assigned to a single reducer. The Reducer result will be stored in the HDFS. The mapper task is always performed before the reducer i.e. a reducer cannot start while a mapper is still in progress. Figure 2.1 shows the architecture of Hadoop.

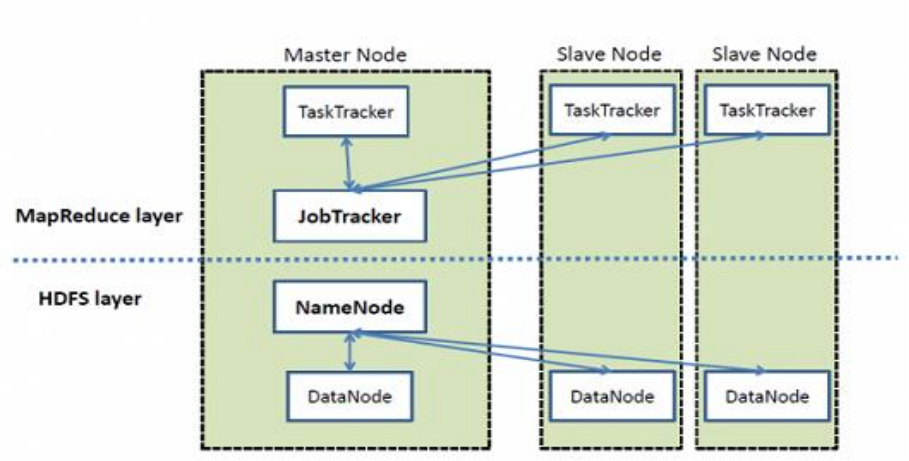

Figure 2.2 High level architecture of Apache Hadoop

2.3.2 Apache Spark

Apache Spark is a general-purpose cluster computing system for real-time distributed data processing. Its in-memory computations feature implies an increase in application processing speed and making it desirable for everyone interested in Big Data analytics. It supports Programming languages such as Java, Scala, Python, and R. It has libraries for SQL (structures data processing), machine learning, graph processing, and stream processing. Spark can work standalone or run on an existing cluster manager.

2.3.2.1 Apache Spark Architecture

Driver program is in the master node and drives the application. The code that the user writes or interactive shell that user use behaves as a driver program. When a client submits spark user application code, the driver implicitly converts user code into a logically directed acyclic graph called DAG. Then, the driver converts DAG into physical execution plan with many stages. After conversion, Tasks which are physical execution units are created under each stage. The tasks are collected and transmitted to the cluster. The driver communicates and negotiates with the cluster manager about the resources. Cluster manager sets up executors in worker nodes on behalf of the driver. At this point, the driver sends the tasks to the executors. The executors register with drivers during the beginning of the execution in order the driver to have a complete view of executors that are executing the task. Driver program monitors the set of executors that runs, and schedules future tasks based on data placement. Figure 2.3 show the architecture of spark which shows the relation between different components.

The very first task performed in the driver program is creating a Spark Context. The Spark context is a gateway to all the Spark functionalities. It is similar to your database connection. Spark context and cluster manager work together to manage number of jobs. The driver program & Spark context manages the job execution within the cluster. A job is divided into multiple tasks and distributed over the worker node. The Tasks are executed by the worker nodes, which are slave nodes. The tasks work on the partitioned RDD and perform operations. Then, results are collected and returned back to the spark context.

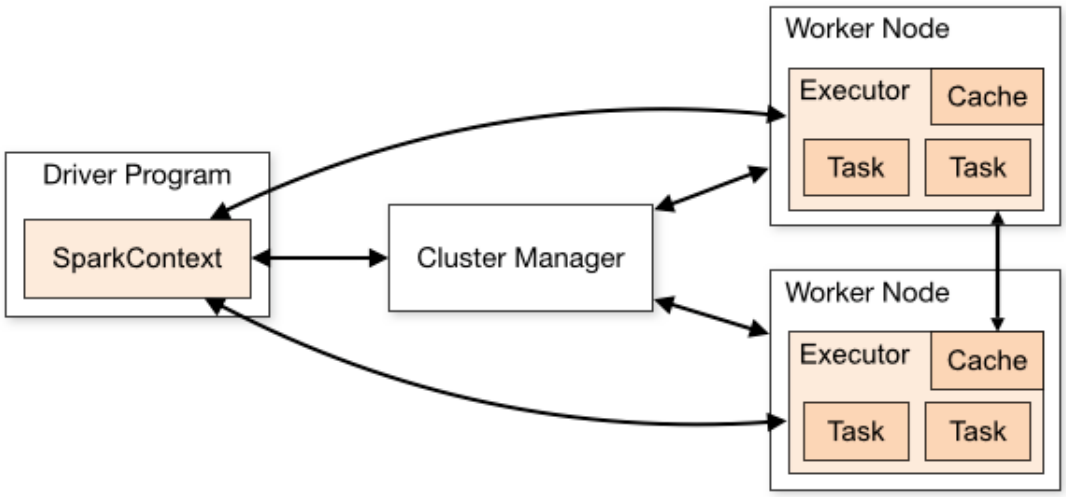

Figure 2.3 Apache Spark Architecture

2.3.2.2 Apache Spark API's

There are three main APIs in spark: RDD, Data Frame and Dataset. Each one of these APIs discussed as follows:

RDD (Resilient Distributed Dataset)

RDD (Resilient Distributed Dataset) is the fundamental data structure of Apache Spark to represent data in the Spark memory. "Resilient", from the name, describes its fault-tolerant behavior. If there is a node failure, the missing or damaged partitions can be recomputed with help of RDD lineage graph. "Distributed" implies resides on multiple nodes in a cluster. "Dataset" is a collection of partitioned data with primitive values or values of values, e.g. tuples or other objects. The Objects are collection of statically typed immutable objects which computes on different nodes of the cluster. Immutable means its state cannot be modified after it is created, but it can be transformed to another RDD. RDD is designed to address issues in a distributed environment like expensive remote data access, high chance of failure, expensive wasting

computing power and difficulty tracking runtime errors. An RDD can be created by parallelizing an existing collection in your driver program, or referencing a dataset in an external storage system, such as a shared file system, HDFS or any data source offering a Hadoop Input Format. A faster data sharing across parallel jobs is required by both Iterative and Interactive applications. Due to replication, serialization, and disk IO, MapReduce data sharing is slow. Performing HDFS read-write operations takes more than ninety percent of the total time in most of the Hadoop applications. The concept of RDD in Spark is used to achieve faster and efficient MapReduce operations. Intermediate results in iterative operation of spark are stored in a distributed memory instead of Disk storage which makes the system faster. If there is more than one query run on the same set of data repeatedly (interactive operations on Spark RDD), this particular data can be kept in memory for better execution times.

There are two main operation performed on RDDs: Transformation and action. Spark Transformation is an operation that produces new RDD from the existing RDDs. It takes the input RDD data and transforms it to one or more output RDD of another form. Transformation is a lazy operation because it will not perform the operation immediately. It keeps on constructing DAG (Directed Acyclic Graphs using the source RDD and function used for transformation., all the transformation that form the RDD, in which the action is applied on, are executed based on DAG when an action operation is performed. Actions, unlike transformation, do not form RDDs. Instead works on the actual dataset to generate non RDD value which will be stored in a driver or to the external storage system. When an action is takes place, data is sent from Executer to the driver. Executors are agents that are responsible for executing a task and the driver is a JVM process that coordinates workers and execution of the task. Example transformations include map, filter, distinct, and groupBykey. Example actions include count, top, reduce, fold or writing data out to file systems [11, 13].

The distribution of data within the cluster is performed using Java serialization by default. This requires sending both data and structure between nodes. The drawback of RDDs is an overhead in serializing Java and Scala objects and on garbage collection that results from creating and destroying individual objects [11, 13] .

DataFrame

Unlike RDD, Data is organized as a distributed collection of data into named columns called DataFrame. Fundamentally, it is conceptually similar a table in a relational database or data frame in R/Python. It does not run directly on spark context but on the SQL context. A wide

array of sources (which include structured data files, tables in Hive, external databases) can be used to construct DataFrame. It allows spark to manage the schema in order to pass the data between nodes efficiently than Java serialization [11, 13] .

The lack of type safety is an in issue in DataFrames. The schema that represents the data holds the column names but not the column types. As the code refers to the name of data attributes, it is impossible for the compiler to detect errors. The users are expected to cast the values to the expected type. For incorrect attribute names, the error will only be detected at runtime [11, 13] .

Dataset

A Dataset is a strongly typed collection of domain-specific objects. The objects can be transformed in parallel using functional or relational operations. Dataset fills the gaps of DataFrame by adding type safety to it. It runs on the SQL context and provides a similar syntax as that of RDD (including Operations like transformations and actions) with lambda expressions. Like RDD, Transformations produce new Datasets, and actions trigger computation and return results [11, 13]. Table 2.1 compares the three spark APIs (RDD, DataFrame and DataSet) based on various features such as Data Representation, Immutability, and Interoperability etc.

Feature	RDD	DataFrame	DataSet
Included since Spark Release	Version 1.0	Version 1.3	Version 1.6
Data Representation	distributed collection of data in the cluster	distributed collection of data organized into named columns	extension of DataFrame API with type-safe functionality
Data format	Structured and unstructured but NO Schema	structured and semi-structured data	structured and unstructured data
Data Sources API	Any e.g. text file, a database via JDBC etc	Different formats e.g. AVRO, CSV, JSON, and HDFS	Different formats e.g. AVRO, CSV, JSON, and HDFS, MySQL
Optimization	No inbuilt optimization engine	Uses catalyst optimizer	Includes the concept of Dataframe Catalyst optimizer for optimizing query plan
Seri1alization	use Java serialization	Uses off heap memory for serialization	performing the operation on serialized data

Table 2.1 Comparison between RDD, DataFrame and DataSet

2.3.2.3 Apache Spark Ecosystem

Apache Spark Ecosystem consists of six basic components which are Apache Spark Core, Spark SQL, Spark Streaming, Spark MLlib, Spark GraphX, and SparkR (see Figure 2.4).

Apache Spark Core

All the functionalities being provided by Apache Spark are built on the top of Spark Core. It delivers speed by providing in-memory computation capability. It provides distributed task dispatching, scheduling, and basic I/O functionalities. This is possible through an application programming interface (for Java, Python, Scala, and R). Thus, Spark Core is the foundation of parallel and distributed processing of huge dataset.

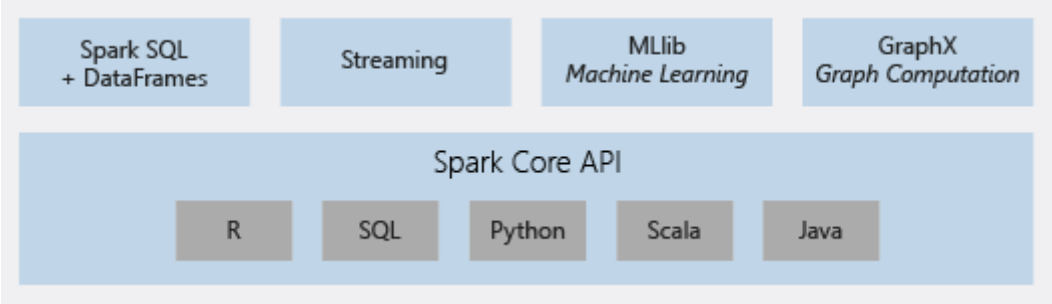

Figure 2.4 Apache spark Ecosystem

Spark SQL

Spark SQL is a module in Spark for working with the structured data. It integrates relational data processing with Spark's functional programming API. Using standard interface, it's possible to query structured data inside the spark program. It provides a uniform way to access a variety of data sources and perform join between the data sources such as Hive, JSON and JDBC. It also integrates with the rest of the Spark ecosystem such as machine learning.

Spark Streaming

An early addition to Apache Spark, Spark Streaming makes it easy to build scalable fault-tolerant streaming applications. It enables data engineers and data scientists to process both real-time and historical data from wide variety of popular data sources including Kafka, Flume, and Amazon Kinesis. Spark Streaming is an extension of concept of Apache Spark batch processing into streaming by dividing a stream of data into small series of batches. Spark Streaming is different from other traditional streaming systems. Some of the major benefits over traditional streaming systems are rapid recovery from failures, better resource usage, and integration of streaming data with static datasets and advanced processing libraries including SQL, machine learning, graph processing.

MLlib

MLlib is Apache Spark's scalable machine learning library. It is designed for simplicity, scalability, and easy integration with other tools. It provides a framework for creating machine learning pipelines, allowing for easy implementation of feature extraction, selections, and transformations on any structured dataset. Spark MLlib smoothly integrates with other Spark components such as Spark SQL, Spark Streaming. It includes the common algorithms such as classification, clustering, regression, dimensionality reduction.

GraphX

GraphX is Apache Spark's API for graphs and graph-parallel execution. It provides simplified graph analytic using growing collection of distributed algorithms for processing graph structures. It is network graph analytics engine and data store. GraphX provides an optimized way to represent vertex and edges as primitive data types. It supports fundamental operators (like subgraph, join Vertices, and aggregate Messages) to support graph computation.

SparkR

It is an R package to use Apache Spark from R. It provides a distributed data frame implementation. It provides a light-weight frontend that supports operations like selection, filtering, aggregation but on large datasets. SparkR also supports distributed machine learning using MLlib.

2.4 Clustering Analysis

Cluster analysis is a method which aims to partition instances/objects into groups such that similar objects are placed in the same group and objects in different group are dissimilar as much as possible. Clustering analysis can be a standalone tool as a data mining function. When it is used as a standalone tool, it helps to gain insight into the distribution of data, to observe the characteristics of each cluster, and to focus on a particular set of clusters for further analysis. In addition, it may assist other algorithms as preprocessing step. Characterization, attribute subset selection, and classification can be mentioned as an example for these algorithms that works further on the detected clusters of the clustering analysis [18, 19].

2.4.1 Application of cluster analysis

Cluster analysis is one of highly active topics in data mining research. It contributes to the areas of research include data mining, statistics, machine learning, spatial database technology, information retrieval, Web search, biology, marketing, and many other application areas. The following are some examples based on the area of application:

Medicine: Different subcategories of medical condition, such as different types of depression, can be identified by cluster analysis. It can be used to differentiate between different types of tissues and blood in Medical imaging. It is also used in the analysis of antimicrobial activity to identify the patterns of antibiotic resistance.

Biology: Clustering techniques are used for extracting/analyzing the biological structures such as categorizing gene with their functionality, detecting different gene expression.

Information retrieval: clustering analysis is applied in search engines for higher efficiency and faster search. A key word in a search may return a very large number of pages relevant to the search. Clustering can be used to group these thousands of pages returned as search results into a small number of clusters, each of which captures a particular aspect of the query.

Business intelligence: clustering can be used to organize many clients into cluster, where clients within a cluster share strong similar behavior. As all clients has not equal profit to an organization, clustering analysis together with customer lifetime value can be used to categorize customers in order to set marketing strategies. It can also be used to group items on the web into a set of products. In retail businesses, data clustering helps with customer shopping behavior, sales campaigns and customer retention. In the insurance industry, clustering is regularly employed in fraud detection, risk factor identification and customer retention efforts. Customer segmentation, credit scoring and analyzing customer profitability are also some of the areas in banking that clustering can be applied in.

Social science: By identifying areas where greater frequency of specific types of crime over a specified time slot occur, it is possible to come through law enforcement resources more effectively. In crime analysis, Cluster analysis can be used to identify these areas.

2.4.2 Cluster analysis algorithm

It is difficult to clearly categorize each clustering methods as a method may contain major clustering approaches characteristic from various categories [18]. However, providing a relatively organized picture of clustering methods is important. Categories of clustering algorithms is shown in Figure 2.5.

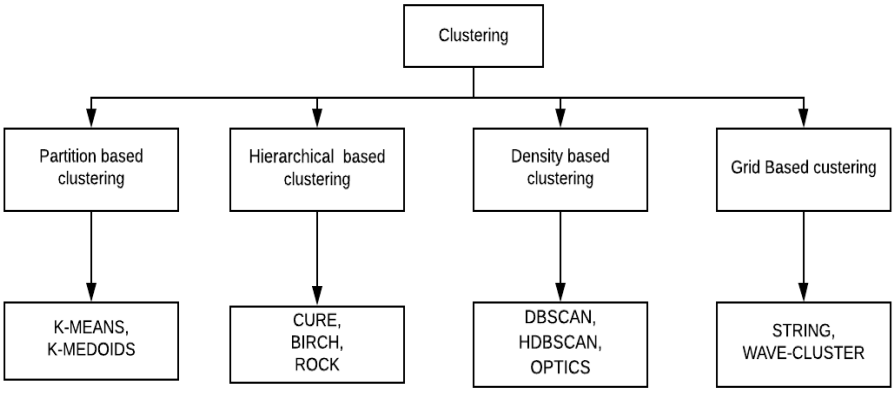

Figure 2.5 Catagories of Clustering Algorithms

2.4.3 Partitioning algorithms

It constructs the instances of the set into various non overlapping groups or partitions. Therefore, each instance is assigned to exactly one partition and each partition representing a cluster. The clusters are formed to optimize an objective partitioning criterion, such as a dissimilarity function based on distance, so that the objects within a cluster are "similar" to one another and "dissimilar" to objects in other clusters in terms of the dataset attributes. It is the simplest and principal version of clustering analysis. Partitioning algorithms are also badly affected by the existence of noise and outliers in the data. Formally, given a dataset, D, of n instances, and k, the number of clusters to form, a partitioning algorithm organizes the instances into k partitions. $k \leq n$, where each partition represents a cluster.

K means

K-means algorithm, an example of partitioning algorithms, is an efficient, effective, and simple clustering algorithm. K- Means partitions the data into K clusters. Centroid is a name given to cluster center.

Given K, provided by the user, this is how the algorithm works:

1. Creates K centroids randomly (based on the predefined value of K)

2. Allocates every instance in the dataset to the closest centroid (minimum Euclidean distances between an instance and centroid), meaning if an instance is near to one cluster's centroid than any other centroid, then that a data instance is considered to be in a particular cluster.
3. Recalculates the centroids by taking the mean of all data instances assigned to that centroid's cluster, hence reducing the total intra-cluster variance in relation to the previous step. The "means" in the K-means refers to averaging the data and finding the new centroid
4. Repeats Steps 2 and 3 until some stopping criteria is met I.e. No (or minimum) changes in centroids value or no (or minimum) re-assignments of data instances to different clusters, the sum of distances between the data instances and their corresponding centroid is minimized, a maximum number of iterations is reached.

Advantages

- Relatively simple to implement.
- Scales to large datasets.
- Guarantees convergence.
- Can warm-start the positions of centroids.
- Easily adapts to new examples.
- Generalizes to clusters of different shapes and sizes, such as elliptical clusters.

Disadvantages

- Difficult to predict and find the optimal K-Value.
- With global cluster, it didn't work well.
- Dependent on initial values, Different initial partitions can result in different final clusters
- Trouble clustering data with clusters (in the original data) of Different size and Different density
- Outliers can drag the Centroids can be dragged by outliers, or outliers might be Considered as a cluster instead of being ignored.
- Scaling with number of dimensions.

K-medoids

The k-medoids algorithm is a partitional clustering algorithm associated to the k-means algorithm and the medoid shift algorithm. It is also another well-known partitioning algorithm. The K- medoids and K-means algorithms behave in a very similar way. However, K- medoids, rather than having the centroid move using the mean distance of the instances, the centroid takes the position of the instance that is closest to the center. While K-means attempts to minimize the total squared error, k-medoids minimizes the sum of dissimilarities between points labeled to be in a cluster and a point designated as the center of that cluster. Because it minimizes a sum of general pairwise dissimilarities instead of a sum of squared Euclidean distance, k-medoids is more robust to noise and outliers as compared to k-means. Outliers are far away from the majority of the data, and thus, when assigned to a cluster, the mean value in k-means of the cluster can be distorted dramatically.

2.4.4 Hierarchical algorithms

Unlike partitioning algorithms, Hierarchical algorithms creates a set of nested clusters that are organized as a tree. In the tree, the root contains all the other cluster and each cluster is the union of its sub clusters. Hierarchical clustering is well suited to hierarchical data, such as taxonomies. Hierarchical clustering is categorized into two types, Divisive clustering and agglomerative clustering. Divisive clustering is also known as top-down approach. An agglomerative, on the other hand, bottom-up approach clustering. Divisive clustering merges all data instances in a single cluster and splits the cluster into two smaller clusters with least similarity whereas a cluster is split up into smaller clusters. Clustering starts with every data instance as a cluster itself and merges the objects or groups that are close to one another. CURE (Clustering Using REpresentatives), BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) and Rock (robust clustering algorithm for categorical attributes) are some of the examples of hierarchical clustering algorithm.

BIRCH

BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) is an integrated agglomerative hierarchical clustering method and designed for clustering large amount of metric data. It attempts to minimize the memory requirements of large datasets; therefore, it is mainly suitable when there is limited amount of main memory. The cluster representation in BIRCH is summarized using two concepts, clustering feature (CF) and clustering feature tree

(CF tree). A CF tree is a height-balanced tree that stores the clustering features for a hierarchical clustering. The algorithm pass through four phases:

Phase 1: Load data into memory: The algorithm starts with an initial threshold value, scans the data, and inserts points into the tree. If it gets out of memory before it finishes scanning the data, it increases the threshold value, and rebuilds a new, smaller CF-tree, Scan DB

Phase 2: Condense data(optional): Given that certain clustering algorithms perform best when the number of objects is within a certain range, it is possible to group crowded sub clusters into larger ones resulting in an overall smaller CF-tree. Data reduction is done by building a smaller CF tree. Most of the data removed are outliers.

Phase 3: Global clustering: Use existing clustering algorithm (like KMEANS but almost all algorithms can be adopted) on CF entries to categorize Clustering Features instead of data points. For example, KMEANS can be applied to categorize a data and BIRCH for minimizing I/O operations.

Phase 4: Cluster refining (optional and offline): corrects the problem with CF by providing additional passes over the data to fix inaccuracies caused by the fact that the clustering algorithm is applied to a coarse summary of the data.

2.4.5 Density-based

As most partitioning methods cluster instances based on the distance between instances, only spherical-shaped clusters are identified by those methods. The methods work well in compact, well separated clusters and dataset with less outliers and noise. Unfortunately, real life data can contain clusters of arbitrary shape such as oval, linear and "S" shape clusters, and many outliers and noise. Density based clustering fills this gap by discovering clusters of arbitrary shapes from a dataset containing noises and outliers. Clusters in density-based clustering are dense regions in the data space, separated by regions of the lower density of instances. Instances that are not part of a cluster are labeled as noise. To find clusters in dataset, three different clustering methods are used in this Clustering. The first one is Defined distance (DBSCAN which uses a specified distance to separate dense clusters from sparser noise. However, it is suitable only if there is a very clear Search Distance to use. This requires that all meaningful clusters have similar densities. The DBSCAN algorithm is the fastest compared to the other clustering methods. The second is Self-adjusting (HDBSCAN). It uses a range of distances to separate clusters of varying densities from sparser noise. This method is the most data-driven of the clustering

methods, therefore, it requires the least user input. Multi-scale (OPTICS) is the last one which uses the distance between neighboring features to create a reachability plot which is then used to separate clusters of varying densities from noise. The OPTICS algorithm offers the most flexibility in fine-tuning the clusters that are detected, though it is computationally intensive, particularly with a large Search Distance.

2.4.6 Grid-based

It is based on a multiple-level granularity structure. It explores multi resolution grid data structure in clustering. It partitions the data structure into a finite number of cells to form a grid structure and assign objects to the appropriate grid cell. From the cell in the grid structure, computes the density of each cell and Forms clusters from contiguous (adjacent) groups of dense cells by eliminating cells, with density is below a certain threshold. Clustering in grid based is fast because clustering is performed on summaries but not on individual objects, and it doesn't compute distance.

2.4.7 Model-based

Model-based clustering considers the data as coming from a distribution that is mixture of two or more clusters. Unlike hierarchical clustering algorithms, Partitioning algorithms and others, Model-based is based on formal models. A model is hypothesized for each of the clusters and the idea is to find the best fit of that model to each other. Clustering is also performed by having several units competing for the current object. The unit whose weight vector is closest to the current object wins. The winner and its neighbors learn by having their weights adjusted Model-based clustering is useful for visualizing high-dimensional data in two- or three-dimensional space.

2.4.8 Constraint-based Method

Constrained clustering is a class of semi-supervised learning algorithms. Semi-supervised clustering algorithms allow the user to incorporate a limited amount of supervision into the clustering procedure. Typically, constrained clustering incorporates either a set of must-link constraints, cannot-link constraints, or both, with a Data clustering algorithm. Both a must-link and a cannot-link constraint define a relationship between two data instances. Must-link (ML) constraints indicate that two instances should be in the same cluster, cannot-link (CL) constraints that they should be in different clusters. These sets of constraints act as a guide for

which a constrained clustering algorithm will attempt to find clusters in a dataset which satisfy the specified must-link and cannot-link constraints.

2.5 Cluster Quality Index

In application of clustering techniques, the evaluation of quality of clustering is an important issue. When clustering method is applied on a dataset, there should be some way to determine whether the result is good or bad. The process of evaluating the results of a clustering algorithm is known as cluster evaluation [8]. Clustering quality index is a tool to assess the quality of the clustering result [7, 8]

Cluster evaluation is performed for a number of reasons. Measuring clustering quality is primary reason. Clustering techniques are applied on a dataset and the goodness of generated result required to be assessed. Several measures can be exploited. Some methods measure how well the clusters match with the ground truth, if the truth is available, while others measure how well the clusters fit the dataset. There are also measures that score clustering and thus can compare two sets of clustering results on the same dataset [18].

Clustering indices can also be used to determine the number of clusters in a dataset. Some algorithms require the number of clusters in a dataset as input. Furthermore, the number of clusters can be regarded as an interesting and crucial summary statistic of a dataset. Therefore, it is desirable to estimate this number even before a clustering algorithm is used to derive detailed clusters [18].

Clustering analysis on a dataset is purposeful only when there is a nonrandom structure in the data. Simply application of clustering method on a dataset returns clusters; but the clusters mined may be misleading. To assess the existence of this nonrandom structure can be identified through cluster evaluation [18]. It is also be used to compare clustering algorithms and two or sets of clusters.

There are three approaches of Clustering validation: External, internal validation and relative criteria. In External validation, external information is used to perform the validation i.e. it validates if the cluster labels match externally supplied classes. Entropy is an example of external validation which evaluates the clusters based on given class labels. Internal validation, on the other hand, uses only the information on the data without any external information. The third approach of clustering validity is based on relative criteria, which consists of evaluating the results by comparing them with other clustering schemes [6, 19].

2.5.1 Internal clustering validation measures

This section introduces basic concepts of internal validation approaches. Generally, compactness a separation are two criteria that internal validation measures are based on. Compactness or Cohesion measures how closely related objects in the clusters are. Variance is a common measure of compactness. Lower variance indicates better compactness. Distance is used in numerous measures to estimate the cluster compactness. It can be maximum or average pairwise distance, and maximum or average center-based distance. Separation measures how distinct or well separated a cluster is from other clusters. It measures the distance between two different clusters. For example, the pairwise distances between cluster centers or the pairwise minimum distances between objects in different clusters are widely used as measures of separation. Also, measures based on density are used in some indices. Both compactness and separation are considered by most of the cluster indices in the way of ratio or summarization. Some indices consider only one [6, 19].

2.5.1.1 Silhouette index

The silhouette value is a measure of how close an object is to objects in its own cluster compared to objects in other clusters. The measure ranges between -1 and 1. When the measuring result is higher (result closer to 1), it shows the object is far away from the objects in neighboring clusters. When the value is negative, it indicates the object might be assigned in a wrong cluster. The object might be in the boundary if it the measuring result is 0. Any distance metric, such as the Euclidean distance or the Manhattan distance can be used to calculate the silhouette.

Assume any clustering technique is used to cluster the data and i is an object in cluster Ci. The Silhouette coefficient (*s (i)*) can be calculated as follows for $|ci|>1$,

$$S(i) = \frac{b(i) - a(i)}{\max\left(b(i), a(i)\right)} \tag{2.1}$$

a(i) is the average dissimilarity between object *i* and all the other objects within the same cluster ci. Therefore, the smallest value of *a (i)* indicates that i is well matched *D (i,j)* is the distance between i and *j*. As the distance between object with itself (i.e. *i=j*), is not considered, the average is calculated $|ci|$ -1.

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i,j) \tag{2.2}$$

B (i) is the minimum average dissimilarity between object *i* and objects in clusters other than ci. A large bi means object i is poorly matched with the neighboring. The cluster with minimum average is the next best fit cluster for object *i*. It is said to be the neighboring cluster.

$$b(i) = \min_{i \neq j} \frac{1}{|C_j|} \sum_{j \in C_j} d(i,j), Ci \neq 0 \tag{2.2}$$

Equation 2.2 can also be rewritten as:

$$\begin{cases} 1 - a\,(i)/b\,(i), if a(i) < b(i) \\ 0, if a(i) = b(i) \\ b\,(i)/a\,(i) - 1, if a(i) > b(i) \end{cases} \tag{2.3}$$

From the definition above, the value of *s (i)* lies between -1 and 1.

$-1 \leqslant s(i) \leqslant 1$

To measure of how appropriately the data have been clustered or how all points are tightly grouped the average *s (i)* over all point of cluster can be used.

2.5.1.2 Calinski-Harabasz index

Calinski index based on the two measures separation and compactness. It is computed by

$$CH = \frac{BSS_K - (K-1)}{WSS_K - (N-K)} \tag{2.4}$$

Where N and K are the total number of objects and number of clusters respectively. BSSk is between cluster sum of square which indicates the separation measure, whereas compactness is computed by WSSk (with in cluster sum of square). The purpose of the index is to find k which maxmizes Calinski index.

2.5.1.3 C index

The c index is calculated based on three quantities. The first quantity is the sum of distances over all pair of objects from the same cluster *(d_w)*. The second is the sum of the smallest distances between all the pairs of objects in the entire dataset *min(dw)*. *Max (dw) is* the sum of the largest distances between all the pairs of objects in the entire dataset. Using these three quantities, the c index is defined as follows:

$$C_{index} = \frac{d_w - min(d_w)}{max(d_w) - min(d_w)} \tag{2.5}$$

The result lays between 0 and 1.

2.5.1.4 Davies-Bouldin index

Davies-Boludin index is well known for its better partition capability [20]. The Davies-Bouldin index is defined as follows:

$$DB = \frac{1}{K} \sum_{i=1, i \neq j}^{K} max(\frac{d_i + d_j}{d(c_i, c_j)}) \tag{2.6}$$

Where DB is Davies-Bouldin index calculated by averaging each pair of clusters as shown in the equation 2.7. K is the total number of clusters, d_i and d_j are the average distance of all object in each cluster. C_i and C_j are the center of cluster *i* and *j*.

2.5.2 External clustering validation measures

As stated in the introduction of this section, external indices perform validation using some predefined knowledge like class label or number of clusters. In this case, a cluster structure is considered as good cluster structure if it's same as predefined class structure in the dataset.

2.6 Cluster evaluation on Big Data

As the traditional indices require high computational cost and inability to be parallelized, BD-CVIs [9]. approximates the traditional indices. Two traditional indices, Silhouette and Dunn

(with two highest ranking based on statistical analysis performed to check a significant difference exist among the effectiveness of multiple CQI), are selected. BD-Silhouette is defined as the ratio between the difference of the inter-cluster and intra-cluster, and the maximum of the two. The average distance between global centroid and the centroid of each cluster is considered for calculation of inter-cluster. Whereas for the intra-cluster is the average of the distances between each point to the centroid of the cluster to which it belongs. The difference between the traditional Silhouette and BD- Silhouette lies in the intra-cluster in such a way that the traditional consider the average distance between the points that belong to the same cluster. Like the traditional Silhouette, the result of BD-Silhouette lies between -1 and 1. The return value is – 1, if there is only one cluster for the whole dataset. The larger number of clusters the return value holds 1. The first maximum is considered as the optimal number of clusters.

The traditional Dunn works on the minimum distance between the centroids and the maximum distance between all the points that belong to the same cluster. BD-Dunn facilitates the original Dunn index computation in such a way that it could be easier in Big Data. BD-Dunn is the ratio between the minimum of the distances from the centroids to the global center and the maximum of the distances from each point in the set to its centroid. As it does not have to calculate in the denominator the distance between each pair of points of the dataset, it simplifies the computations.

Chapter Three

Big Data and Cluster Quality Index Computation

This chapter briefly discusses the proposed approach that have been used to evaluate Big Data clusters. As a big size clustered dataset cannot be directly analyzed using the traditional cluster quality indices, we introduce an approach to reduce the size of the dataset to the size that can be processed through traditional clustering quality indices in a main memory. The sampling is performed by dividing the clustered dataset into small equal size cells and applying specific criteria on each cell to generate a new instance out of the instance in the cell. The sampled dataset is described with all the features inherited from the original dataset and additional weight feature which describe the value of the instances in the original dataset. The sampling method introduced is named as “smart sampling”.

3.1 Fundamental concepts

Generally, for a dataset to be evaluated using any sampling method, it may go through four steps. Figure 3.1 shows the general workflow of the Big Data clustering evaluation process considering sampling method that has been performed in this work.

The preprocessing step involves transforming the input dataset into a useful and efficient format. A dataset may contain errors, duplication within data and outliers. A Clean dataset is produced by Ignoring or removing missing and duplicated records, correcting erroneous records. For simpler and feasible analysis processes, all datasets are transformed to a similar format corresponding to the format used in clustering and evaluation implementation.

Using a clean structured dataset as an input, clustering analysis is performed in order to be applied in the clustering evaluation process. In this work, the k-mean is selected as clustering tool. Clustering analysis is not in the scope of this work. However, some of the datasets available are not clustered and it is important to perform cluster analysis to generate a clustered dataset to be used as an input to the sampling process.

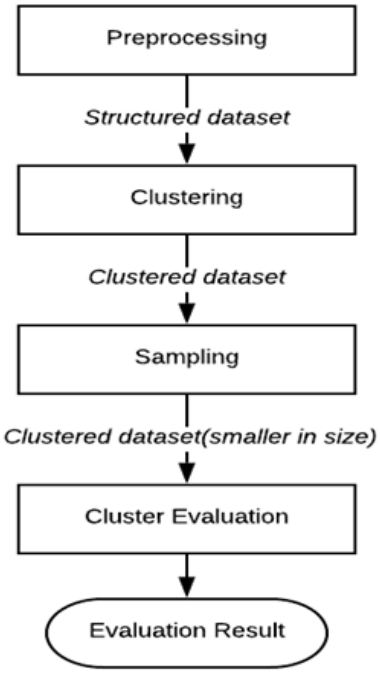

Figure 3.1 General architecture of sample-based Big Data cluster evaluation

The sampling process is a process of reducing the size of Big Data clustered dataset in such a way that the traditional cluster quality index algorithms be able to execute it. Fundamentally, two sampling methods are used in this work. The first one is random sampling that selects a given percentage of the instances from each cluster arbitrarily and forms smaller size dataset. The other is the smart sampling method which places instances in the cell, verifies each instance in a cell are from a same cluster and considers the center of a cell as a representative of all the other instances instead of considering all. The cell containing instances from different cluster passed to the sample dataset with all instances. Section 3.2 describes the smart sampling in detail with examples.

The last part of the process is cluster evaluation. This evaluates the sampled cluster dataset using the traditional cluster quality indices. During the random sampling, the features of the original dataset is not modified, or no additional feature is included on the sampled dataset. Therefore, it is possible to evaluate the random sampled dataset using traditional clustering indices directly. On the other hand, an adopted silhouette cluster quality index is required in order to support the newly generated sample using smart sampling. The adopted silhouette considers the weight feature in the distance measurement. The brief explanation of the adopted silhouette index is included in Section 3.3.

3.2 Smart sampling using Spark

Smart sampling is a systematic transformation of a large clustered dataset into a new smaller dataset in order to process the smaller size dataset into an adopted centralized silhouette measure. It partitions the cluster space into cells and visits each instance in each cell to check if it meets a merging criterion. The merging criteria expects all instances within the cell to be from the same cluster. Instances that meets the merging criteria grouped together and passed to sampled dataset with information such as cluster id, weight and modified feature values. The modified value holds common value to all the instances in the cell i.e. the center of the cell.

Cells are squares formed using vertical and horizontal line across the cluster space. Different cell size may produce different sample size. The total number of instances produced depends on the cell size and the cluster distribution. An increasing or decreasing cell size has no relation with the increase and decrease of sample size. A very large cell size may produce same number of instances as the original dataset as all the cells may contain mixed cluster instances. A very small cell size may as well give the same result as each cell may contain a single instance. The optimal cell size for a given dataset lies between 0 and the maximum value of features to get the required sampled size. The input of the sampling process is the cell size and the clustered dataset to be sampled. Each instance of the sampled dataset has a weight feature in addition to cluster id and features in the original dataset. The sampling process iterates through each cell to assess the instances inside and terminates after a visit of the last cell in the space. The workflow of the smart sampling technique shown in Figure 3.2 are the following:

STAGE 1 — Define cell identifier — each cell should be uniquely identified. This helps to recognize the cell in which an instance belongs. For simplicity, the cells are assigned a value holding the center of the cell. Central value of the cell is determined by the cell size and the value of the edges of the cells.

STAGE 2 — Assign cell identifier to instances — based on the feature values of an instance, it is possible to determine to which cell identifier the instance fits. This stage discovers the instance's cell and assign cell identifier to the instances in order to recognize its' container cell. After this stage, each instance has a temporary feature: cell identifier.

STAGE 3 — Initialize instance weight — this stage initializes each instance an initial weight value. The modified silhouette requires weight to find a distance between two instances. Therefore, it is important to assign every instance an initial weight which possibly be modified in the next stages of sampling process. All instance has weight value one in the beginning (right after

this STAGE). At this point an instance is defined with its initial weight in addition to its cell identifier, cluster identifier and feature values.

STAGE 4 — Place instance in cell— the instances are compared using cell identifiers and instances of the same cell identifier grouped together. In this stage, the dataset is represented with cell identifier and group of instances within the cell (each instance with all the content defined in stage three) as a key and value respectively.

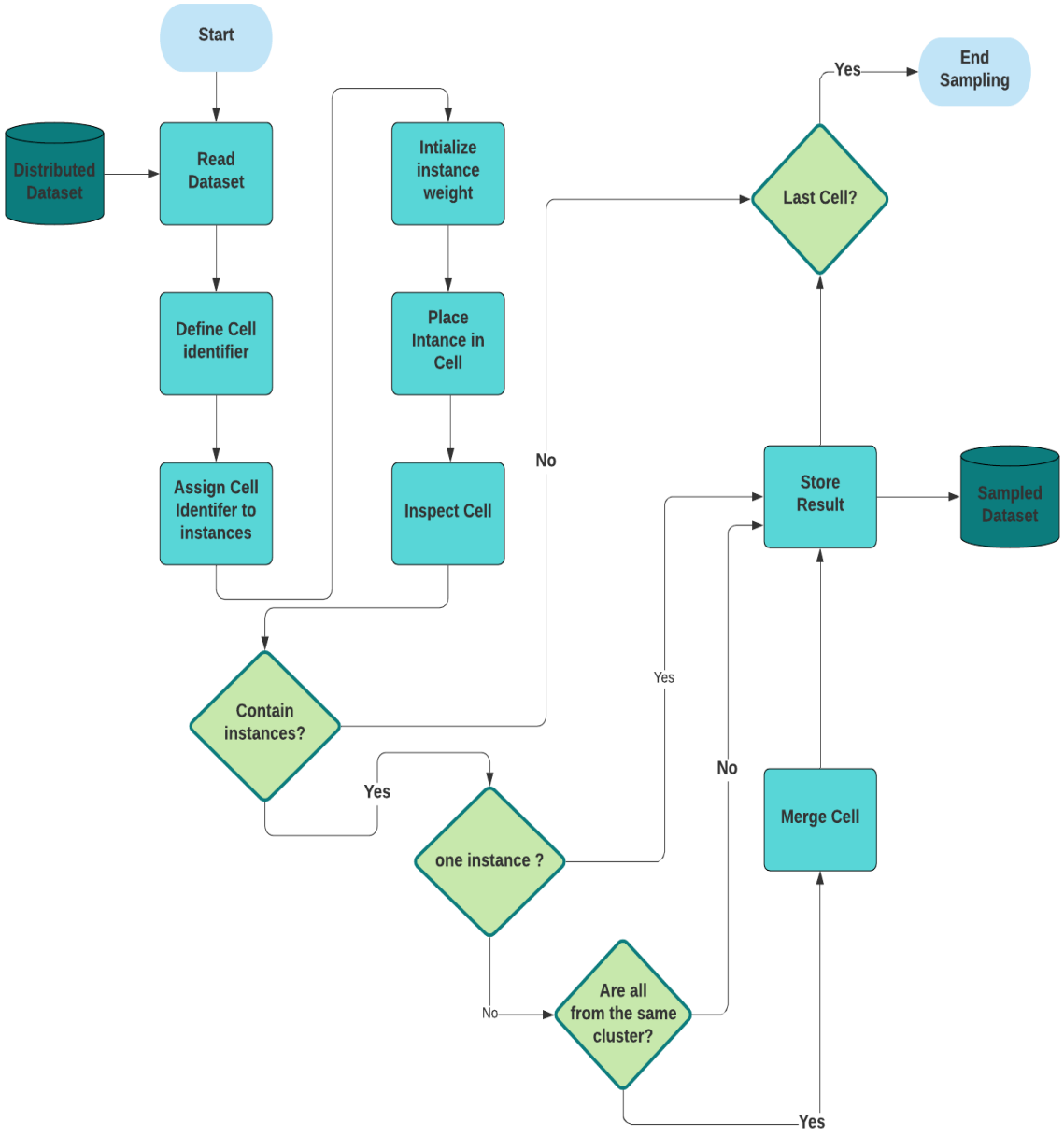

Figure 3.2 The workflow of Smart Sampling using Spark

STAGE 5 — Inspect cell — this stage checks if a cell contains more than one instance. If it has only one instance, the instance is stored to the sampled dataset with no change. For those cells that contain more than one instance, the merging criteria is checked. The merging criteria controls homogeneity and heterogeneity of a cell. A cell is homogeneous if all the instances are from the same cluster, heterogeneous otherwise. If the cell fulfills the criteria, merge instances stage receives all the instances in the cell to follow the merging procedure. Instances of heterogeneous cells are passed to the sampled dataset with the initial weight assigned at STAGE 3(no modification is done on the fields of the instances).

STAGE 6 — Merge cell — only homogeneous cells are passed to this stage. A new instance is created with updated information to replace all the instances in the cell. The weights of the instances in the cell is summed up to generate a weight for the newly generated instance. The feature values of the new instance are the value which represents the center of the cell and the cluster id is the cluster id of the instance which is commonly shared by all the instances.

STAGE 7 — Store Result — discard the cell identifier attached to the instances as group and store the instances information into the sampled dataset. It iterates through instances in the cell that need to be stored and store all instances one by one.

To make sampling process clear, Figure 3.3 reports an example. The dataset used in the example has 36 instances of two clusters; cluster1 and cluster2 are represented in the Figure 3.3 using nineteen white and seventeen black points respectively. Initially each instance is described through its feature values and cluster identifier. An instance $i = [x, y, Cr_{id}]$ belongs to cluster Cr_{id} with features values x and y.

A cell is identified by its identifier. For instance, a cell surrounded by lines of edges (0, 0), (0, 1), (1, 0) and (1, 1) has an identifier as (0.5, 0.5). Cell identifier is given to the cells in the first stage of the sampling process (see Figure 3.2). Then, every instance is linked to its' cell identifier, i.e. cell center Cl_{id} is added as additional field of the instance; $i = [x, y, Cr_{id}, \boldsymbol{Cl_{id}}]$ for cell size of s. In the Figure 3.3, there are sixteen with cells of size $s = 1$ with four columns and four rows. Only seven of these cells contain instances and the rest are empty cells. Cells with instances are one in the first and fourth row, two in the second row and three third row.

In stage three, the content of the instances includes the weight field w; $i = [x, y, Cr_{id}, Cl_{id}, \boldsymbol{w}]$. All instances are initialized with weight $w = 1$ in this stage. When instances are placed in the cell (stage four), instances with the same cell identifier grouped together. As a result, a unique list of cell identifier with the corresponding instances is generated.

$$D = \{\{c_1, [i_1, i_2, i_3...]\}, \{c_2, [i_4, i_7, i_8...]\}, \{c_3, [i_9, i_7, i_8....]\} ... \{c_n, [i_3, i_7, i_8...i_m]\}\}$$

where D is the temporary dataset generated, c_1, c_2 and c_n are n cell identifiers, $i_1, i_2, i_3...i_m$ are m instances of the original dataset dispersed inside the cells, m is not equal to the total number of instances in the original dataset as instances on the border of the cell are not a part of the cell.

During the inspection of the instances (see Figure 3.2), the cluster identifiers of instances in a cell are compared and similar instances be merged. Suppose D_{mr} is the dataset after the merge is applied, c_1 and c_3 are the cell with instances from the same cluster, i_{mr1} and i_{mr2} are the newly generated instances after a merge is applied on c_1 and c_3 respectively:

$$D_{mr1} = \{\{c_1, i_{mr1}\}, \{c_2, [i_4, i_7, i_8...]\}, \{c_3, i_{mr2}\} ... \{c_n, [i_3, i_7, i_8...i_m]\},$$

$$i_{mr} = [x, y, Cr_{id}, Cl_{id}, w],$$

Where x and y are the value of Cl_{id}, Cr_{id} is the cluster id of the merged instances in the cell, w is the sum of the weights of all instances in the cell.

Instances are extracted from the group and represented in the weight-based format before storing both merged and un-merged instances into the sampled dataset. Fundamentally, it takes a form of $[x, y, Cr_{id}, w]$.

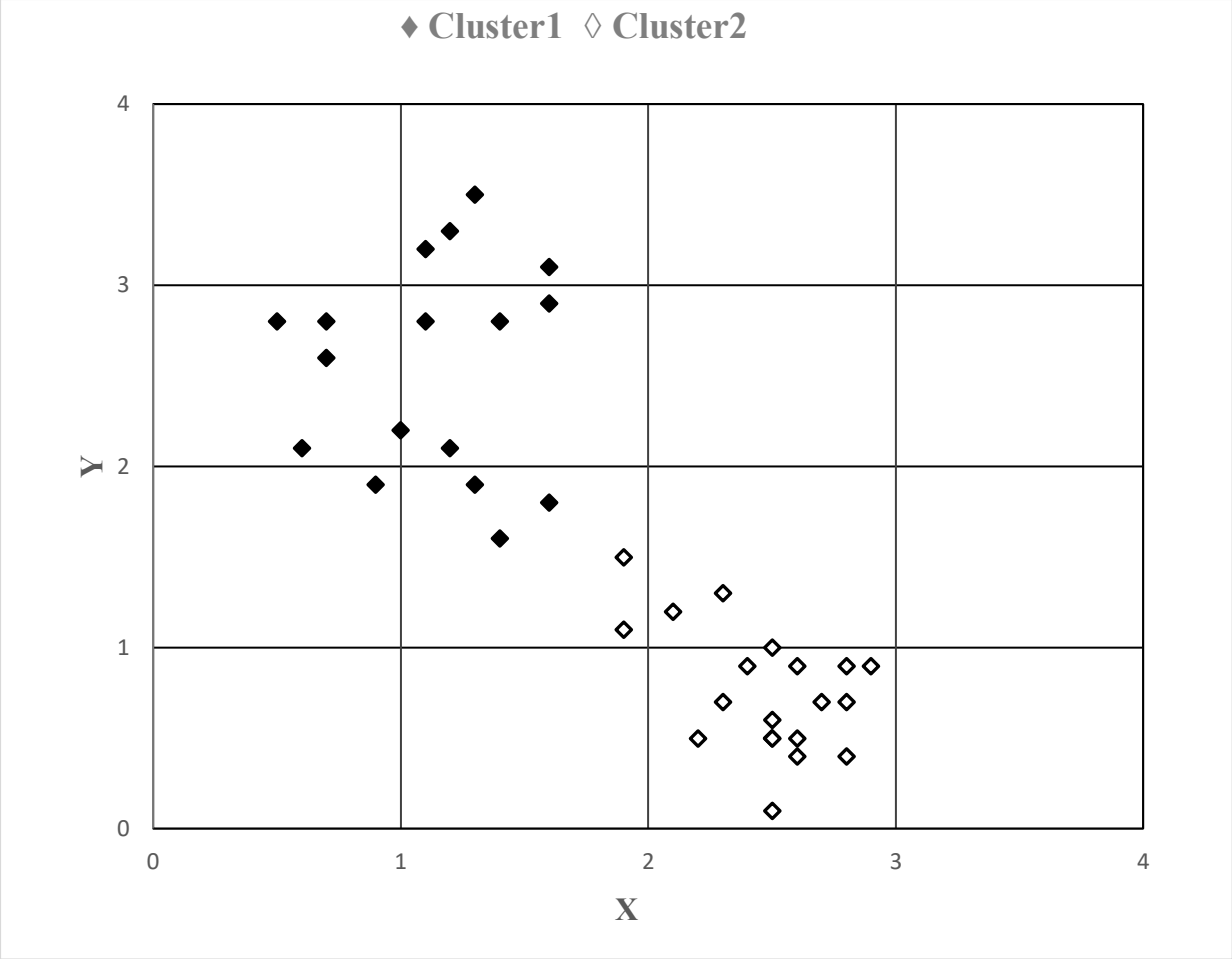

Figure 3.3 An example x-y cluster graph with 36 instances of two clusters

Table 3.4 shows 14 instances of sampled dataset generated from 36 instances shown in Figure 3.3. In the table reported in Table 3.1, number 6 – 11 show the instances placed in cell centered in (1.5, 1.5) are heterogeneous cell i.e. contains a mixed instance of three and two instances from cluster1 and cluster2 respectively. Therefore, each instance of this cell is transformed with its original feature values and cluster identifier and the initialized weight value $w = 1$ as weight is required for measurement of the distance of the adopted version of silhouette index. In addition, the cell contains one instance (number 14) and those instances that are on the border of the cell (number 12 and 13) are passed as it is.

Other cells, except the empty cells, have points from the same cluster so all passed through merge stage (Stage six, see Figure 3.2). Basically, a new instance is generated for cells centered in (0.5, 2.5), (1.5, 2.5), (2.5, 1.5), (2.5, 0.5) and (1.5, 3.5). The new instance has a format [*x, y,* Cr_{id}*,* w]. *x and y* are replaced cluster identifier $Cl_{id;}$ point (0.5, 2.5), (1.5, 2.5), (2.5, 1.5), (2.5, 0.5) and (1.5, 3.5) are present values of x and y for the five cells centered accordingly. The

weight *w* of the cells centered in (0.5, 2.5), (1.5, 2.5), (2.5, 1.5), (2.5, 0.5) and (1.5, 3.5) are 4, 3 , 2 , 14 and 4 respectively, which are the total number of instances in each cell.

NO	Cluster id (Cr_{id})	X value	Y value	Weight (w)
1	1	1.5	3.5	4
2	1	1.5	2.5	3
3	1	0.5	2.5	4
4	2	2.5	1.5	2
5	2	2.5	0.5	14
6	1	1.3	1.9	1
7	1	1.4	1.6	1
8	1	1.6	1.8	1
9	1	1.2	1.9	1
10	2	1.9	1.1	1
11	2	1.9	1.5	1
12	2	2.5	1.0	1
13	1	1.0	2.2	1
14	1	0.9	1.9	1

Table 3.1 Smart sampled dataset from a dataset in Figure 3.3(with cell size s = 1)

As traditional centralized version of CQIs are not capable of analyzing large datasets, in this manner, a “smaller” size sample dataset is generated from a large size dataset. The transformed dataset is an input to the compute silhouette measure. The silhouette coefficient is calculated using the newly generated compact representation of the original dataset considering the weight feature. Section 3.3 discusses the adopted version of the silhouette measure, which considers also the weight feature.

3.3 Silhouette index for Big Data cluster

Based on the sample generated using smart sampling (see Section 3.2), the equation for the traditional silhouette index is adapted in a way that it includes the weights attached to each instance during the sample generation. Basically, when the Euclidian distance between two instances is calculated the weight of the instances are multiplied on the result.

The Euclidean distance between instances are calculated as square root of the sum of the squares of the differences between i and j in each dimension.

Considering two dimensional instances,

$$d(i,j) = \sqrt{(j_1 - i_1)^2 + (j_2 - i_2)^2} \tag{3.1}$$

Where instance $i = \{i_1, i_2\}$ and j= $\{j_1, j_2\}$

The distance between two instances with the corresponding weight assigned during sample generation is the weight of the instances multiplied with the sum of the squares of the differences between *i* and *j*. The weighted distance alters both the average dissimilarity between object *i* and all the other objects with in the same cluster (inter-cluster) and the minimum average dissimilarity between object *i* and objects in clusters other(intra-cluster) i.e. the *a(i)* and *b(i)* in the silhouette equation (see Eq 2.1). The modified version of the Euclidean distance equation stated in (Eq. 3.1):

$$d(i,j) = \sqrt{\boldsymbol{w_i}.\boldsymbol{w_j}[(j_1 - i_1)^2 + (j_2 - i_2)^2]} \tag{3.2}$$

Where w_i and w_j are weight associated to instance i and j respectively.

Chapter Four

Experimental Results

This chapter gives a brief explanation of the experiments performed during this work in three different section. The first section of this chapter discusses the experiments on two small size datasets on a random sample and the second and the third section compare silhouette index of the proposed sampling approach (smart sampling) , the random sampling and original dataset with manually clustered and automatically clustered larger(compared to the dataset used on first section, see Section 4.1) dataset.

4.1 Experiment on random sample

This part of the experiment discusses a comparison silhouette index result on a dataset and random samples taken from it. The experiment is mainly conducted to examine the performance of the random sampling using three different percentage of the original dataset (10%, 30%, and 50%) on four different number of cluster (2,3,4 and 5) as compared to the original dataset.

Random Cluster sampling extracts a given percentage of instances from each cluster and generates new sample dataset by combining the samples of each cluster. For instance, considering clustered dataset of two clusters and 50% sample generation, 50% of cluster1 and 50% of cluster2 are generated to be merged to form the final randomly sampled dataset. As a result of random sample varies on each execution, it is important to perform number of executions on each percentage of a sample.

The first dataset considered in this experiment, named as DS750, has 750 instances with 90 features each. The original dataset (DS750) is clustered using k-mean and the generated clustered dataset is evaluated using the silhouette index. An average of five different runs is generated on each percentage of the sample. The obtained result is shown in Table 4.1. According to the result on DS750, it is possible to tell that four is the possible number of clusters for this dataset (the higher the silhouette indexes the better the clustering result). On the other hand, all the three samples indicate the better number of clusters as three. The results show no correlation between the increase or decrease of percentage of the sample and the performance of sampling procedure.

Number of clusters	Silhouette value			
	Original dataset	50% Sampled dataset(average)	30% Sampled dataset(average)	10% Sampled dataset(average)
2	0.28349128	0.276251187	0.29251904	0.254819397
3	0.40959278	0.344200123	0.355857657	0.33142755
4	0.42755792	0.33129269	0.335203993	0.304830573
5	0.29263905	0.27820566	0.28275602	0.270626493

Table 4.1 Experiment using random sampling on DS750

As the dataset used on first experiment shown on Table 4.1 is quite small, second experiment on a larger dataset (DS2310, see Appendices for detail) with 19 features and 2310 instances have done. Also, in second experiment, 50%, 30% and 10% of the dataset are extracted for 2, 3, 4 and 5 number of clusters. Each result of the sample is an average of five different execution as random sample generate different result on each run. The computed silhouette result by using all instances of the dataset and three different sample (10%, 30%, and 50%) is shown Table 4.2.

Based on the result, all the three samples show the higher silhouette index when cluster number is three which indicate the same possible number of cluster (i.e. 3) as of the original dataset. The second possible number of clusters, according to the original dataset is two, which is also true for all the three different percentage of samples. Similar to the result shown in DS750, an increase or decrease of the size of the sample and the performance show no significant relation for second experiment as well.

Number of clusters	Silhouette value			
	Original dataset	50% Sampled dataset(average)	30% Sampled dataset(average)	10% Sampled dataset(average)
2	0.47094202	0.470076787	0.46387185	0.45136499
3	0.5375713	0.477187833	0.48219241	0.48486719
4	0.45918366	0.429315367	0.44714727	0.40220799
5	0.4053393	0.3938975	0.41212885	0.41400546

Table 4.2 Experiment using the random sampling on DS2310

4.2 Experiment on manually clustered dataset

This part of the experiment illustrates results on three manually clustered datasets named DS18000, DS6500 and DS3000 containing 18000, 6500 and 3000 instances respectively (see appendix for detail). Two different Sampling method are performed on the original datasets. The first one is the random sampling which selects specific percentage of the original dataset randomly. The second sampling method experimented on this section of the experiment is the smart sampling which is the approach implemented in this work using spark. It allows a comparison between the smart sampled dataset with the random sampled and original dataset based on silhouette index.

As reported in Table 4.3, DS6500 and DS3000 show appropriately clustered datasets with silhouette index closer to one on both the original and the random samples. In DS18000, the silhouette index indicates more than average on distribution of instances for a specified number of clusters (i.e. 5) on the sampled datasets. Due to its size, experiment performed on DS18000 is only on the sampled datasets: random and smart sampled. The silhouette index on the smart sampled dataset and original dataset is quite similar in dataset DS6500; original dataset is slightly better. However, the number of instances of the sampled cluster (with the cell size s = 5) is less than a one-half of the original dataset. The same holds true for the dataset DS3100. Generally, silhouette index of smart sampled and original dataset on, all the three datasets, are similar which indicates the smart sampling can be considered as a solution for the cluster evaluation of huge size dataset.

The random sample is generated by computing the average of multiple results. As it is randomly selected, a single execution may show a better or less performance than the original dataset and/or the smart sampled dataset. From those average results of three different percentage(10%, 30% and 50%), the one that is closer to the smart sample dataset (based on the size of the sample) is selected. The random sampled dataset results show somewhat better silhouette index than both the original and smart sampled dataset. However, except for dataset DS18000, the number of instances of the spark sampled datasets are fewer in number than the random sampled dataset. It is also important to consider that finding the average silhouette index on random sample requires many executions for a single dataset as results vary on each run. On the other hand, the smart sampling is executed one time if the generated sample size is small enough to be executed in a centralized manner. This is a good point for the smart sampling.

Original Dataset Name	Number of clusters	Dataset	Sample size	Minimum distance	Maximum distance	Silhouette index
DS6500	8	Original	6500	4.47	439294.47	0.90459067
		Random sample (50%)	3250	5.0	439294.47	0.90532905
		smart sampled (cell size=5)	2448	500.0	439270.12	0.89975625
DS3100	31	Original	3100	0.0022993088	33.056683	0.88470066
		Random sample (50%)	1550	0.0051435432	33.056683	0.8872491
		smart sampled (cell size=5)	1476	0.0022993088	69.057945	0.8671787
DS18000	5	Original	18000	-	-	-
		Random sampled (30%)	5445	1.0	1238.1163	0.66860116
		smart Clustered (cell size=5)	6297	5.0	1241.8378	0.60396504

Table 4.3 Experiment using random and smart sample on three manually clustered dataset

In DS8000, the silhouette index indicates average result on distribution of instances for specified number of clusters which is closer to 0.5. The silhouette index of the smart sampled dataset, the random and original dataset is quite similar; random sampled dataset is slightly better. However, the number of instances of the smart sampled cluster (with the cell size $s = 5$) is less than a one-half of the original dataset. The random sample dataset shows even a better performance than the original dataset.

Original Dataset Name	Number of clus-ters	Dataset	Sample size	Minimum distance	Maximum distance	Silhouette index
DS8000	5	Original	8000	0.012992859	679.15796	0.5575914
		Random sam-ple (50%)	3999	0.05771768	674.44666	0.5576016
		smart sampled (cell size=5)	3449	0.302002	682.3672	0.5476162

Table 4.4 Experiment using random and smart sample on auto-clustered dataset

Chapter Five

Conclusion and Feature work

5.1 Conclusion

In this thesis, a technique for evaluating clustered big datasets has been designed, developed and tested. As traditional CQIs have processing data size limitation, the proposed technique generates a smaller size dataset in a "smart" way that the newly generated dataset can be processed by using traditional CQIs. It requires a clustered large size dataset as input and generates the smaller size sampled clustered dataset with an additional weight field given to each instances of the new sampled dataset. Silhouette CQI is selected and adopted to support the weight feature that is included in the sampling process. In order to perform the testing process, k-mean clustering method is used for the dataset that is not clustered. The implementation of the smart sampling technique is carried out using the Spark big data framework.

The smart sampling technique accepts the clustered dataset and analyzes each instance in each cluster by placing instances in cells (small groups). It iterates though the cells to apply a particular criterion either to create one new instance out of the instances in each cell or pass all instances as it is to the sampled dataset. The criteria states instances in a cell can only be reduced if the cell contains only instances from the same cluster. Based on this important rule, a reduction is performed on the cells. A reduced cell has single instance with new features generated from the values of initial instances of the cell. Fundamentally, the sampled dataset is composed of features from the original dataset including cluster identifier plus the weight feature added through the sampling process. On the other hand, the features of the reduced cell are modified considering all the instance in the cell i.e. a center of the cell. As the sampled dataset has an additional weight feature, the designed evaluation method adopts silhouette CQI in such a way that the weight of each instance is included in intra-cluster and inter-cluster distance computation.

The experiment was conducted in two test cases: manually clustered and auto-clustered datasets. In the manually clustered dataset, three datasets containing 18000, 6500 and 3000 instances with 5, 8 and 31 number of clusters respectively are considered. A Single dataset is used in the second test case (the auto-clustered dataset) which has 8000 instances. The second

test case uses the k-mean clustering method to generate clustered dataset in order to give it as an input to the smart sampling and the random sampling. Both cases of the experiment compare the silhouette index of the original dataset, randomly sampled dataset and the smart sampled dataset. Generally, silhouette index of smart sampled and original dataset on, all the three datasets, are similar with a slight better index of the original dataset. This indicates the smart sampling can be considered as a solution for the cluster evaluation of huge size datasets. The random sample results indicate better result than both the smart sampled and the original dataset. However, it is also important to consider that finding the average silhouette index on random sample requires the number executions for a single dataset as results vary on each run. On the other hand, the smart sampling is executed one time if the generated sample size is small enough to be executed in a centralized manner. This is a good point for the smart sampling.

5.2 Future work

This work can be extended in many ways that may increase the performance and upgrade the cluster evaluation of big data. This may also be choosing other methods which improve the cluster evaluation process. The following points are suggested as future work.

- In this work, the smart sampling approach is tested only for two-dimensional datasets. The performance of this sampling technique still needs to be experimented for multi-dimensional datasets.
- The sample generated using the smart sampling is only checked on one cluster quality index (silhouette index). Considering the performance seen in the experiments, it is highly recommended to run experiment with other CQIs with the same sampling method.

Bibliography

[1] M. Chen, S. A. Ludwig e K. Li, Big Data Management and Processing, K. Li, H. Jiang e Y. Z. Albert , A cura di, 2017.

[2] S. R. Youssra Riahi, «Big Data and Big Data Analytics: Concepts, Types and Technologies,» International Journal of Research and Engineering, vol. 5, pp. 524-528, 2018.

[3] T. Pang-Ning, S. Michael e K. Vipin, Introduction to Data Mining, (First Edition), Boston,: Addison-Wesley Longman Publishing Co, 2005.

[4] R. Clemens, F. Peter, G. Robert e D. Rudolf, Statistical Data Analysis Explained: Applied Environmental Statistics with R, 2008.

[5] E. Rendón, I. Abundez, A. Arizmendi e E. M. Quiroz, «Internal versus External cluster validation,» INTERNATIONAL JOURNAL OF COMPUTERS AND COMMUNICATIONS, vol. 5, n. 1, pp. 27 - 34, 2011.

[6] Y. Liu, Z. Li, H. Xiong, X. Gao e J. Wu, «Understanding of Internal Clustering Validation Measures,» IEEE International Conference on Data Mining, pp. 911 - 916, 2010.

[7] J. Hämäläinen, S. Jauhiainen e T. Kärkkäinen, «Comparison of Internal Clustering Validation Indicesfor Prototype-Based Clustering,» MDPI algorithms, 6 September 2017.

[8] L. Guerra, V.Roblesb, C.Bielza e P. L.˜naga, «A comparison of clustering quality indicesusing outliers and noise,» Intelligent Data Analysis, p. 703–715, 2012.

[9] J. M. Luna-Romera, J. García-Gutiérrez, M. Martínez-Ballesteros e J. C. R. Santos, «An approach to validity indices for clustering techniques in BigData,» 2017.

[10] J. Dean e S. Ghemawat, «Mapreduce: simplified data processing on large clusters,» Commun. ACM , pp. 107 -113, 2004.

[11] Apache, «https://spark.apache.org/,» Apache. [Online]. [Consultato il giorno 20 Aprile 2018].

[12] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker e I. Stoica, «Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing,» in NSDI, san Jose, CA, 2012.

[13] S. Hari, «https://hackr.io/,» 19 October 2019. [Online]. Available: https://hackr.io/blog/hadoop-vs-spark. [Consultato il giorno 5 November 2019].

[14] Z. Btissam, A. Ait Lahcen e M. Salma, «Big Data Clustering: Algorithms and Challenges,» in International Conference on Big Data, Cloud and Applications, Tetuan, Morocco, 2015.

[15] H. S. Deshmukh e P. L. Ramteke, «COMPARING THE TECHNIQUES OF CLUSTERANALYSIS FOR BIG DATA,» IJARCET, vol. 4, n. 12, December 2015.

[16] R. Suganya, M. Pavithra e P. Nandhini, «Algorithms and Challenges in Big Data Clustering,» International Journal of Engineering and Techniques, vol. 4, n. 4, pp. 40 - 47, July - August 2018.

[17] H. J. Watson, «Tutorial: Big Data Analytics: Concepts,Technologies, and Applications,» vol. 42, pp. 1247-1268, April 2014.

[18] H. Jiawei, K. Micheline e P. Jian, Data MiningConcepts and Techniques, Third a cura di, Waltham, MA, 225 Wyman Street: Morgan Kaufmann, 2012.

[19] R. ERÉNDIRA, A. I. M., G. CITLALIH, Z. S. DÍAZ, A. ALEJANDRA, Q. E. M. and H. E. ARZATE, "A comparison of internal and external cluster validation indexes," Applications of Mathematics and Computer Engineering, 2015.

[20] «CLUSTER VALIDITY MEASURES DYNAMIC CLUSTERING ALGORITHMS,» ARPN Journal of Engineering and Applied Sciences , vol. 10 , pp. 4009 - 4012, MAY 2015.

[21] S. C. N., S. L. P. D. e P. Sudhakar, «Evaluating and Analyzing Clustersin Data Miningusing Different Algorithms,» A Monthly Journal of Computer Science and Information Technology, vol. 3, n. 2, p. 86–99, February 2014.

Appendices

This is information about the dataset used in the experiment is given in the table. It includes the source of the dataset sources and description.

Name	Number of Instances:	Number of features	Number of Clusters	Cluster type	Source	Description
DS8000	8000	2	5		G. Karypis, E.H. Han, V. Kumar, CHAMELEON: A hierarchical 765 clustering algorithm using dynamic modeling, *IEEE Trans. on Computers*, 32 (8), 68-75, 1999.	
DS6500	6500	2	5		M. Rezaei and P. Fränti, "Set-matching measures for external cluster validity", *IEEE Trans. on Knowledge and Data Engineering*, 28 (8), 2173-2186, August 2016. (Bibtex)	
DS3100	3100	2	31		C.J. Veenman, M.J.T. Reinders, and E. Backer, A maximum variance cluster algorithm. *IEEE Trans. Pattern Analysis and Machine Intelligence* 2002. 24(9): p. 1273-1280.	
DS18000	18000	2	5			
DS720	720	90				
DS2310	2310	19	NOT KNOWN		Creators: Vision Group, University of Massachusetts Donor: Vision Group (Carla Brodley, brodley '@' cs.umass.edu)	Image data described by high-level numeric-valued attributes. The instances were drawn randomly

					Site : https://archive.ics.uci.edu/ml/datasets/Image+Segmentation	from a database of 7 outdoor images. The images were handsegmented to create a classification for every pixel